A CONSISTENCY-BASED SYSTEM
FOR KNOWLEDGE BASE MERGING

by

Daphne Hao Liu
B.Sc. Honours, Simon Fraser University, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

© Daphne Hao Liu 2006
SIMON FRASER UNIVERSITY
Summer 2006

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
APPROVAL

Name: Daphne Hao Liu

Degree: Master of Science

Title of thesis: A Consistency-Based System for Knowledge Base Merging

Examining Committee: Dr. Arthur Kirkpatrick
Assistant Professor of Computing Science
Chair

Dr. James Delgrande
Professor of Computing Science
Senior Supervisor

Dr. Torsten Schaub
Adjunct Professor of Computing Science
Supervisor

Dr. Oliver Schulte
Associate Professor of Computing Science and Philosophy
Examiner

Date Approved: August 11, 2006
DECLARATION OF
PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection, and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
Abstract

The ability to change one's beliefs consistently is essential for sound reasoning in a world where the new information one acquires may invalidate or augment one's current beliefs. Belief revision is the process wherein an agent modifies its beliefs to incorporate the new information received, and knowledge base merging the process wherein the agent is given two or more knowledge bases to merge.

We present a binary decision diagram (BDD) - based implementation of Delgrande and Schaub's consistency-based belief change framework. Our system focuses on knowledge base merging with the possible incorporation of integrity constraints, using a BDD solver for consistency checking. We show that the result of merging finite knowledge bases can be represented as a finite formula, and that merging can be streamlined algorithmically by restricting attention to a subset of the vocabulary of the propositional formulas involved. Experimental results and comparisons with related systems are also given.

Keywords: knowledge base merging, consistency-based belief change system, binary decision diagram solver, symmetric merge, projected merge.
To my parents and my brother.
Acknowledgments

My sincere gratitude goes to my thesis examining committee: my upstanding senior supervisor, Dr. Jim Delgrande, who has tirelessly guided me through the fascinating area of knowledge representation and reasoning for the past two years since he first hired me as an undergraduate research assistant; my supervisor, Dr. Torsten Schaub, who expeditiously answered my questions regarding answer set programming and logic programming; my examiner, Dr. Oliver Schulte, who kindly offered valuable suggestions as to how to improve my thesis.

I also express my thanks to my defense chair, Dr. Arthur (Ted) Kirkpatrick, for his tremendous efforts to help his students learn and succeed academically. I am very fortunate and appreciative to have been an undergraduate student of his. I have always enjoyed conversing with him, and I will always hold his kind words of advice and encouragement dear to my heart.

I am indebted to NSERC Canada and Simon Fraser University for their gracious offers of scholarships and fellowships. In particular, the NSERC Canada Graduate Scholarship has afforded me the comfort of focusing exclusively on my research without having had to take on teaching assistantships in the past 15 months. Of course, it is only fitting that I take this opportunity to acknowledge my nominators Drs. Delgrande, Schaub, and Kirkpatrick. Dr. Schulte and Dr. Eugenia Ternovska have also been extremely generous with their time in writing recommendations for me.

The research culminating in this thesis would not have been possible had it not been for the unwavering guidance of my supervisors and the timely replies from people I contacted via email. I am grateful to Mr. Stefan Woltran and Dr. Paolo Liberatore not only for their prompt and informative responses, but also for making the documentation and code for QUIP and BReLS available to me for testing and comparison purposes.
My relatives and friends have been supportive of my academic endeavours. Mrs. and Mr. Tsai (Tracy and John) along with their endearing children, whom I have adoringly considered my “surrogate family,” voluntarily took care of me in the most critical period of my thesis work. As well, I thank my fellow members of the Computational Logic Laboratory for having helped create a work environment conducive to research and study.

Lastly, I extend my heartfelt gratitude to my loving parents and to my amazing brother, Derek. I could not have asked for a more ideal family, and I hope that I have been as nurturing a sister to Derek as he has been a wonderful brother to me. I thank God for all the beautiful people in my life.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>Contents</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Background</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Belief Revision and Contraction</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Knowledge Base Merging</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Binary Decision Diagrams</td>
<td>13</td>
</tr>
<tr>
<td>3 Consistency-Based Belief Change</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Preliminaries</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Consistency-Based Belief Revision and Contraction</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Integrity Constraints</td>
<td>20</td>
</tr>
<tr>
<td>3.4 Consistency-Based Knowledge Base Merging</td>
<td>21</td>
</tr>
<tr>
<td>3.4.1 Symmetric Merge</td>
<td>22</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Skeptical Revision Examples ........................................ 19
3.2 Skeptical Contraction Examples .................................. 20

5.1 Example Input Files .................................................... 47
5.2 Skeptical Symmetric Merge Experimental Results ............ 50
5.3 Skeptical Projected Merge Experimental Results .......... 51

6.1 COBA 2.0 Belief Revision Experimental Results ............... 55
6.2 CBBC Projected Merge (with BuDDy and No DVO) Experimental Results 56
6.3 QUIP vs. CBBC vs. COBA 2.0 Belief Revision Experimental Results .... 58
6.4 CBBC Projected Merge (with BuDDy and DVO) Experimental Results .... 59
List of Figures

2.1 BDD for the Boolean Variable $x_1$ ........................................ 13
2.2 ROBDD for $(x_1 \land x_2) \lor (x_3 \land x_4)$ with Variable Order $x_1 < x_2 < x_3 < x_4$ . 15
2.3 ROBDD for $(x_1 \land x_2) \lor (x_3 \land x_4)$ with Variable Order $x_1 < x_3 < x_2 < x_4$ . 15
5.1 CBBC’s Interactive Task Menu ............................................. 46
Chapter 1

Introduction

*Nonmonotonic reasoning* is a process that enables an intelligent agent to conjecture plausibly in the face of inconclusive or changing information. Decision-making capability in the absence of complete information is a requisite for any agent wishing to perform common sense reasoning [4]. Similarly, an agent must be equally proficient with changing its beliefs rationally when it acquires new information; a challenge arises when the newly received information contradicts the agent’s existing beliefs.

We distinguish among three (by no means exhaustive) well-studied ways in which an intelligent agent may change its beliefs coherently. Belief revision is the process wherein an intelligent agent modifies its beliefs to incorporate a new piece of information received; belief contraction the complementary process wherein the agent removes a belief; and belief set merging the process wherein the agent is given two or more (possibly mutually contradictory) knowledge bases to merge. Furthermore, it is imperative that the consistency of a set of beliefs be maintained in any belief change, as *anything* can be deduced from an inconsistent set of premises. [4] illustrates the gravity of inconsistency by a real-life example in which a jet engine exploded upon receiving two mutually conflicting commands.

In this thesis, we present a binary decision diagram (BDD) - based system implementation of Delgrande and Schaub’s consistency-based belief change framework [15, 16, 17]. This flexible framework allows for the simultaneous specification of revision, multiple contractions, together with integrity constraints, with respect to a given knowledge base. The framework also accommodates knowledge base merging with respect to integrity constraints.

Delgrande and Schaub’s approach to belief change is rooted in the same intuition as a group of related consistency-based reasoning methodologies, such as Theorist [52] and
consistency-based diagnosis [55], in Artificial Intelligence. [15] cites the essential characteristic of consistency-based reasoners as including a nonmonotonic minimization (or maximization) step based on a designated set of atoms.

Delgrande and Schaub's framework [15, 16, 17] applies the maximization step to a distinguished set of atomic equivalences (each asserting the equivalence between a pair of corresponding atoms in disjoint languages). For example, informally stated in consistency-based knowledge base merging [16, 17], the framework first re-expresses the source knowledge bases in disjoint languages, then forces (via the aforementioned maximization step) the languages to agree on truth values of atoms wherever consistently possible, and finally re-expresses (according to the set of atomic equivalences found) the result in the original language of the knowledge bases.

Our system implementation is named CBBC, an acronym for consistency-based belief change. CBBC focuses on knowledge base merging with the possible incorporation of integrity constraints, using a BDD solver for consistency checking. We show that the result of merging finite knowledge bases while possibly incorporating integrity constraints can be represented as a finite formula, and that merging with the possible incorporation of integrity constraints can be streamlined algorithmically by restricting attention to a pertinent subset of the vocabulary of the propositional formulas involved.

Our distinguishing contribution is a reasonably efficient consistency-based system for knowledge base merging with respect to integrity constraints, the first such system of its kind to the best of our knowledge. Moreover, motivated by the availability of several practicable BDD libraries (such as CUDD [59], BuDDy [47], CAL [53], etc.), we adopt a BDD solver for consistency checking in our system. This allows us to avoid the conjunctive normal form translation of input formulas required by most Boolean satisfiability (SAT) solvers.

We devote Chapter 2 mainly to reviewing existing literature on belief revision, belief contraction, and knowledge base merging. Notably, distance-based approaches to belief revision, which stand in contrast to consistency-based approaches, are discussed. Chapter 2 concludes with a brief introduction to BDD, the data structure of our underlying BDD solver.

Chapter 3 formally describes Delgrande and Schaub's consistency-based belief change framework [15, 16, 17] upon which our system implementation, CBBC, is based. The discussion also bridges the framework to the theoretical underpinnings reviewed in Chapter 2, demonstrating that the framework has good formal properties. As well, we discuss the
computational complexity results of belief change defined in the framework.

In Chapter 4, we formalize the implementation considerations for finite and vocabulary-restricted representation as theorems. Specifically, we show that the result of merging finite knowledge bases with respect to integrity constraints can be represented as a finite formula, and that merging with respect to integrity constraints can be streamlined algorithmically by restricting attention to a relevant subset of the vocabulary of the propositional formulas involved.

Chapter 5 provides a detailed discussion of our system implementation, CBBC, taking into account the considerations for finite and vocabulary-restricted representation presented in Chapter 4. We address CBBC's algorithm, its data structures, and its underlying BDD solver. We also include detailed examples, a concise user manual, and experimental results.

In Chapter 6, we compare CBBC with such related or similar systems as COBA 2.0 [14], QUIP [19], and BReLS [46]. The belief revision capability of the former two is investigated, as are the distance-based belief revision and merging approaches of BReLS [46]. The comparisons are complemented by experimental results and an example.

Finally, Chapter 7 concludes the thesis with a summary and remarks pertaining to possible future directions of our work. Proofs of our theorems presented in Chapter 4 are included in Appendix A.
Chapter 2

Background

Gärdenfors [26] observes two general strategies for belief change, namely, to present constructions of the belief change process and to formulate rationality postulates for such constructions. The former (constructive approach) appeals to those who wish to design algorithms for computing belief change, while the latter (axiomatic approach) allows us to assess how successful an algorithm is by comparing its result with that of a belief change function that is appropriate on philosophical grounds.

In this chapter, we provide a review of literature on belief revision, belief contraction, and knowledge base merging; in particular, we present both the axiomatized and constructive approaches to belief revision and contraction, as well as the logical characterization of knowledge base merging. Additionally, we give a succinct introduction to binary decision diagrams (BDDs), the data structure upon which our underlying BDD solver is based.

Throughout this thesis, we consider propositional languages only and construct propositional formulas by using the logical symbols ¬, ∧, ∨, ⊃, ⊤, and ⊥ in the standard way. Knowledge bases are initially identified with deductively closed sets of formulas; this definition is relaxed later to allow knowledge bases to be arbitrary belief bases. \( \mathcal{L}_P \) denotes the propositional language over an alphabet \( \mathcal{P} \) of propositional atoms. Propositional formulas are denoted by lower case Greek letters possibly annotated with numbered subscripts (e.g. \( \alpha, \beta, \beta_1 \), etc.), and knowledge bases by the letter \( K \) possibly with numbered subscripts (e.g. \( K, K_2 \), etc.).
2.1 Belief Revision and Contraction

Belief revision is a process wherein an intelligent agent modifies its knowledge base to incorporate the new information it acquires. In contrast, belief contraction, complementary to the notion of belief revision, is a process wherein an intelligent agent retracts some or all of its existing knowledge when it receives new information that some particular knowledge is erroneous. Both belief revision and belief contraction, however, operate under the crucial assumption that the newly obtained information is entirely reliable.

Consider the two possible scenarios encountered by an agent revising a knowledge base $K$ by a propositional formula $\alpha$. If $\alpha$ is already consistent with $K$, then the agent can trivially add $\alpha$ to $K$. Otherwise, in order not to compromise the integrity of $K$, the agent must rationally decide what of the existing $K$ must be given up to accept $\alpha$ into $K$. The former corresponds to belief expansion, and the latter to belief revision; belief expansion is essentially a special case of belief revision. On the other hand, a contraction of $K$ with respect to $\alpha$ entails the removal of a (possibly empty) set of propositional formulas from $K$ such that $\alpha$ can no longer be inferred from the resulting $K$, with the proviso that $\alpha$ not be a tautology.

One would expect an agent’s rational modifications to its knowledge base to be guided by the minimal change principle, also dubbed the conservativity principle by Harman [30]. That is, when an agent changes its beliefs to accommodate a newly acquired piece of information, it should retain as much of its existing beliefs as consistently possible. The rationale for avoiding unnecessary losses of information is attributed to the criterion of informational economy [26], which expounds that information is generally not extraneous. Refer to pages 66 - 68 of [26] for a detailed discussion of minimal change.

Given a knowledge base $K$ and a propositional formula $\alpha$, we define $K + \alpha$ as the revision of $K$ by $\alpha$, $K + \alpha$ as the expansion of $K$ by $\alpha$ (i.e. the deductive closure of $K \cup \{\alpha\}$), and $K - \alpha$ as the contraction of $K$ by $\alpha$; $K_\perp$ is defined to be the set comprised of all propositional formulas.

The AGM paradigm [2] – named after its inventors C. Alchourrón, P. Gärdenfors, and D. Makinson – is a formal framework for modeling rational belief changes to knowledge bases under the minimal change principle. The AGM rationality postulates comprise a well-known axiomatic approach specifying the properties of belief revision and contraction functions.

In the AGM framework, a revision function $\vdash : \mathcal{L} \times \mathcal{L} \rightarrow 2^\mathcal{L}$ satisfies the following
CHAPTER 2. BACKGROUND

revision postulates [2]:

(R1) $K \downarrow \alpha$ is a belief set.

(R2) $\alpha \in K \downarrow \alpha$.

(R3) $K \downarrow \alpha \subseteq K + \alpha$.

(R4) If $\neg \alpha \notin K$, then $K + \alpha \subseteq K \downarrow \alpha$.

(R5) $K \downarrow \alpha = K_L$ iff $\vdash \neg \alpha$.

(R6) If $\vdash (\alpha \equiv \beta)$, then $K \downarrow \alpha \equiv K \downarrow \beta$.

(R7) $K \downarrow (\alpha \land \beta) \subseteq (K \downarrow \alpha) + \beta$.

(R8) If $\neg \beta \notin (K \downarrow \alpha)$, then $(K \downarrow \alpha) + \beta \subseteq K \downarrow (\alpha \land \beta)$.

(R1) and (R2) stipulate that the result of revising $K$ by $\alpha$ be a belief set containing $\alpha$. (R3) and (R4) together state that whenever $\alpha$ does not contradict what is already in $K$, belief revision is identified with belief expansion. (R5) captures the intuition that the purpose of a belief revision is to generate a new consistent belief set by guaranteeing that the result of revising $K$ by $\alpha$ is consistent unless $\alpha$ is a contradiction. (R6) specifies that belief revision should be analyzed at the knowledge level (i.e. the content of $K$ and $\alpha$) rather than the syntactic level (i.e. the linguistic form of $K$ and $\alpha$). Lastly, (R7) and (R8) address the relationship between expansion and revising by a conjunction. Given (R1) - (R6), postulates (R7) and (R8) considered together suggest that $K \downarrow (\alpha \lor \beta)$ is equivalent to $K \downarrow \alpha$, $K \downarrow \beta$, or $(K \downarrow \alpha) \land (K \downarrow \beta)$ [26, 25, 4].

In the AGM framework, a contraction function $\cdot : 2^\mathcal{L} \times \mathcal{L} \to 2^\mathcal{L}$ satisfies the following contraction postulates [2]:

(C1) $K \neg \alpha$ is a belief set.

(C2) $K \neg \alpha \subseteq K$.

(C3) If $\alpha \notin K$, then $K \neg \alpha = K$.

(C4) If $\not\vdash \alpha$, then $\alpha \notin K \neg \alpha$.

(C5) If $\alpha \in K$, then $K \subseteq (K \neg \alpha) + \alpha$. 
(C6) If \( \vdash (\alpha \equiv \beta) \), then \( \vdash \neg \alpha \equiv \neg \beta \).

(C7) \( (\neg \alpha \cap \neg \beta) \subseteq \neg (\alpha \wedge \beta) \).

(C8) If \( \beta \notin \neg \alpha \wedge \beta \), then \( \neg (\alpha \wedge \beta) \subseteq \neg \beta \).

(C1) and (C2) stipulate that the result of contracting \( K \) by \( \alpha \) be a belief set to which no new beliefs are added. In compliance with the informational economy criterion [26], (C3) requires that no beliefs be retracted from \( K \) when \( K \) is contracted by a belief it does not hold. (C4) states that \( \alpha \) is not a logical consequence of \( \neg \alpha \) unless \( \alpha \) is a tautology. (C5) is the recovery postulate, which allows us to undo contractions. (C6) specifies that belief contraction is independent of the syntactic form of \( K \) and \( \alpha \). Lastly, (C7) and (C8) address relationship between contracting by a conjunction and contracting by each conjunct individually. Given (C1) - (C6), postulates (C7) and (C8) considered together imply that \( \neg (\alpha \wedge \beta) \) is equivalent to \( \neg \alpha, \neg \beta \), or \( (\neg \alpha) \cap (\neg \beta) \) [26, 25, 4].

The recovery postulate (C5) is controversial. Gardenfors [25] illustrates (C5)'s questionable validity in probabilistic contexts with an example paraphrased here. Suppose \( K \) describes my current state of beliefs, including \( \alpha \) representing the event that I accidentally dropped a slice of bread at breakfast today and \( \beta \) representing the event that it landed with the buttered side down (assuming it is just as likely for an accidentally dropped slice of bread to land with the buttered side up as with the buttered side down). As \( \beta \) is a causal consequence of \( \alpha \), I no longer accept \( \beta \) after \( K \neg \alpha \). However, if I later reinstate \( \alpha \) via an expansion (of \( K \neg \alpha \)) by \( \alpha \), \( \beta \) does not follow as \( \neg \beta \) is equally likely. An enlightening example by Hansson showing that (C5) is contrary to intuition can be found in [28].

While the set of revision postulates (R1)-(R8) and the set of contraction postulates (C1)-(C8) do not refer to belief contraction and belief revision, respectively, revision and contraction functions are interdefinable as evidenced by the following two identities [26]:

(Levi Identity [44]) \[ K + \alpha = (K \neg \neg \alpha) + \alpha \]

(Harper Identity [31]) \[ K \neg \alpha = K \cap (K \neg \neg \alpha) \]

According to the Levi Identity, a revision by \( \alpha \) can be regarded as a composition of a contraction by \( \neg \alpha \) followed by an expansion by \( \alpha \). Conversely, the Harper Identity states that a contraction by \( \alpha \) amounts to a selection of only those formulas of \( K \) that remain when \( K \) is revised by \( \neg \alpha \).
Although the AGM rationality postulates describe classes of functions, they offer no mechanisms for constructing a particular function. Since, for any theory, there might be several functions satisfying these postulates, it would prove instructive to consider selection criteria for uniquely determining a function. To this end, Gärdenfors and Makinson [27] present a set of epistemic entrenchment postulates as a basis for a constructive definition of appropriate functions and show that the problem of constructing apt revision and contraction functions can be reduced to the problem of providing a suitable epistemic entrenchment ordering. This relative ranking of formulas in a given knowledge base \( K \) helps identify what is to be retained in, retracted from, or added to \( K \), when \( K \) is revised or contracted; the formulas in \( K \) to be forgone are those with the lowest degrees of epistemic entrenchment.

Extensive work has been done on distance-based revision operators focusing on the distance between models of a knowledge base, say \( K \), and a formula, say \( \alpha \), for revision. Representative operators proposed by Dalal [13], Borgida [8], and Weber [62] all conform to Dalal's principle of irrelevance of syntax; that is, the meaning of the resulting belief set, arbitrarily called \( K_{\text{res}} \), is independent of both the syntax of \( K \) and the syntax of \( \alpha \). Another commonality among these three proposals is that the models of \( K_{\text{res}} \) coincide with the models of \( \alpha \) which are closest to the models of \( K \), using the Hamming distance between models as a metric. In [13], the revision operator \( +D \), satisfying the AGM postulates, is given by \( \text{Mod}(K + D \alpha) = \text{Min}(\text{Mod}(\alpha), \leq_K) \), where the total pre-order relation \( \leq_K \) on models is defined by \( A \leq_K B \) iff \( \text{dist}(\text{Mod}(K), A) \leq \text{dist}(\text{Mod}(K), B) \) for any two models \( A \) and \( B \); \( \text{dist}(\text{Mod}(K), A) = \text{Min}_{M \in \text{Mod}(K)} \text{dist}(A, M) \), where \( \text{dist}(A, M) \) is the total number of propositional atoms on which models \( A \) and \( M \) disagree. [8] and [62] similarly concentrate on sets of propositional atoms on which a model of \( K \) and a model of \( \alpha \) disagree.

A distinct direction in belief revision is base revision or contraction, where revision or contraction is performed on a (arbitrary, syntactic) belief base, which is a set of formulas representing beliefs held independently of any other belief or set of beliefs. For example, a belief set containing the assertion that "Ottawa is the capital of Canada" also includes the disjunctive assertion that "Ottawa is the capital of Canada or Ottawa is the capital of USA" as a result of logical closure; however, this disjunctive assertion is a logical consequence that cannot stand on its own. While this approach [50, 29] accords with the intuition that some of one's beliefs arise only as inferences from one's more basic beliefs, its arbitrary syntactic nature renders it sensitive to the syntax of the description of the world; that is, base revision or contraction on two different belief bases whose deductive closures are the same can lead
to disparate results whose deductive closures are not the same.

2.2 Knowledge Base Merging

Knowledge base merging is the process wherein an intelligent agent is given two or more (possibly mutually contradictory) knowledge bases to merge. Merging operators prove useful in a variety of practical applications ranging from reaching an agreement in a decision-making committee, resolving a conflict between several agents, to reconciling inconsistency in a distributed database system.

Knowledge base merging bears relevance to the AGM framework of revision theory. [40] elaborates on the inter-connection between operators for merging with respect to integrity constraints and revision operators by showing that the former are a generalization of the latter to multiple knowledge bases. Furthermore, given that an agent with a knowledge base has just received new information, we can intuitively discern three possible cases with different actions the agent can take in response [38]:

- The new piece of information is less reliable than the knowledge base: either the agent completely disregards this new piece of information, or the agent revises the new piece of information by the knowledge base.

- The new piece of information is more reliable than the knowledge base: this being precisely the assumption made in belief revision theory, the agent can revise its knowledge base by the new piece of information.

- The new piece of information is as reliable as the knowledge base: the agent merges the new piece of information and the knowledge base with the goal of reaching a coherent knowledge base from the two (possibly conflicting) sources.

Before discussing the well-known sets of postulates that have been suggested to guide merging, we first introduce relevant notation following [39]. A knowledge set is a finite multi-set of the form \( \Psi = \{K_1, \ldots, K_n\} \), where \( K_1, \ldots, K_n \) are finite, not necessarily different, consistent propositional knowledge bases each represented by a finite propositional formula. We denote the conjunction of the knowledge bases in \( \Psi \) by \( \bigwedge \Psi \); i.e. \( \bigwedge \Psi = K_1 \land \ldots \land K_n \). We denote a multi-set consisting of \( n \) copies of \( K \) by \( K^{+n} \), and multi-set union by \( \sqcup \); e.g. \( \{\Psi_1\} \sqcup \{\Psi_2\} = \{\Psi_1, \Psi_2\} \).
Konieczny and Pérez [39] investigate the problem of merging possibly mutually contradictory knowledge bases according to integrity constraints; specifically, they consider operators $\Delta$ mapping a knowledge set $\Psi$ and a knowledge base $\mu$ to a knowledge base $\Delta_\mu(\Psi)$ resulting from merging $\Psi$ according to $\mu$.

**Definition 1 (IC Merging Operator)** [39] $\Delta$ is an IC merging operator iff it satisfies the following postulates:

1. **(IC0)** $\Delta_\mu(\Psi) \vdash \mu$.
2. **(IC1)** If $\Delta_\mu(\Psi) \vdash \bot$, then $\mu \vdash \bot$.
3. **(IC2)** If $\Psi_1 \equiv \Psi_2$ and $\mu_1 \equiv \mu_2$, then $\Delta_{\mu_1}(\Psi_1) \equiv \Delta_{\mu_2}(\Psi_2)$.
4. **(IC3)** If $\phi \vdash \mu$ and $\phi' \vdash \mu$, then $\Delta_\mu(\phi \cup \phi') \vdash \bot$ implies $\Delta_\mu(\phi \cup \phi') \vdash \bot$.
5. **(IC4)** If $\Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2) \vdash \bot$, then $\Delta_{\mu_1}(\Psi_1 \cup \Psi_2) \vdash \Delta_{\mu_1}(\Psi_1) \wedge \Delta_{\mu_1}(\Psi_2)$.
6. **(IC5)** $\Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2) \vdash \Delta_\mu(\Psi_1 \cup \Psi_2)$.
7. **(IC6)** If $\Delta_\mu(\Psi_1) \wedge \Delta_\mu(\Psi_2) \vdash \bot$, then $\Delta_{\mu_1}(\Psi_1 \cup \Psi_2) \vdash \Delta_{\mu_1}(\Psi_1) \wedge \Delta_{\mu_1}(\Psi_2)$.
8. **(IC7)** $\Delta_{\mu_1}(\Psi) \wedge \mu_2 \vdash \Delta_{\mu_1 \wedge \mu_2}(\Psi)$.
9. **(IC8)** If $\Delta_{\mu_1}(\Psi) \wedge \mu_2 \vdash \bot$, then $\Delta_{\mu_1 \wedge \mu_2}(\Psi) \vdash \Delta_{\mu_1}(\Psi)$.

The objective is that $\Delta_\mu(\Psi)$ is the knowledge base closest to $\Psi$ satisfying $\mu$. (IC0) ensures that the result of the merging satisfies the integrity constraints. (IC1) specifies that the result of the merging is consistent provided the integrity constraints are consistent. (IC2) states that whenever mutually consistent, the result of the merging incorporating integrity constraints amounts to the conjunction of the knowledge bases and integrity constraints. (IC3) is the principle of irrelevance of syntax, while (IC4) is the fairness postulate that merging must not give preference to any of the knowledge bases being merged. (IC5) and (IC6) together stipulate that if two mergings are mutually consistent, then their merging is logically equivalent to their conjunction. (IC7) and (IC8) are a generalization of the extended AGM revision postulates (R7) and (R8). Notably, (IC1)-(IC6) with only tautologous integrity constraints correspond to basic merging (i.e. merging without integrity constraints) [38].
Konieczny and Pérez [39] also define two classes of merging operators, namely majority merging operators and arbitration operators, which seek to minimize global dissension and individual dissatisfaction, respectively. The former class strives to satisfy the largest number of source knowledge bases possible, while the latter aims to satisfy each source knowledge base to the largest extent possible. [41] shows that the two classes are not disjoint.

**Definition 2 (Majority Merging Operator and Arbitration Operator)** [40] A majority merging operator is an IC merging operator satisfying the (Maj) postulate, and an arbitration operator an IC merging operator satisfying the (Arb) postulate:

\[(\text{Maj}) \exists \mu \exists \psi (\psi \cup \psi \vdash \mu (\psi))\]

\[(\text{Arb}) \text{ If } \mu_1 \text{ and } \mu_2 \text{ are logically independent (i.e. } \mu_1 \not\vdash \mu_2 \text{ and } \mu_2 \not\vdash \mu_1 \text{), } \mu_1 (\psi_1) \equiv \mu_2 (\psi_2), \text{ and } \mu_1 \equiv \mu_2 (\psi_1 \cup \psi_2) \equiv (\mu_1 \equiv \mu_2).\]

Liberatore and Schaerf [45] propose a commutative revision operator for merging (only) two finite and consistent propositional knowledge bases considered to be of equal importance in arbitration. Reflective of the nature of arbitration, given any two (possibly mutually inconsistent) knowledge bases \(K_1\) and \(K_2\) to merge, the proposed operator satisfies the commutativity postulate: \(K_1 \Delta K_2 \equiv K_2 \Delta K_1\). [45] assumes that knowledge bases are represented by their corresponding sets of models and propounds the following postulates, where operands of an arbitration are each a set of models:

\((\text{LS1}) A \Delta B \text{ is a set of models.}\)

\((\text{LS2}) A \Delta B = B \Delta A.\)

\((\text{LS3}) A \cap B \subseteq A \Delta B.\)

\((\text{LS4}) \text{ If } A \cap B \neq \emptyset, \text{ then } A \Delta B \subseteq A \cap B.\)

\((\text{LS5}) \text{ If } A \Delta B = \emptyset, \text{ then } A = B = \emptyset.\)

\((\text{LS6}) A \Delta (B \cup C) = \begin{cases} A \Delta B & \text{ or } \\ A \Delta C & \text{ or } \\ (A \Delta B) \cup (A \Delta C). \end{cases}\)

\((\text{LS7}) A \Delta B \subseteq A \cup B.\)
(LS8) If \( A \neq \emptyset \), then \( A \cap (A \Delta B) \neq \emptyset \).

(LS1)-(LS5) minus the commutativity postulate (LS2) are strongly similar to the AGM revision postulates (R1)-(R6), while (LS6) is the natural extension of (R7) and (R8) to arbitration. (LS7), which guarantees that the arbitration of any two knowledge bases \( K_1 \) and \( K_2 \) logically implies \( K_1 \lor K_2 \), is a parallel to the AGM postulate (R2) (i.e. \( \alpha \in K \vdash \alpha \)). [38], however, challenges the validity of (LS7) with an example; the example demonstrates that one would be advised to entertain some models even if they did not belong to either of the two input sets of models in an arbitration.

Early work on knowledge base merging includes combination operators [5], which are based on the selection of maximal (with respect to set containment) consistent subsets in the union of all the source knowledge bases. Citing these operators' inability to account for the source of information in the fusion process as a deficiency, [36] offers an example motivating the need for eliminating this deficiency. Given the knowledge bases \( K_1 = K_2 = \{\alpha, \beta\} \), \( K_3 = \{\alpha, \beta \supset \phi\} \), and \( K_4 = \{\neg \alpha, \psi\} \) with their union being \( \{\alpha, \neg \alpha, \beta, \beta \supset \phi, \psi\} \), a combination operator will generate the maxiconsistent sets \( \{\alpha, \beta \supset \phi, \psi\} \) and \( \{\neg \alpha, \beta, \beta \supset \phi, \psi\} \); we cannot decide whether \( \alpha \) or \( \neg \alpha \) holds, but it might be plausible to place \( \alpha \) in the resulting knowledge base on the grounds that \( \alpha \) is supported by three of the four sources whereas only one source supports \( \neg \alpha \). To this end, [36] devises a definition of selection functions in the manner of AGM for combination operators that take into account the source of each piece of information.

A development in knowledge base merging is to envisage an iterative approach to merging various pieces of information each received from a different source [7, 41, 37]. Booth's two-stage approach [7] is founded upon the Levi Identity [26], which decomposes a revision into a sequence of a contraction followed by an expansion. Specifically, the first stage, coined social contraction [7], involves weakening the individual pieces of information to the extent that they become jointly consistent, and the second stage trivially adds these (weakened) pieces of information to the knowledge base. In a similar vein, Konieczny [37] likens merging to a game between sources: until a coherent set of sources is reached, each round finds and weakens the weakest sources that should concede before proceeding to the next round. The weakest source is the furthest one from the other sources, using the intuition that each source will attempt to form some "coalition with near-minded sources" [37]. [41] further examines the links between merging operators and social choice rules.
2.3 Binary Decision Diagrams

A Boolean function of \( n \) arguments is a function mapping \( \{0, 1\}^n \) to \{0, 1\}. The time taken to determine its satisfiability depends on its representation – whether by truth tables, propositional formulas, or binary decision diagrams. As a result of truth tables' space-inefficiency (e.g. \( 2^{50} \) rows required for a Boolean function of 50 variables), checking the satisfiability of the Boolean function represented by an already constructed truth table takes at most time exponential in the number of function arguments. Determining whether an arbitrary propositional formula is satisfiable, however, is a well-known NP-complete problem [12]. As will become apparent shortly, deciding the satisfiability of the Boolean function represented by a reduced ordered binary decision diagram (ROBDD) amounts to testing in constant time whether the ROBDD is constantly false.

A binary decision diagram (BDD), due to Lee and Akers [43, 1], is a rooted directed acyclic graph with all its non-terminal nodes (i.e. nodes of out-degree two) labeled with Boolean variables, and all its terminal nodes (i.e. nodes of out-degree zero) labeled with either 0 or 1 denoting false or true, respectively. For any non-terminal node \( u \), its two outgoing edges are given by functions \( \text{low}(u) \) and \( \text{high}(u) \) corresponding to \( u \)'s successor nodes if \( \text{var}(u) \) is assigned 0 and 1, respectively, where \( \text{var}(u) \) denotes the variable associated with \( u \). Figure 2.1 is the BDD representation of the Boolean variable \( x_1 \).

![Figure 2.1: BDD for the Boolean Variable \( x_1 \)](image)

A BDD is ordered (OBDD), a notion due to Bryant [9], iff on all paths from the root through the graph to a terminal node, the occurrence of variables respects a given linear order \( v_1 < v_2 < ... < v_n \) although not all variables may occur on a path. In general, the chosen variable ordering has a significant impact on the size of the OBDD representing a given function. [32] provides an illustrated example paraphrased here. Given the propositional formula \((v_1 \lor v_2) \land (v_3 \lor v_4) \land ... \land (v_{2n-1} \lor v_{2n})\), the ordering \( v_1 < v_2 < ... < v_{2n} \) results in an OBDD with \( 2n + 2 \) nodes (assuming no duplicate terminals), whereas the ordering \( v_1 < v_3 < ... < v_{2n-1} < v_2 < v_4 < ... < v_{2n} \) leads to an OBDD with \( 2^{n+1} \) nodes (assuming
no duplicate terminals). Finding the optimal ordering is computationally expensive; how-
ever, good heuristics for choosing orderings avoiding exponential growth have been devised
by [49, 22, 24] and via dynamic variable reordering in [56, 21].

An (O)BDD is reduced (R(O)BDD) iff it has the following three properties [3, 32]:

- (no duplicate non-terminals): No two non-terminal nodes \( u \) and \( v \) have the same
  variable and successors; i.e. \( \text{var}(u) = \text{var}(v), \text{low}(u) = \text{low}(v), \) and
  \( \text{high}(u) = \text{high}(v) \), then \( u = v \).

- (no duplicate terminals): Any and all edges going into a terminal node (0 or 1) are
  directed to just one (the same) copy of that terminal node (0 or 1, respectively).

- (non-redundant tests): No non-terminal node \( u \) has identical successors; i.e. \( \text{low}(u) \neq \text{high}(u) \).

As Andersen [3] shows, for any function \( f : \{0, 1\}^n \to \{0, 1\} \), there is exactly one ROBDD
representing \( f \) for a given variable ordering. [32] elaborates on the importance of having a
canonical form (ROBDDs) for OBDDs by offering the following reasons:

- (absence of redundant variables): Any arguments for a Boolean function \( f \) that have
  no effect on \( f \)'s value are removed from any ROBDD representing \( f \).

- (semantic equivalence test): Given any two Boolean functions \( f(v_1, v_2, \ldots, v_n) \) and
  \( g(v_1, v_2, \ldots, v_n) \) represented by OBDDs \( B_f \) and \( B_g \), respectively, both with compatible
  variable orderings (i.e. there are no variables \( v_i \) and \( v_j \) such that \( v_i \) comes before \( v_j \)
  in \( B_f \)'s ordering and \( v_j \) comes before \( v_i \) in \( B_g \)'s ordering), \( f \) and \( g \) are semantically
  equivalent iff the reduced versions of \( B_f \) and \( B_g \) have identical structure.

- (validity test): A Boolean function is valid iff its ROBDD is the terminal node 1.

- (implication test): Given any two Boolean functions \( f(v_1, v_2, \ldots, v_n) \) and \( g(v_1, v_2, \ldots, v_n) \),
  \( f \) implies \( g \) (i.e. \( g \) also computes 1 whenever \( f \) does so) iff the ROBDD for \( f \cdot \overline{g} \) is the
  terminal node 0.

- (satisfiability test): A Boolean function is satisfiable iff its ROBDD is not the terminal
  node 0.
Figures 2.2 and 2.3 illustrate the impact of variable ordering on the size of a BDD. Figure 2.2 is the unique ROBDD representation of \((x_1 \land x_2) \lor (x_3 \land x_4)\) with respect to the variable order \(x_1 < x_2 < x_3 < x_4\). Similarly, Figure 2.3 is the unique ROBDD representation of the same Boolean function but with respect to the variable order \(x_1 < x_3 < x_2 < x_4\). With relatively fewer nodes and fewer edges, Figure 2.2 is clearly more space-efficient than Figure 2.3.
ROBDDs have a wide scope of potential applications, ranging from classical chess problems such as the 8 queens and the knight's tour, to testing the correctness and equivalence of combinational circuits [3]. One major use of ROBDDs is in formal verification and symbolic model checking [10], wherein given a model of a hardware or software finite-state system $M$ along with some properties $P$ that $M$ should satisfy, we determine whether $M$ satisfies $P$. Many BDD software packages are publicly available, such as CUDD [59], BuDDy [47], CAL [53], and JDD [61]; there are also publicly available Java interfaces JavaBDD [63] and JBDD [60] to some of the aforementioned BDD packages. See [58, 34, 33] for a comparison of some of the aforementioned and other BDD packages.
Chapter 3

Consistency-Based Belief Change

We describe the consistency-based belief change framework which was proposed by Delgrande and Schaub in [15, 16, 17], and upon which our implementation CBBC is based. After first presenting our formal preliminaries, we discuss how belief revision, belief contraction, integrity constraints and knowledge base merging are treated in Delgrande and Schaub’s framework [15, 16, 17]. As well, we present the computational complexity results of belief change defined in the framework.

3.1 Preliminaries

For $\alpha \in \mathcal{L}_P$, $\alpha'$ is the result of replacing in $\alpha$ each proposition $p \in \mathcal{P}$ by the corresponding proposition $p' \in \mathcal{P}'$; This definition applies analogously to sets of formulas.

A belief change scenario [15] in $\mathcal{L}_P$ is a triple $B = (K, R, C)$ where $K$, $R$, and $C$ are sets of formulas in $\mathcal{L}_P$. Stated informally, $K$ is a knowledge base that is to be modified so that it will contain all the formulas in $R$ and no formulas in $C$. An extension determined by a belief change scenario is defined as follows.

Definition 3 (Belief Change Extension) [15] Let $B = (K, R, C)$ be a belief change scenario in $\mathcal{L}_P$, and a maximal set of equivalences $EQ \subseteq \{ p \equiv p' \mid p \in \mathcal{P} \}$ be such that $Cn(K' \cup R \cup EQ) \cap (C \cup \{ \bot \}) = \emptyset$. Then $Cn(K' \cup R \cup EQ) \cap \mathcal{L}_P$ is a belief change extension of $B$, and $EQ$ determines a belief change extension of $B$. If no such $EQ$ exists, then $B$ is inconsistent and $\mathcal{L}_P$ is defined to be the sole (inconsistent) belief change extension of $B$.

Definition 3 provides a very general framework for specifying belief change. Note that
the exclusive use of "\{⊥\}" in Definition 3 is to take care of consistency when $C = \emptyset$, and "maximal" is with respect to set containment. For a given belief change scenario, there may be more than one consistent belief change extension; thus, we use a selection function $c$ that, given any non-empty set $I$ of indices of determining $EQ$ sets as argument, returns an index indicating the chosen $EQ$ set.

Definition 3 is motivated by the observation [15]: when revising a knowledge base $K$ by a formula $\alpha$, we may consider interpretations of $K$ and $\alpha$ as the syntactic form of either $K$ or $\alpha$ does not clearly impart which formulas should or should not be retained in the resulting knowledge base. Intuitively, a model of $K \vdash \alpha$ can be obtained by choosing a model of $\alpha$ and adding to it all the parts of a model of $K$ that do not conflict with the chosen model of $\alpha$; That is, we have $\text{Mod}(K \vdash \alpha) \subseteq \text{Mod}(\alpha)$.

To this end, with respect to $K \vdash \alpha$, the approach in [15] proceeds sequentially. First, we express $K$ as $K'$ in order that $K' \cup \{\alpha\}$ is satisfiable (assuming that $K$ and $\alpha$ are each satisfiable). Next, we build a maximal set $EQ$ of atomic equivalences between the disjoint but isomorphic languages $P'$ (of $K'$) and $P$ (of $K$ and $\alpha$) such that $K' \cup \{\alpha\} \cup EQ$ is consistent. As a result, any model of $K' \cup \{\alpha\} \cup EQ$ is a model of $\alpha$ in the original language $P$ with $EQ$ linking the truth values of atoms in $K'$ and $\alpha$.

A candidate choice revision of $K$ by $\alpha$ consists of the conjunction of $\alpha$ with $K'$ re-expressed in the original language $P$ according to $EQ$: every primed atom $p'$ in $K'$ is replaced with $p$ if $(p \equiv p') \in EQ$, but with $\neg p$ otherwise. General or skeptical revision is the intersection of all candidate choice revisions, one per possible $EQ$. A choice change represents one feasible way in which one or more knowledge bases can be changed to accommodate new information, while a skeptical change conservatively takes all choice changes into account.

### 3.2 Consistency-Based Belief Revision and Contraction

Definition 3 can be restricted to provide the following specific functions for belief revision and belief contraction in Delgrande and Schaub’s framework [15].

**Definition 4 (Revision) [15]** Let $K$ be a knowledge base, $\alpha$ a formula, and $(E_i)_{i \in I}$ the family of all belief change extensions of $(K, \{\alpha\}, \emptyset)$. Then, we define

1. $K \vdash_\alpha E_i$ as a choice revision of $K$ by $\alpha$ with respect to some selection function $c$ with $c(I) = i$. 

CHAPTER 3. CONSISTENCY-BASED BELIEF CHANGE

Table 3.1: Skeptical Revision Examples

<table>
<thead>
<tr>
<th>(K')</th>
<th>(\alpha)</th>
<th>(EQ)</th>
<th>(K+\alpha)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(~s' \land t')</td>
<td>(s \lor \neg t)</td>
<td>({s \equiv s', t \equiv t'})</td>
<td>(~s \equiv \neg t)</td>
</tr>
<tr>
<td>(~s' \lor t')</td>
<td>(s \lor \neg t)</td>
<td>({s \equiv s', t \equiv t'})</td>
<td>(s \equiv t)</td>
</tr>
<tr>
<td>(~s' \land t')</td>
<td>(s \land \neg t)</td>
<td>({})</td>
<td>(s \land \neg t)</td>
</tr>
</tbody>
</table>

2. \(K+\alpha = \bigcap_{i \in I} E_i\) as the (skeptical) revision of \(K\) by \(\alpha\).

**Definition 5 (Contraction)** [15] Let \(K\) be a knowledge base, \(\alpha\) a formula, and \((E_i)_{i \in I}\) the family of all belief change extensions of \((K, \emptyset, \{\alpha\})\). Then, we define

1. \(K_\rightarrow \alpha = E_i\) as a choice contraction of \(K\) by \(\alpha\) with respect to some selection function \(c\) with \(c(I) = i\).

2. \(K_\rightarrow \alpha = \bigcap_{i \in I} E_i\) as the (skeptical) contraction of \(K\) by \(\alpha\).

Notably, multiple contraction [23], wherein contraction is carried out with respect to a set of (not necessarily mutually consistent) formulas, can be expressed in the framework with \(|C| > 1\). Consequently, we can use the belief change scenario \((K, \emptyset, \{\phi, \neg \phi\})\) to represent a symmetric contraction [35] of \(\phi\) from \(K\), whereby the resulting knowledge base is non-committal about the truth value of \(\phi\).

Table 3.1 gives examples of skeptical revision. The knowledge base is in the first column, but with atoms already renamed. The second column gives the revision formula, while the next lists the \(EQ\) set(s); the last column gives a finite representation of \(Cn(K+\alpha)\). In the example \(\{s \land t\} \rightarrow (s \lor \neg t)\), the two \(EQ\) sets \(\{s \equiv s'\}\) and \(\{t \equiv t'\}\) yield two choice belief change extensions \(Cn(\{\neg s' \land t'\} \cup \{s \lor \neg t\} \cup \{s \equiv s'\}) \cap L_P = Cn(\neg s \land \neg t)\) and \(Cn(\{\neg s' \land t'\} \cup \{s \lor \neg t\} \cup \{t \equiv t'\}) \cap L_P = Cn(s \land t)\), respectively.

Table 3.2 gives examples of skeptical contraction, using Table 3.1's format except with the second column giving the contraction formula and the last column giving a finite representation of \(Cn(K_\rightarrow \alpha)\). In the example \(\{s \land t\} \rightarrow s\), the sole \(EQ\) set \(\{t \equiv t'\}\) generates the belief change extension \(Cn(\{s' \land t'\} \cup \emptyset \cup \{t \equiv t'\}) \cap L_P = Cn(s)\).

The belief functions \(+, \_c, \rightarrow, \_c\) have good formal properties, satisfying most of the AGM postulates (R1)-(R8) and (C1)-(C8) enumerated in Section 2.1. However, postulate (R8) and the recovery postulate (C5) are not met. Refer to [15] for counter-examples to (R8) and (C5), as well as the proof of the following theorem.
CHAPTER 3. CONSISTENCY-BASED BELIEF CHANGE

Table 3.2: Skeptical Contraction Examples

<table>
<thead>
<tr>
<th>$K'$</th>
<th>$\alpha$</th>
<th>$EQ$</th>
<th>$K_{\neg\alpha}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s' \wedge t'$</td>
<td>$s$</td>
<td>${t \equiv t'}$</td>
<td>$t$</td>
</tr>
<tr>
<td>$\neg s \wedge \neg t$</td>
<td>$\neg s \wedge \neg t$</td>
<td>${s \equiv s', t \equiv t'}$</td>
<td>$\neg s \wedge \neg t$</td>
</tr>
<tr>
<td>$s' \vee t'$</td>
<td>$s \wedge t$</td>
<td>${s \equiv s', t \equiv t'}$</td>
<td>$s \vee t$</td>
</tr>
</tbody>
</table>

Theorem 1 [15] Let $+_{c}$ and $+_{c}$ be as given in Definition 4, and $\neg$ and $\neg_{c}$ be as given in Definition 5.
1. $+$ and $+_{c}$ satisfy the following postulates:
   - (R1) to (R4), (R6), and (R7)
   - a weaker version of (R5): $K + \alpha = K_{\bot}$ iff ($K = K_{\bot}$ or $\vdash \neg \alpha$)
   - a stronger version of (R6): $\vdash \alpha \equiv \beta$ and $\vdash K_1 \equiv K_2$, then $K_1 + \alpha = K_2 + \beta$.

2. $\neg$ satisfies (C7) and both $\neg$ and $\neg_{c}$ satisfy the following postulates:
   - (C1) to (C3), and (C6)
   - a weaker version of (C4): $\not\vdash \alpha$ and $\vdash K \neq K_{\bot}$, then $\alpha \notin K_{\neg \alpha}$
   - a stronger version of (C6): $\vdash \alpha \equiv \beta$ and $\vdash K_1 \equiv K_2$, then $K_1 \neg \alpha = K_2 \neg \beta$.

3. For any selection function $c$, there exists some selection function $c'$ such that
   - $K_{\neg \alpha} (\alpha \land \beta) = K_{\neg c} \neg \alpha$ or $K_{\neg \alpha} (\alpha \land \beta) = K_{\neg c} \neg \beta$
   - If $K_{\neg \alpha} (\alpha \land \beta) \not\vdash \neg \alpha$, then $K_{\neg \alpha} (\alpha \land \beta) = K_{\neg c} \neg \alpha$.

In addition, [15] shows the near-interdefinability results in the framework: the Levi Identity (i.e. $K + \alpha = (K_{\neg \alpha}) + \alpha$) is preserved, whereas the Harper Identity is only partially maintained with $K_{\neg \alpha} \subseteq K \cap (K + \neg \alpha)$.

3.3 Integrity Constraints

The generality of Delgrande and Schaub's framework [15], as formalized in Definition 3, lends itself to simultaneous revision and contraction by sets of formulas. As such, it also provides a natural treatment of integrity constraints. There are two standard definitions of
a knowledge base $K$ satisfying a static integrity constraint $IC$, namely the consistency-based approach of [42, 57] and the entailment-based approach of [54]. In the former, $K$ satisfies $IC$ iff $K \cup \{IC\}$ is satisfiable; in the latter, $K$ satisfies $IC$ iff $K \vdash IC$.

To specify a belief change incorporating integrity constraints, we can formalize it as the belief change scenario $(K, R \cup IC_e, C \cup IC_c)$, where $K$ is a knowledge base, $R$ a set of formulas for revision, $C$ a set of formulas for contraction, $IC_e$ a set of formulas as entailment-based integrity constraints, and $IC_c$ a set of formulas as consistency-based integrity constraints such that $IC_c = \{\neg \phi | \phi \in IC_e\}$.

Delgrande and Schaub [15] prove the following theorem attesting to the preservation of both entailment-based and consistency-based integrity constraints in their framework.

**Theorem 2 [15]** Let $K^{+IC_e,IC_c} \alpha$ denote the (skeptical) revision of a knowledge base $K$ by a formula $\alpha$ incorporating entailment-based integrity constraints $IC_e$ and consistency-based integrity constraints $IC_c$. Then, we have

1. $(K^{+IC_e,IC_c} \alpha) \vdash IC_e$ and

2. If $K \not\vdash \bot$, then
   for every $\phi \in IC_c$, we have $(K^{+IC_e,IC_c} \alpha) \not\vdash \neg \phi$ if $IC_e \cup \{\alpha\} \not\vdash \neg \phi$.

### 3.4 Consistency-Based Knowledge Base Merging

The framework given by Definition 3 can be modified to handle knowledge base merging, in which multiple, not necessarily mutually consistent, knowledge bases are combined into a single belief set. [16, 17] propose two different approaches to knowledge base merging, both expressible in the general framework of [15].

In the first approach, the common information in the source knowledge bases is collected to produce a resulting knowledge base retaining as much as consistently possible of the contents of the source knowledge bases. In the second approach, source knowledge bases are projected onto a target knowledge base which we desire to augment; in the simplest case, the target knowledge base could consist solely of $\top$.

**Definition 6 (Multi Belief Change Scenario) [16, 17]** A multi belief change scenario in $\mathcal{L}_P$ is a triple $((K_j)_{j \in J}, R, C)$ where $(K_j)_{j \in J}$ is a family of sets of formulas in $\mathcal{L}_P$, and $R$ and $C$ are sets of formulas in $\mathcal{L}_P$. 


CHAPTER 3. CONSISTENCY-BASED BELIEF CHANGE

Stated informally, \((K_j)_{j \in J}\) is a collection of knowledge bases that are to be merged in a resulting knowledge base that will contain all the formulas in \(R\) and no formulas in \(C\). Definition 6 resembles the belief change scenario definition given in Section 3.1 except that the single set \(K\) of formulas is extended here to several sets of formulas.

In accordance with Section 3.3, \(R\) and \(C\) can be used to express entailment-based integrity constraints [54] and the negations of consistency-based integrity constraints [42, 57], respectively. Furthermore, \(R\) and \(C\) can each just as easily be regarded as a set of formulas for revision and for contraction, respectively. Consequently, the flexible framework [15, 16, 17] allows for the simultaneous specification of merging, revision, and multiple contractions with respect to integrity constraints.

To set the stage for our discussion of the two approaches, we first extend the notation \(\alpha'\) from Section 3.1 to handle numbered superscripts \(i > 0\). For any alphabet \(\mathcal{P}\), \(\mathcal{P}^i\) is defined as \(\mathcal{P}^i = \{p^i \mid p \in \mathcal{P}\}\). Similarly, for any \(\alpha \in \mathcal{L}_\mathcal{P}\), \(\alpha^i\) is the result of replacing in \(\alpha\) each proposition \(p \in \mathcal{P}\) by the corresponding proposition \(p^i \in \mathcal{P}^i\). This definition applies to sets of formulas as well; e.g. \(K^i\) is the result of replacing in each formula of \(K\) each proposition \(p \in \mathcal{P}\) by the corresponding proposition \(p^i \in \mathcal{P}^i\). We also generalize the notation analogously for collections of positive integers; for example, for any set or list \(J\) of positive integers, \(\alpha^J = \{\alpha^i \mid i \in J\}\) and \(R^J = \bigcup_{j \in J} R^j\).

3.4.1 Symmetric Merge

In symmetric merge, the common information in the source knowledge bases is collected to produce a resulting knowledge base retaining as much as consistently possible of the contents of the source knowledge bases.

Definition 7 (Symmetric Belief Change Extension) [16, 17] Let \(B = ((K_j)_{j \in J}, R, C)\) be a multi belief change scenario in \(\mathcal{L}_\mathcal{P}\), and \(EQ \subseteq \{p^k \equiv p^m \mid p \in \mathcal{P}, k \neq m, \text{ and } k, m \in J\}\) be a maximal set of equivalences such that \(\text{Cn}(\bigcup_{j \in J} K^j \cup R^J \cup EQ) \cap (C^J \cup \bot) = \emptyset\). Then \(\{\alpha \mid \{\alpha^i \mid j \in J\} \subseteq \text{Cn}(\bigcup_{j \in J} K^j \cup R^J \cup EQ)\}\) is a consistent symmetric belief change extension of \(B\), and \(EQ\) determines a consistent symmetric belief change extension of \(B\). If no such \(EQ\) exists, then \(B\) is inconsistent and \(\mathcal{L}_\mathcal{P}\) is defined to be the sole (inconsistent) symmetric belief change extension of \(B\).

The use of \(R^J\) and \(C^J\) in Definition 7 ensures that the integrity constraints in \(R\) and \(C\) are respected by each source knowledge base \(K_j\) with \(j \in J\).
CHAPTER 3. CONSISTENCY-BASED BELIEF CHANGE

Definition 8 (Symmetric Merge) [16, 17] Let $B = ((K_j)_{j \in J}, R, C)$ be a multi belief change scenario in $L_P$, and $(E_i)_{i \in I}$ be the family of all symmetric belief change extensions of $B$. Then, we define

1. $\triangle_c ((K_j)_{j \in J}, R, C) = E_i$ as a symmetric choice merge of $(K_j)_{j \in J}$ with respect to $R$, $C$, and some selection function $c$ with $c(I) = i$.

2. $\triangle ((K_j)_{j \in J}, R, C) = \bigcap_{i \in I} E_i$ as the symmetric (skeptical) merge of $(K_j)_{j \in J}$ with respect to $R$ and $C$.

Example 1 Let $K_1 = \{s \land t\}$ and $K_2 = \{\neg s \land \neg t\}$. $\triangle ([K_1, K_2], \emptyset, \emptyset)$ yields (informally) $(s^1 \land t^1) \land (\neg s^2 \land \neg t^2)$ along with $EQ = \emptyset$, thus resulting in $Ch([s \land t] \lor (\neg s \land \neg t))$.

Example 2 Let $K_1 = \{q\}$ and $K_2 = \{\neg r\}$. $\triangle ([K_1, K_2], \{r\}, \emptyset)$ yields (informally) $Ch(q^1 \land \neg r^2 \land r^1 \land r^2) \vdash \bot$ along with no $EQ$, thus resulting in $L_P$, which is the sole (inconsistent) symmetric belief change extension.

Example 3 Let $\Psi_1 = \{K_1, K_2\}$ and $\Psi_2 = \{K_3\}$, where $K_1 = \{s \land t\}$, $K_2 = \{\neg s\}$, and $K_3 = \{s\}$. Then $\triangle (\Psi_1, \emptyset, \emptyset)$ yields (informally) $(s^1 \land t^1) \land s^2$ along with $EQ = \{t^1 \equiv t^2\}$, thus resulting in $Ch(\{t\})$. Furthermore, $\triangle (\Psi_2, \emptyset, \emptyset) = Ch(\{s\})$. $\triangle (\Psi_1 \cup \Psi_2, \emptyset, \emptyset) = \triangle ([K_1, K_2, K_3], \emptyset, \emptyset)$ yields (informally) $(s^1 \land t^1) \land (s^2 \land s^3)$ along with $EQ = \{t^1 \equiv t^2, t^1 \equiv t^3, t^2 \equiv t^3, s^1 \equiv s^3\}$, thus resulting in $Ch(\{t\})$.

The following example shows that symmetric merge is not associative.

Example 4 Let $K_1 = \{s \land t\}$, $K_2 = \{s\}$, and $K_3 = \{\neg s\}$. $\triangle ([K_1, K_2], \emptyset, \emptyset)$ call it $K_{1,2}$. $\triangle ([K_2, K_3], \emptyset, \emptyset) = Ch([s \lor \neg s])$, call it $K_{2,3}$. However, $[\triangle ([K_{1,2}, K_3], \emptyset, \emptyset) = Ch(\{s \land t\})] \neq [\triangle ([K_1, K_{2,3}], \emptyset, \emptyset) = Ch(\{s \land t\})]$.

Delgrande and Schaub [16, 17] show that the belief functions $\triangle$ and $\triangle_c$ have good formal properties, satisfying most of the IC merging postulates (IC0)-(IC8) enumerated in Section 2.2.

Theorem 3 [16, 17] Let $\triangle$ and $\triangle_c$ be as given in Definition 8. Then $\triangle$ and $\triangle_c$ satisfy the postulates (IC0), (IC2)-(IC5), (IC7), (IC8), and a weaker version of (IC1): If $\mu \not\vdash \bot$ and $K \not\vdash \neg \mu$ for each $K \in \Psi$, then $\triangle (\Psi, \mu, \emptyset) \not\vdash \bot$. 


Counter-example to (IC1) is given by Example 2 by setting \( \mu = \{ r \} \) and \( \Psi = \{ K_1, K_2 \} \); counter-example to (IC6) by Example 3 by setting \( \mu = \emptyset \), wherein \( \triangle (\Psi_1, \emptyset, \emptyset) \land \triangle (\Psi_2, \emptyset, \emptyset) \not\models \perp \) but \( \triangle (\Psi_1 \cup \Psi_2, \emptyset, \emptyset) \not\models \triangle (\Psi_1, \emptyset, \emptyset) \land \triangle (\Psi_2, \emptyset, \emptyset) \).

Delgrande and Schaub [16, 17] also demonstrate that the symmetric merge of two belief sets can be expressed in terms of belief revision \( + \) as given by Definition 4, and vice versa.

**Theorem 4** [16, 17] Let \( \triangle \) and \( + \) be as given in Definitions 8 and 4, respectively. Then, we have

1. \( \triangle(\{K_1,K_2\},\emptyset,\emptyset) = (K_1 + K_2) \cap (K_2 + K_1) \) and
2. \( (K_1 + K_2) = \triangle(\{K_1, T\}, K_2, \emptyset) \).

### 3.4.2 Projected Merge

In *projected merge*, source knowledge bases are projected onto a target knowledge base which we desire to augment; in the simplest case, the target knowledge base could consist solely of \( \top \).

**Definition 9 (Projected Belief Change Extension)** [16, 17] Let \( B = ((K_j)_{j \in J}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_P \), and \( EQ \subseteq \{ p^j \equiv p \mid p \in P, j \in J \} \) be a maximal set of equivalences such that \( Cn(\bigcup_{j \in J} K_j^j \cup R \cup EQ) \cap (C \cup \{ \perp \}) = \emptyset \). Then \( Cn(\bigcup_{j \in J} K_j^j \cup R \cup EQ) \cap \mathcal{L}_P \) is a consistent projected belief change extension of \( B \), and \( EQ \) determines a consistent projected belief change extension of \( B \). If no such \( EQ \) exists, then \( B \) is inconsistent and \( \mathcal{L}_P \) is defined to be the sole (inconsistent) projected belief change extension of \( B \).

Note the similarity between belief revision and projected merge. Belief revision projects the original knowledge base onto the formula with which it is revised such that the revision formula is believed in the resulting knowledge base. Likewise, projected merge projects the source knowledge bases onto a resulting knowledge base in which \( R \) is believed.

**Definition 10 (Projected Merge)** [16, 17] Let \( B = ((K_j)_{j \in J}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_P \), and \( (E_i)_{i \in I} \) be the family of all projected belief change extensions of \( B \). Then, we define

1. \( \nabla_c ((K_j)_{j \in J}, R, C) = E_i \) as a projected choice merge of \( (K_j)_{j \in J} \) with respect to \( R, C \), and some selection function \( c \) with \( c(I) = i \).
2. \( \nabla ((K_j)_{j \in J}, R, C) = \bigcap_{i \in I} E_i \) as the projected (skeptical) merge of \((K_j)_{j \in J}\) with respect to \(R\) and \(C\).

Example 5 Let \(K_1 = \{s \land t\}\) and \(K_2 = \{\neg s \land \neg t\}\). \(\nabla ((K_1, K_2), \emptyset, \emptyset)\) yields (informally) \((s^1 \land t^1) \land (\neg s^2 \land \neg t^2)\) along with \(EQ_1 = \{s^1 \equiv t, t^1 \equiv t\}, EQ_2 = \{s^2 \equiv s, t^1 \equiv t\}, EQ_3 = \{t^2 \equiv t, s^1 \equiv s\}, EQ_4 = \{t^2 \equiv t, s^2 \equiv s\}\). The skeptical projection result is, thus, \(Ch(\top)\).

Example 6 Let \(K_1 = \{s \land \neg s\}\) and \(K_2 = \{t\}\). \(\nabla ((K_1, K_2), \{r\}, \emptyset)\) yields (informally) \(Ch(s^1 \land \neg s^1 \land t^2 \land r) \vdash \bot\) along with no EQ, thus resulting in \(LP\), which is the sole (inconsistent) projected belief change extension. This serves as a counter-example to the IC merging postulate (IC1) listed in Section 2.2.

The following example shows that projected merge is not associative.

Example 7 Let \(K_1 = \{s \land t\}, K_2 = \{s\},\) and \(K_3 = \{\neg s\}\). \(\nabla ((K_1, K_2, K_3), \emptyset, \emptyset) = Ch(\{s \land t\})\), call it \(K_{1,2}\). \(\nabla ((K_2, K_3), \emptyset, \emptyset) = Ch(\{s \lor \neg s\})\), call it \(K_{2,3}\). However, \(\nabla ((K_{1,2}, K_3), \emptyset, \emptyset) = Ch(\{t\}) \neq Ch((\{K_1, K_{2,3}\}, \emptyset, \emptyset) = Ch(\{s \land t\}))\).

Delgrande and Schaub [16, 17] show that the belief functions \(\nabla\) and \(\nabla_c\) have good formal properties, satisfying most of the IC merging postulates (IC0)-(IC8) enumerated in Section 2.2.

Theorem 5 [16, 17] Let \(\nabla\) and \(\nabla_c\) be as given in Definition 10. Then \(\nabla\) and \(\nabla_c\) satisfy the following postulates:

- (IC0), (IC2), (IC3), (IC5), (IC7), and (IC8)
- a version of (IC1): If \(\Psi \vdash \neg \mu\) and \(\mu \nvdash \bot\), then \(\nabla (\Psi, \mu, \emptyset) \nvdash \bot\)
- a version of (IC4): If \(\phi_1 \nvdash \bot, \phi_2 \nvdash \bot, \phi_1 \vdash \mu,\) and \(\phi_2 \vdash \mu,\) then \(\nabla ((\phi_1 \cup \phi_2), \mu, \emptyset) \land \phi_1 \nvdash \bot\).

Delgrande and Schaub [16, 17] also demonstrate that belief revision \(\vdash\) as given by Definition 4 can be expressed in terms of the projected merge of two belief sets incorporating entailment-based integrity constraints.

Theorem 6 [16, 17] Let \(\nabla\) and \(\vdash\) be as given in Definitions 10 and 4, respectively. Then, we have \((K_1 \vdash K_2) = \nabla(\{K_1, \top\}, K_2, \emptyset)\).
Finally, the following result obtained by [16, 17] sheds light on the relationship between symmetric merge and projected merge.

**Theorem 7** [16, 17] Let \( ((K_j)_{j \in J}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_\mathcal{P} \), and let \( \Delta \) and \( \nabla \) be as given in Definitions 8 and 10, respectively. Then, we have \( \nabla((K_j)_{j \in J}, R, C) \subseteq \Delta((K_j)_{j \in J}, R, C) \).

### 3.5 Computational Complexity

In [18], Delgrande, Schaub, Tompits, and Woltran explore the computational complexity of several reasoning tasks in the belief change scenario context, basing the analysis on their proposed polynomial-time constructible reductions of these tasks to closed quantified Boolean formulas (QBFs) in prenex form. Specifically, for any belief change scenario \( B \), they provide a QBF such that the satisfying truth assignments to the free variables coincide exactly with \( B \)'s belief change extensions.

The reasoning tasks considered in [18] are the following decision problems and their corresponding search problems.

**EXT**: Given any belief change scenario \( B \), decide whether \( B \) has a consistent belief change extension.

**CHOICE**: Given any belief change scenario \( B \) and any formula \( \alpha \), decide whether \( \alpha \) is contained in some consistent belief change extension of \( B \).

**SKEPTICAL**: Given any belief change scenario \( B \) and any formula \( \alpha \), decide whether \( \alpha \) is contained in all the consistent belief change extensions of \( B \).

These reasoning tasks can be specialized to \( REXT \), \( RCHOICE \), and \( RSKEPTICAL \) for belief revision; and to \( CEXT \), \( CCHOICE \), and \( CSKEPTICAL \) for belief contraction.

The prenex QBF representation lends itself naturally to the investigation of computational complexity. Observe that in Definition 3 given in Section 3.1, a belief change scenario \( B = (K, R, C) \) has a consistent belief change extension iff \( K' \cup R \cup \{ \neg \psi \} \models \bot \) for each \( \psi \in C \cup \{ \bot \} \). Consequently, \( B \) has a consistent belief change extension iff \( \exists V \forall V'( (\forall K') \land (\forall R) \land \neg \psi ) \) evaluates to true for each \( \psi \in C \cup \{ \bot \} \) with \( V = (P(K) \cup P(R) \cup P(C)) \), as proved in [18]. This result establishes \( EXT \) as an \( NP \) problem. Furthermore, [18] shows that \( CHOICE \) and \( SKEPTICAL \) can each be transformed in polynomial time into a
CHAPTER 3. CONSISTENCY-BASED BELIEF CHANGE

closed QBF of the prenex form $\exists Y_1 \forall Y_2 \gamma$ and $\forall Z_1 \exists Z_2 \phi$, respectively, where $\gamma$ and $\phi$ are both strictly propositional. Thus, $\text{CHOICE}$ is in $\Sigma^P_2$ while $\text{SKEPTICAL}$ is in $\Pi^P_2$.

[18] formally states all the pertinent computational complexity results in the following theorem.

**Theorem 8** [18] The decision problems $\text{EXT}$, $\text{CHOICE}$, and $\text{SKEPTICAL}$, as well as their specialized versions for belief revision and contraction, have the following completeness properties:

1. $\text{EXT}$, $\text{REXT}$, and $\text{CEXT}$ are NP-complete.

2. $\text{CHOICE}$, $\text{RCHOICE}$, and $\text{CCHOICE}$ are $\Sigma^P_2$-complete.

3. $\text{SKEPTICAL}$, $\text{RSKEPTICAL}$, and $\text{CSKEPTICAL}$ are $\Pi^P_2$-complete.

[16, 17] assert that as corollaries of Theorem 8, the variants of the decision problems $\text{EXT}$, $\text{CHOICE}$, and $\text{SKEPTICAL}$ for symmetric merge and projected merge fall in the same respective complexity classes. The ease with which the computational complexity results for symmetric merge and projected merge are obtained is ascribed to the formulation of all the belief change operations (revision, contraction, and knowledge base merging) within one unified framework proposed by [15, 16, 17].
Chapter 4

Implementation Considerations for Merging

In this chapter, we examine two crucial implementation considerations. First, we address expressing the results of symmetric merge and projected merge in a finite representation. Second, we consider limiting the range of $EQ$ in symmetric merge and projected merge.

4.1 Finiteness Representation of Knowledge Base Merging

Formal definitions of symmetric merge and projected merge are presented in Sections 3.4.1 and 3.4.2, leading to a deductively closed belief set in each case. After introducing our notations, we prove that logically equivalent operators can be defined so that they yield a resulting knowledge base consisting of a finite formula.

For a given set $K$ of formulas, we denote the conjunction of the formulas in $K$ by $\land K$. For any $\alpha^i \in L_{pi}$ with $i > 0$, we define $\text{nosup}(\alpha^i)$ as the formula in $L_P$ obtained from $\alpha^i$ by dropping all superscripts appearing in $\alpha^i$; $\text{nosup}(K)$ is analogously defined on any set $K$ of formulas or any set $K$ of sets of formulas.

Let $i, j \in \mathbb{Z}^+$ and $EQ$ be a set of equivalences. Then we define $\sigma_{i,j}^{EQ} = \sigma_{i,j}^{+} \cup \sigma_{i,j}^{-}$ as a set of atomic substitutions, where $\sigma_{i,j}^{+} = \{p^i/p^j \mid (p^i \equiv p^j \in EQ \text{ or } i = j) \text{ and } p \in \mathcal{P}\}$ and $\sigma_{i,j}^{-} = \{p^i/p^j \mid (p^i \equiv p^j) \notin EQ, \text{ } p \in \mathcal{P}, \text{ and } i \neq j\}$. In other words, $\sigma_{i,j}^{EQ}$ specifies the substitutions of the corresponding (possibly negations of) $j$-superscripted atoms for $i$-superscripted atoms. For any $\alpha^k \in L_{pk}$ with $k > 0$, $\alpha^k \sigma_{i,j}^{EQ}$ is the formula obtained from...
\( \alpha^k \) by applying substitutions according to \( \sigma^{EQ}_{i,j} \). \( K\sigma^{EQ}_{i,j} \) is analogously defined on any set \( K \) of formulas or any set \( K \) of sets of formulas.

Any \( EQ \) determining a consistent projected belief change extension of a given consistent multi belief change scenario \( ((K_j)_{j \in J}, R, C) \) induces a binary partition of each alphabet \( P^j \) with \( j \in J \), via \( P_{EQ} = \{ p^j \mid p^j \equiv p \in EQ \} \) and \( P_{\overline{EQ}} = (\bigcup_{j \in J} P^j) \setminus P_{EQ} \).

Via Definitions 7 and 8 for symmetric merge, we consider maximal sets \( EQ \) such that \( Cn(\bigcup_{j \in J} K^j \cup R^j \cup EQ) \cap (C^j \cup \{ \bot \}) = \emptyset \). For each such set \( EQ_i \), we carry out substitutions in each \( K^j \) as follows, where \( \uparrow KB \uparrow_i \) is a finite propositional formula comprising part of the finite representation of the symmetric belief change extension determined by \( EQ_i \).

\[(S1) \quad \uparrow KB \uparrow_i := \bot.\]
\[(S2) \quad \text{For each } j \in J \{\]
\[(S3) \quad \uparrow K_j \uparrow_i := \bigwedge \text{nosup}(K^j_i).\]
\[(S4) \quad \text{For each } m \in J \setminus \{j\} \{\]
\[(S5) \quad \uparrow K_j \uparrow_i := \uparrow K_j \uparrow_i \land \bigwedge \text{nosup}( K^m_{\sigma^{EQ}_{m,j}}).\]
\}\n\[(S6) \quad \uparrow KB \uparrow_i := \uparrow KB \uparrow_i \lor \uparrow K_j \uparrow_i.\]
\}

**Definition 11 (Finite Representation of Symmetric Merge)** Let \( B = ((K^j)_{j \in J}, R, C) \) be a multi belief change scenario in \( L_P \), and \( (EQ_i)_{i \in I} \) be the family of all sets of equivalences as defined in Definition 7. Also, let \( \uparrow KB \uparrow_i \) be as given by the substitutions (S1)-(S6) with respect to a given \( EQ_i \).

Then, we define

1. \( \uparrow B \uparrow_c \) as \( \uparrow KB \uparrow_i \) for some selection function \( c \) with \( c(I) = i \), and

2. \( \uparrow B \uparrow \) as \( \bigvee_{i \in I} \uparrow KB \uparrow_i \).

Accordingly, we define

1. \( \uparrow (\{K_1, K_2\}, R, C) \uparrow_c \land (\land R) \) as the finite representation of \( \triangle_c (\{K_1, K_2\}, R, C) \),

2. \( \uparrow (\{K_1, K_2\}, R, C) \uparrow \land (\land R) \) as the finite representation of \( \triangle (\{K_1, K_2\}, R, C) \).

We obtain the following result whose proof is given in Appendix A.1.
CHAPTER 4. IMPLEMENTATION CONSIDERATIONS FOR MERGING

Theorem 9 (Finite Symmetric Merge)

Let $B = (\{K_1, K_2\}, R, C)$ be a multi belief change scenario in $\mathcal{L}_p$, and $(EQ_i)_{i \in I}$ be the family of all sets of equivalences determining the symmetric belief change extensions of $B$. Also, let $\uparrow K_1 \uparrow_i$ and $\uparrow K_2 \uparrow_i$ be as given by the substitutions $(S1)$-$\text{(S6)}$ with respect to a given $EQ_i$.

Then, we have $\Delta (\{K_1, K_2\}, R, C) = \bigvee_{i \in I} (\uparrow K_1 \uparrow_i \lor \uparrow K_2 \uparrow_i) \land (\land R)$.

Consider, as an example, $B = (\{K_1, K_2\}, \{q\}, \emptyset)$, where $K_1 = \{p\}$ and $K_2 = \{-p\}$.

$\Delta (\{K_1, K_2\}, \{q\}, \emptyset)$ yields only one $EQ$ set $EQ_1 = \{q^1 \equiv q^2\}$, and thus $\uparrow K_1 \uparrow_1 = (p \land \neg \neg p) \equiv p$ and $\uparrow K_2 \uparrow_1 = (\neg p \land \neg p) \equiv \neg p$, resulting in the belief set $Cn(\{(p \lor \neg p) \land q\})$.

We now shift our attention to projected merge. Via Definitions 9 and 10 for projected merge, we consider maximal sets $EQ$ such that $Cn(\bigcup_{j \in J} K_j^i \cup R \cup EQ) \cap (C \cup \{\bot\}) = \emptyset$.

For each such set $EQ_i$, we obtain the finite propositional formula $\downarrow KB \downarrow_i$ comprising part of the finite representation of the projected belief change extension determined by $EQ_i$; we do this by carrying out the following substitutions in $\bigwedge_{j \in J} (\land K_j^i)$:

(P1) for $(p^j \equiv p) \in EQ_i$, substitute $p$ for $p^j$, and

(P2) for $(p^j \equiv p) \notin EQ_i$, substitute $\neg p$ for $p^j$.

Definition 12 (Finite Representation of Projected Merge) Let $B = ((K_j)_{j \in J}, R, C)$ be a multi belief change scenario in $\mathcal{L}_p$, and $(EQ_i)_{i \in I}$ be the family of all sets of equivalences as defined in Definition 9. Also, let $\downarrow KB \downarrow_i$ be as given by the substitutions (P1) and (P2) with respect to a given $EQ_i$.

Then, we define

1. $\downarrow B \downarrow_i$ as $\downarrow KB \downarrow_i$ for some selection function $c$ with $c(I) = i$, and

2. $\downarrow B \downarrow$ as $\bigvee_{i \in I} \downarrow KB \downarrow_i$.

Accordingly, we define

1. $\downarrow ((K_1, K_2), R, C) \downarrow \land (\land R)$ as the finite representation of $\nabla_c ((K_1, K_2), R, C)$,

2. $\downarrow ((K_1, K_2), R, C) \downarrow \land (\land R)$ as the finite representation of $\nabla ((K_1, K_2), R, C)$.

We have the following result whose proof is given in Appendix A.2.
CHAPTER 4. IMPLEMENTATION CONSIDERATIONS FOR MERGING

Theorem 10 (Finite Projected Merge)
Let $B = ([K_1, K_2], R, C)$ be a multi belief change scenario in $\mathcal{L}_P$, and $(EQ_i)_{i \in I}$ be the family of all sets of equivalences determining the projected belief change extensions of $B$. Also, let $\\downarrow KB_i$ be as given by the substitutions $(P1)$ and $(P2)$ in $(\land K_1^i) \land (\land K_2^i)$ with respect to a given $EQ_i$.

Then, we have $\nabla ([K_1, K_2], R, C) = \bigvee_{i \in I}(\downarrow KB_i) \land (\land R)$.

Consider, as an example, $\nabla ([K_1, K_2], \{q\}, \emptyset)$, where $K_1 = \{p\}$ and $K_2 = \{-p\}$. We get only two $EQ$ sets $EQ_1 = \{q^1 \equiv q, q^2 \equiv q, p^1 \equiv p\}$ and $EQ_2 = \{q^1 \equiv q, q^2 \equiv q, p^2 \equiv p\}$, and thus $\downarrow KB_1 = (p \land \neg p) \equiv p$ and $\downarrow KB_2 = (\neg p \land \neg p) \equiv \neg p$ accordingly. The belief set resulting from skeptical projection is, therefore, $\wedge n(\{(p \lor \neg p) \land q\})$.

4.2 Vocabulary-Restricted Knowledge Base Merging

Let $\mathcal{P}(\phi)$ be the set of all atoms appearing in formula, or set of formulas, $\phi$. Consider any subset $Q$ of any alphabet $\mathcal{P}$ and any formula $\alpha \in \mathcal{L}_P$. We denote by $\alpha^j|Q$ the formula the same as $\alpha$ but with all atoms also in $Q$ annotated with the superscript $j$. This notation is extended similarly to sets of formulas.

Intuitively, if an atom appears in a knowledge base $K$ but not in the revision formula with which $K$ is revised, or vice versa, then that atom plays no part in the revision process. Indeed, it is proved in [15] that for computing a belief change extension of a single belief change scenario $B = (K, R, C)$, we need consider only those atoms common to both $K$ and $(R \cup C)$.

Similarly, for a multi belief change scenario $B = ((K_j)_{j \in J}, R, C)$, if an atom appears only in one source knowledge base $K_i$, then it does not interfere or pose a challenge in the merging process. To begin with, we first define the vocabulary-restricting sets $KCA$ and $CA$ of common atoms.

Definition 13 (Vocabulary-Restricting Common Atoms) Let $B = ((K_j)_{j \in J}, R, C)$ be a multi belief change scenario in $\mathcal{L}_P$. Then, we define

- $KCA \subseteq \mathcal{P}$ as the set of common atoms among $(K_j)_{j \in J}$ such that $KCA = \{p \mid p \in (\mathcal{P}(K_j) \cap \mathcal{P}(K_m)), j \neq m, \text{ and } j, m \in J\}$,

- $CA \subseteq \mathcal{P}$ as the set of atoms that $(K_j)_{j \in J}$ shares with $R$ or with $C$ such that $CA = \bigcup_{j \in J}(\mathcal{P}(K_j) \cap (\mathcal{P}(R) \cup \mathcal{P}(C)))$. 
Definitions 7 and 8 for symmetric merge are modified to apply to the vocabulary-restricting set $KCA$.

**Definition 14 (Vocabulary-Restricted Symmetric Belief Change Extension)** Let $B = ((K_j)_{j \in J}, R, C)$ be a multi belief change scenario in $L_P$, and a maximal equivalence set $EQ[KCA] \subseteq \{ p^m \equiv p^l \mid p \in KCA, p \in \mathcal{P}(K_m) \cap \mathcal{P}(K_l), m \neq l, and m, l \in J \}$ be such that $\text{Cn}(\bigcup_{j \in J} K_j^I[KCA] \cup R^I[KCA] \cup EQ[KCA]) \cup (C^J[KCA] \cup \{ \bot \}) = \emptyset$. Then,

\[
\{ \alpha \mid \alpha^j \cap \mathcal{P}(\alpha) \neq \emptyset \text{ and } j \in J \} \subseteq \text{Cn}(\bigcup_{j \in J} K_j^I[KCA] \cup R^I[KCA] \cup EQ[KCA])
\]

is a consistent vocabulary-restricted symmetric belief change extension of $B$. If $KCA \neq \emptyset$ and there is no such set $EQ[KCA]$, then $B$ is inconsistent and $L_P$ is defined to be the sole (inconsistent) vocabulary-restricted symmetric belief change extension of $B$.

**Definition 15 (Vocabulary-Restricted Symmetric Merge)** Letting $(E_i)_{i \in I}$ be the family of all vocabulary-restricted symmetric belief change extensions of $B$, we define

1. $\triangle_c[KCA] ((K_j)_{j \in J}, R, C) = E_i$ as a vocabulary-restricted symmetric choice merge of $(K_j)_{j \in J}$ with respect to $R, C$, and some selection function $c$ with $c(I) = i$.

2. $\triangle[KCA] ((K_j)_{j \in J}, R, C) = \bigcap_{i \in I} E_i$ as the vocabulary-restricted symmetric (skeptical) merge of $(K_j)_{j \in J}$ with respect to $R$ and $C$.

We get the following result whose proof can be found in Appendix A.3.

**Theorem 11 (Vocabulary-Restricted Symmetric Merge)** Let $B = \{ K_1, K_2 \}, R, C \}$ be a multi belief change scenario in $L_P$. We have $\triangle ((K_1, K_2), R, C) \equiv \triangle[KCA] ((K_1, K_2), R, C)$.

Similarly, Definitions 9 and 10 for projected merge are modified to apply to the vocabulary-restricting sets $KCA$ and $CA$.

**Definition 16 (Vocabulary-Restricted Projected Belief Change Extension)** Let $B = ((K_j)_{j \in J}, R, C) be a multi belief change scenario in $L_P$, and $W = KCA \cup CA$ where $KCA$ and $CA$ are sets of atoms as given in Definition 13.
First define $EQ[w] \subseteq \{ p^j \equiv p \mid p \in W, p \in \mathcal{P}(K_j), \text{ and } j \in J \}$ as a maximal equivalence set such that $Cn(\bigcup_{j \in J} K_j^j[w] \cup R \cup EQ[w]) \cap (C \cup \{ \bot \}) = \emptyset$.

Then, $Cn(\bigcup_{j \in J} K_j^j[w] \cup R \cup EQ[w]) \cap \mathcal{L}_p$ is a consistent vocabulary-restricted projected belief change extension of $B$. If $W \neq \emptyset$ and there is no such set $EQ[w]$, then $B$ is inconsistent and $\mathcal{L}_p$ is defined to be the sole (inconsistent) vocabulary-restricted projected belief change extension of $B$.

**Definition 17 (Vocabulary-Restricted Projected Merge)** Letting $(E_i)_{i \in I}$ be the family of all vocabulary-restricted projected belief change extensions of $B$, we define

1. $\nabla_c [w] ((K_j)_{j \in J}, R, C) = E_i$ as a vocabulary-restricted projected choice merge of $(K_j)_{j \in J}$ with respect to $R$, $C$, and some selection function $c$ with $c(I) = i$.

2. $\nabla [w] ((K_j)_{j \in J}, R, C) = \bigcap_{i \in I} E_i$ as the vocabulary-restricted projected (skeptical) merge of $(K_j)_{j \in J}$ with respect to $R$ and $C$.

We obtain the following result analogous to Theorem 11. Its proof is given in Appendix A.4.

**Theorem 12 (Vocabulary-Restricted Projected Merge)** Let $B = ((K_j)_{j \in J}, R, C)$ be a multi belief change scenario in $\mathcal{L}_p$. We have $\nabla ((K_j)_{j \in J}, R, C) \equiv \nabla [w] ((K_j)_{j \in J}, R, C)$.
Chapter 5

Implementation CBBC

In this chapter, we discuss our implementation of Delgrande and Schaub's framework [16, 17] described in Chapter 3 (particularly in Section 3.4). Our implementation is named CBBC, an acronym for consistency-based belief change. Taking into account the considerations for finite and vocabulary-restricted representation presented in Chapter 4, we implement both symmetric and projected merge with the possible incorporation of entailment-based and consistency-based integrity constraints. In the following sections within this chapter, we address CBBC's algorithm, its data structures, and its underlying BDD solver. We also provide detailed examples, a brief user manual, and experimental results.

5.1 Algorithm

The finite and vocabulary-restricted results of Chapter 4 lead to algorithms for computing an arbitrary belief change extension of a multi belief change scenario. Specifically, we develop algorithms ComputeSymmetricBCE and ComputeProjectedBCE to compute a symmetric belief change extension and a projected belief change extension, respectively, of any given multi belief change scenario.

Given a set $(K_j)_{j \in J}$ of $|J|$ sets of formulas in $\mathcal{L}_P$, and sets $IC_e$ and $IC_c$ of formulas in $\mathcal{L}_P$ as entailment-based integrity constraints and consistency-based integrity constraints, respectively, we formalize the belief change scenario $B = ((K_j)_{j \in J}, IC_e, IC_c)$ with $\overline{IC_c} = \{ \neg \phi \mid \phi \in IC_c \}$.

Algorithm ComputeSymmetricBCE($(K_j)_{j \in J}, IC_e, IC_c$) returns a formula whose deductive closure represents a symmetric belief change extension of the given $B$. 

34
Algorithm ComputeSymmetricBCE\((K_j)_{j \in J}, \ IC_e, \ IC_c\)

A1. If \(\text{CheckPreliminaryConsistency}((K_j)_{j \in J}, \ IC_e, \ IC_c) = \bot\), then return \(\bot\).

A2. \(KCA := \text{FindKCA}((K_j)_{j \in J})\).

A3. For each \(j \in J\) {
   \(K_j^l := \text{Number}(K_j, \ KCA, \ j)\).
   \(IC_e^l := \text{Number}(IC_e, \ KCA, \ j)\).
   \(IC_c^l := \text{Number}(IC_c, \ KCA, \ j)\).
}

A4. \(KIC_eIC_c := \text{Initialize}((K_j^l)_{j \in J}, \ \cup_{j \in J} IC_e^l, \ \cup_{j \in J} IC_c^l)\).

A5. \(In := B_1\).

A6. For each \(e \in \{\text{getBDD}(p^i \equiv p^j) \mid p \in KCA, \ i \neq j, \ i, j \in J, \ \) and \(p \in (\text{Atoms}(K_j) \cap \text{Atoms}(K_i))\}\) {
   If (for all \(\theta \in KIC_eIC_c\), \(\text{getAndBDD}(e, \ \{\theta\}, \ In) \neq B_0)\)
   Then \(In := \text{getAndBDD}(In, \ e)\).
}

A7. \(K_{all} := \bot\).

A8. For each \(i \in J\) {
   \(K_{curr} := (\land K_i^l)\).
   \(\) For each \(j \in J \setminus \{i\}\) {
   \(K_{other} := (\land K_j^l)\).
   \(\) For each \(p \in KCA\) such that \(p \in \text{Atoms}(K_i) \cap \text{Atoms}(K_j)\) {
   If \((p^i \equiv p^j) \in In\)
   Then \(K_{other} := \text{Replace}(K_{other}, \ p^j, \ p^i)\).
   Else \(K_{other} := \text{Replace}(K_{other}, \ p^i, \ \neg p^i)\).
   \(\) End If.
   
   \(K_{curr} := K_{curr} \land K_{other}\).
   
   \(\) End For.
   
   \(K_{curr} := \text{noSuperscripts}(K_{curr})\).
   \(K_{all} := K_{all} \lor K_{curr}\).
}

A9. Return \(K_{all} \land (\land IC_e)\).

Polynomial-time algorithms ComputeSymmetricBCE and ComputeProjectedBCE first re-express the source knowledge bases \((K_j)_{j \in J}\) in disjoint languages \((P^j)_{j \in J}\) by annotating
the common atoms appearing in each $K_j$ with the superscript $j$; $\text{ComputeSymmetricBCE}$ also similarly re-expresses $IC_e$ and $IC_c$ in $(P^j)_{j \in J}$. Next, via one iteration over all (polynomially many) possible atomic equivalences in an arbitrary order, they find a maximal equivalence set $EQ$ such that $EQ$ is consistent with the union of the re-expressed $\bigcup_{j \in J} K_j^j$, $IC_e$ and $\{\psi\}$ for each $\psi \in IC_c$. Lastly, they re-express $(K_j^j)_{j \in J}$ in the original language (i.e. back to $(K_j)_{j \in J}$) according to $EQ$, and obtain the resulting belief change extension by conjoining all the formulas in the re-expressed $(K_j^j)_{j \in J}$ with the formulas in the original $IC_e$.

Similarly, algorithm $\text{ComputeProjectedBCE}((K_j^j)_{j \in J}, IC_e, IC_c)$ returns a formula whose deductive closure represents a projected belief change extension of the given $B$.

**Algorithm** $\text{ComputeProjectedBCE}((K_j^j)_{j \in J}, IC_e, IC_c)$

B1. If $\text{CheckPreliminaryConsistency}((K_j^j)_{j \in J}, IC_e, IC_c) = \bot$, then return $\bot$.
B2. $CA := (\bigcup_{j \in J} \text{Atoms}(K_j)) \cap (\text{Atoms}(IC_e) \cup \text{Atoms}(IC_c))$.
B3. $KCA := \text{FindKCA}((K_j^j)_{j \in J})$.
B4. For each $j \in J$, $K_j^j := \text{Number}(K_j, KCA \cup CA, j)$.
B5. $KIC_eIC_c := \text{Initialize}((K_j^j)_{j \in J}, IC_e, IC_c)$.
B7. For each $e \in \{\text{getBDD}(p^i \equiv p) \mid p \in (KCA \cup CA), i \in J, \text{and } p \in \text{Atoms}(K_i)\}$ {
   If (for all $\theta \in KIC_eIC_c$, $\text{getAndBDD}(e, \{\theta\}, In) \neq B_0$)
   Then $In := \text{getAndBDD}(In, e)$.
}
B9. For each $i \in J$
   $K_{curr} := (\land K_i^j)$.
   For each $p \in (KCA \cup CA)$ such that $p \in \text{Atoms}(K_i)$ {
   If $(p^i \equiv p) \in In$
   Then $K_{curr} := \text{Replace}(K_{curr}, p^i, p)$.
   Else $K_{curr} := \text{Replace}(K_{curr}, p^i, \neg p)$.
   End If.
   }
   $Kall := Kall \land K_{curr}$.
}
B10. Return $Kall \land (\land IC_e)$. 
Algorithms \textit{ComputeSymmetricBCE} and \textit{ComputeProjectedBCE} invoke other functions: \textit{CheckPreliminaryConsistency}, \(B_1\), \(B_0\), \textit{Atoms}, \textit{FindKCA}, \textit{Number}, \textit{Initialize}, \textit{getAndBDD}, \textit{getBDD}, \textit{Replace}, and \textit{noSuperscripts} (only by \textit{ComputeSymmetricBCE}). These auxiliary functions are defined as follows.

- **\textit{CheckPreliminaryConsistency}((K_j)_{j \in J}, IC_e, IC_c)**
  
  \begin{enumerate}
  \item \(IC_c \text{BDD} := \text{getBDD}(IC_c)\).
  \item If \(IC_c \text{BDD} = B_0\), then return \(\bot\).
  \item For each \(j \in J\)
    
    If \(\text{getBDD}(K_j) = B_0\), then return \(\bot\).
  \item If (for any \(\psi \in IC_e\), \(\text{getBDD}(\psi) = B_0\) or
    
    \(\text{getAndBDD}(IC_e \text{BDD}, \text{getBDD}(\psi)) = B_0\)), then return \(\bot\).
  \item Return \(T\).
  \end{enumerate}

- \(B_1\) is a constant function always returning the BDD composed solely of the terminal node 1.

- \(B_0\) is a constant function always returning the BDD composed solely of the terminal node 0.

- \textit{Atoms}(S), given a set \(S\) of formulas, returns the set of all atoms appearing in any formula in \(S\).

- \textit{FindKCA}((K_j)_{j \in J})

  \begin{enumerate}
  \item \(KCA := \emptyset\).
  \item For each \(i\) from 1 to \(|J| - 1\) {
    \begin{enumerate}
    \item For each \(j\) from \(i + 1\) to \(|J|\) {
      \begin{enumerate}
      \item \(KCA := KCA \cup (\text{Atoms}(K_i) \cap \text{Atoms}(K_j))\).
      \end{enumerate}
    \end{enumerate}
  \end{enumerate}
  \item Return \(KCA\).  
  \end{enumerate}
CHAPTER 5. IMPLEMENTATION CBBC

- \textit{Number}(K_j, KCA, j), given a set \( K_j \) of formulas, a set \( KCA \) of atoms, and an integer \( j \), returns a set of formulas obtained from \( K_j \) with atoms also in \( KCA \) superscripted with \( j \).

- \textit{Initialize}(\( (K_j^j)_{j \in J}, IC_e, IC_c \)), given a set \( (K_j^j)_{j \in J} \) of sets of formulas, and sets \( IC_e \) and \( IC_c \) of formulas, returns a collection of BDDs corresponding to formulas of the form \((\bigwedge_{j \in J} (\bigwedge K_j^j) \land (\bigwedge IC_e) \land \psi))\), for each \( \psi \in (IC_e \cup \{T\}) \).

- \textit{getAndBDD}(A, B) (or \textit{getAndBDD}(A, B, C)), given BDDs as arguments, returns a BDD representing the conjunction of the input BDDs.

- \textit{getBDD}(IC_e), given a set \( IC_e \) of formulas, returns a BDD representing the conjunction of all formulas in \( IC_e \); analogously, if given a formula \( \psi \), \textit{getBDD}(\( \psi \)) returns a BDD representing \( \psi \).

- \textit{Replace}(K_{curr}, p^i, p), given a formula \( K_{curr} \), an atom \( p^i \), and a literal \( p \), returns \( K_{curr} \) with every occurrence of \( p^i \) replaced by \( p \).

- \textit{noSuperscripts}(K_{curr}), given a formula \( K_{curr} \), returns \( K_{curr} \) but with all its superscripts removed.

Further details regarding ComputeSymmetricBCE and ComputeProjectedBCE are given in subsequent sections. Specifically, Section 5.3 elaborates on the consistency checks in ComputeSymmetricBCE and ComputeProjectedBCE, while Section 5.2 discusses the search for EQ sets. In addition, examples illustrating how ComputeSymmetricBCE and ComputeProjectedBCE operate are included in Section 5.4.

5.2 Data Structures

We chose Java as CBBC's implementation language over C++ and C for several reasons. First, while enjoying the benefits of Java's object-oriented paradigm, thanks to Java’s automatic garbage collection, we can dispense with manual memory allocations and deallocations. Second, in addition to Java's portability to different platforms, Java allows for native code via the Java Native Interface technology, and thus the integration of existing BDD libraries written in C and C++. Third, the ease with which user-friendly interfaces can be created in Java would accommodate any future adaptation of CBBC into a Java applet.
In accordance with the object-oriented paradigm, CBBC has the following Java classes with self-explanatory names reflecting the central entities in the belief change framework:

- **BCS**: a belief change scenario,
- **CASet**: a Hashtable whose entries are of the form \((key, value)\), where \(key\) is an atomic string and \(value\) the corresponding set of indices of the source knowledge bases in which the atomic string appears,
- **EquivalenceSet**: a set of atomic equivalences, and
- **IFormula**: an abstract class specifying the interface and some default implementations for the concrete sub-classes Atom, Equivalence, Implication, Contradiction, Tautology, Negation, Conjunction and Disjunction.

There are four additional classes. Class **CBBCApp** is our main application with a user-interactive task menu and a reference to the user-specified BCS to be computed. Class **InputParser** parses strings in input files into corresponding instances of **IFormula**'s concrete sub-classes. Auxiliary classes **InconsistencyException** and **SyntaxErrException** encapsulate an inconsistency error message and a syntax error message, respectively, for input checks (see Section 5.5.2).

Class **BCS** has several characterizing attributes, among others, including:

- a Vector \(K\) of the source knowledge bases,
- a Vector \(IC_e\) of the entailment-based integrity constraints,
- a Vector \(IC_c\) of the consistency-based integrity constraints,
- Boolean flags specifying the type of belief change (skeptical or choice) and the type of merge (symmetric or projected),
- a Vector \(maxConsEQSets\) of bit-strings representing all the maximal consistent equivalence sets,
- a Vector \(minInconsEQSets\) of bit-strings representing all the minimal inconsistent equivalence sets,
- a CASet \(commonAtoms\) of common atoms \((KCA\) for symmetric merge vs. \(KCAUC\) for projected merge) and its corresponding EquivalenceSet \(CAEQSet\), and
• a Hashtable $atomToBDDTable$ mapping an atomic string to its BDD representation.

We refer readers interested in CBBC’s class hierarchy tree and function specifications to CBBC’s JavaDocs included with the source code that can be downloaded from http://www.cs.sfu.ca/~cl/software/CBBC/CBBC.zip. Also available in the download are JavaDocs for JavaBDD [63], including classes BDDFactory and BDD, among others.

To interact with the BDD libraries via the JavaBDD interface [63], BCS also has a static (i.e. shared by all BCS instances) BDDFactory reference responsible for the BDD operations involved in checking satisfiability. Furthermore, each concrete sub-class of IFormula implements the function $getBDD(atomToBDDTable)$, which not only updates $atomToBDDTable$ to facilitate the re-use of BDDs, but also returns a BDD instance representing the sub-class instance. To further conserve computation resources, some BDD instances, once created, are retained for later or repeated use. For example, each atomic equivalence in BCS’s $CAEQSet$ is stored as a BDD, and BDD instances corresponding to formulas in $KICcICc$ in algorithms $ComputeSymmetricBCE$ and $ComputeProjectedBCE$ from Section 5.1 are saved in class BCS.

The cardinality of BCS’s $CAEQSet$ varies according to the type of merge to be computed. As an example, suppose $commonAtoms$ has the entry < “p”, {1, 2, 3, 4} >. Then, for symmetric merge, whose $commonAtoms = KCA$, the corresponding atomic equivalences $(p^1 \equiv p^2), (p^1 \equiv p^3), (p^2 \equiv p^3), (p^2 \equiv p^4)$, and $(p^3 \equiv p^4)$ are added to $CAEQSet$; for projected merge, whose $commonAtoms = KCA \cup CA$, $(p^1 \equiv p), (p^2 \equiv p), (p^3 \equiv p)$, and $(p^4 \equiv p)$ are added to $CAEQSet$.

The selection function for the “preferred” EQ set is left implicit in line A6 of algorithm $ComputeSymmetricBCE$ and in line B7 of algorithm $ComputeProjectedBCE$ in Section 5.1; it is realized by the particular order chosen when treating the common atoms. In CBBC, though, we create an ordered (in ascending cardinality) list $L$ of all $2^{|CAEQSet|}$ possible subsets of $CAEQSet$. To help streamline the search for EQ sets and minimize memory usage, we represent each atomic equivalence in $CAEQSet$ by a single bit such that it is included in an equivalence set $e$ iff its corresponding bit is 1 in $e$’s bit-string.

Albeit opaque to users, $L$ can be ordered in such a way as to accommodate either breadth-first search (BFS) or depth-first search (DFS) of EQ sets. As an example, consider the projected merge of $\{(\{a \land b \land c\}, \{s\}), (\neg a \land \neg b \land \neg c), \emptyset\}$, whose $CAEQSet$ is $\{a^1 \equiv a, b^1 \equiv b, c^1 \equiv c\}$, but whose sole EQ set is $\emptyset$. Then BFS produces the ordered list $L = 111, 011, 101, 110, 001, 010, 100, 000$, thereby requiring seven satisfiability checks;
DFS yields the ordered list \( L = 000, 001, 011, 111, 101, 010, 110, 100 \), thereby requiring only three satisfiability checks.\(^1\) As another example, consider the projected merge of \( (\{a \wedge b \wedge c\}, \{s\}, \{a \vee b \vee c\}, \emptyset) \), whose sole EQ set is precisely \( \text{CAEQSet} \). Then BFS requires only one satisfiability check, while DFS requires three satisfiability checks.\(^1\) These two examples affirm our observation that on average, the running time and memory usage of BFS are comparable to those of DFS, with neither search being consistently preferable. Currently in CBBC, \( L \) arbitrarily lists the all-1s bit-string (corresponding to precisely \( \text{CAEQSet} \)) first, followed by the rest of the bit-strings\(^1\) in ascending cardinality order.

Class BCS keeps \( \text{maxConsEQSets} \) and \( \text{minInconsEQSets} \) in order to minimize the number of satisfiability checks.\(^2\) Given an equivalence set \( Q \), we first check whether it is a superset of any equivalence set in \( \text{minInconsEQSets} \) or a subset of any equivalence set in \( \text{maxConsEQSets} \). If the former holds, \( Q \) must not be a determining EQ set for BCS and is, thus, discarded. If the latter holds, \( Q \), although a consistent equivalence set for BCS, is not maximal and thus is also discarded. Only if neither holds do we proceed with the satisfiability check on \( Q \) for BCS. Subsequently, if \( Q \) is ascertained as a consistent equivalence set, it is added to \( \text{maxConsEQSets} \) only if none of its supersets are already in \( \text{maxConsEQSets} \); otherwise (i.e. \( Q \) is deemed an inconsistent equivalence set), it is added to \( \text{minInconsEQSets} \) only if none of its subsets are already in \( \text{minInconsEQSets} \).

In practice, algorithms for computing symmetric merge or projected merge operate in exponential time, as deciding whether a given multi belief change scenario has a symmetric or projected belief change extension takes at most deterministic exponential time with the dominating factor being the \( 2^{\vert \text{CAEQSet} \vert} \) possible equivalence sets to consider. Alternatively, we can non-deterministically pick a set of polynomially many atomic equivalences and verify whether it is a determining EQ set in polynomial time, as we have observed in algorithms \( \text{ComputeSymmetricBCE} \) and \( \text{ComputeProjectedBCE} \) from Section 5.1.

\(^1\)The all-0s bit-string corresponding to the empty equivalence set need not be tested under the assumption that the consistent multi belief change scenario in question has passed \( \text{CheckPreliminaryConsistency} \) from Section 5.1.

\(^2\)The author thanks James Delgrande, Torsten Schaub, Sven Thiele, and Chris Balavessov for this idea.
5.3 BDD Solver

For consistency checking purposes, CBBC employs JavaBDD [63], a Java interface to the BDD solvers BuDDy [47], CUDD [59], CAL [53], and JDD [61]. As described in Section 5.1, deciding the satisfiability of a set of formulas essentially consists of two steps. First, we represent each formula in the given set of formulas as a BDD. Then we check whether the conjunction of the BDD(s) corresponds to $B_0$ (the BDD composed solely of the terminal node 0); the given set of formulas is inconsistent iff the resulting BDD is $B_0$.

Most widely-used and well-tested BDD libraries in the industry, such as BuDDy [47], CUDD [59], and CAL [53], were written in C and C++. To exploit both the strengths of these BDD libraries and the advantages of Java programming as briefly discussed in Section 5.2, we use JavaBDD [63] to link our CBBC to the BDD libraries BuDDy, CUDD, CAL, and JDD. See [58, 34, 33, 61] for a comparison of these aforementioned and other BDD packages. Our experimental results in Section 5.6 also provide a comparison of these four BDD libraries.

We selected JavaBDD in lieu of the equally well-known JBDD [60], which is an interface to BuDDy and CUDD, for three reasons. First, the more selection of BDD libraries to experiment with in JavaBDD would serve us well in future optimizations of CBBC. Second, JBDD is somewhat more cumbersome to use as it requires us to manually apply reference-counting to BDDs for garbage collection purposes [60]. Lastly, with JavaBDD's uniform interface to external BDD libraries, we can specify at run-time the BDD library to be used without making any changes to CBBC's Java code.

5.4 Examples

We demonstrate how CBBC computes belief change extensions by working through two examples, one for symmetric merge and the other for projected merge.

Consider merging knowledge bases $K_1 = \{p, \neg r\}$ and $K_2 = \{p, r, s\}$, with respect to the entailment-based integrity constraint $IC_e = \{s\}$ and the consistency-based integrity constraints $IC_c = \{p, v\}$.

We first show how CBBC computes the symmetric merge of the multi belief change scenario ($\{\{p, \neg r\}, \{p, r, s\}\}, \{s\}, \{p, v\}$).

1. Find the common atoms appearing in at least two of the knowledge bases $(K_j)_{i \in J}$ to
be merged.

\[ KCA = \{ p, r \} \]

2. Create a conjunction \( K \) of all formulas in \( K^j \) obtained from \( K_j \) by superscripting any atom \( b \) appearing in \( K_j \) with \( j \) if \( b \in KCA \).

\[
K = (\bigwedge K^1_j) \land (\bigwedge K^2_j) = (p^1 \land \lnot r^1) \land (p^2 \land r^2 \land s)
\]

3. Create a conjunction \( R \) of all formulas in \( IC^j_e \) obtained from \( IC_e \) by superscripting any atom \( b \) appearing in \( IC_e \) with \( j \) if \( b \in KCA \).

\[
R = (\bigwedge IC^1_e) \land (\bigwedge IC^2_e) = s \land s = s
\]

4. Create a set \( CB \) of all formulas in \( IC^j_e \) obtained from \( IC_e \) by superscripting any atom \( b \) appearing in \( IC_e \) with \( j \) if \( b \in KCA \).

\[
CB = IC^1_e \cup IC^2_e = \{p^1, v\} \cup \{p^2, v\} = \{p^1, p^2, v\}
\]

5. For each formula \( \phi \in CB \), create a BDD for the conjunction of \( K \), \( R \), and \( \phi \).

\[
KIC_e IC_e BDD_1 = getAndBDD(\text{getBDD}(K), \text{getBDD}(R), \text{getBDD}(p^1))
\]

\[
KIC_e IC_e BDD_2 = getAndBDD(\text{getBDD}(K), \text{getBDD}(R), \text{getBDD}(p^2))
\]

\[
KIC_e IC_e BDD_3 = getAndBDD(\text{getBDD}(K), \text{getBDD}(R), \text{getBDD}(v))
\]

6. Find all maximal equivalence sets \( EQ_i \subseteq \{ b^d \equiv b^e | b \in KCA, d \neq e, d, e \in J, \text{ and } b \in (\text{Atoms}(K_d) \cap \text{Atoms}(K_e)) \} \) such that each \( \text{getBDD}(EQ_i) \) conjoined with each \( KIC_e IC_e BDD_k \) does not yield \( B_0 \).

\[ EQ_1 = \{ p^1 \equiv p^2 \} \]

7. For each \( EQ_i \), create a symmetric belief change extension by doing the following substeps:
   
   (a) for each \( K^j_i \), first express all other \( (K^b_i)^l \in \bigcup \{ j \} \) in \( P^j \) according to \( EQ_i \) (by replacing any atom \( b^l \) appearing in \( K^b_i \) by \( \lnot b^l \) if \( (b^l \equiv b^j) \notin EQ_i \) or by \( b^j \) otherwise), then form a conjunction \( \uparrow K_{j_i} \) of formulas in \( K^j_i \) and all formulas in \( (K^b_i)^l \in I, l \neq j \) expressed in \( P^j \), and finally remove all the superscripts appearing in the conjunction \( \uparrow K_{j_i} \);
   (b) create a disjunction \( \uparrow KB_{j_i} \) with each \( \uparrow K_{j_i} \) as its disjunct;
   (c) create a conjunction \( E_i \) of \( \uparrow KB_{j_i} \) and the formulas in \( IC_e \).

For \( EQ_1 \), we have \( \uparrow K_1 \uparrow_1 = (p \land \lnot r \land p \land \lnot r \land s) = (p \land \lnot r \land s) \) and \( \uparrow K_2 \uparrow_1 = (p \land \lnot r \land p \land r \land s) = (p \land r \land s) \) after step (a); \( \uparrow KB \uparrow_1 = (p \land r \land s) \lor (p \land \lnot r \land s) \) after
step (b); and the resulting conjunction $E_1 = ((p \land r \land s) \lor (p \land \neg r \land s)) \land s$, which is logically equivalent to $(p \land s)$ after step (c). Thus, $\Delta_{c1} (\{\{p, \neg r\}, \{p, r, s\}\}, \{s\}, \{p, v\}) = E_1$ is finitely representable as $(p \land s)$, where $c1(\{1\}) = 1$.

8. The resulting knowledge base is the deductive closure of either the disjunction of all symmetric belief change extensions for skeptical change, or one symmetric belief change extension for choice change.

$\Delta (\{\{p, \neg r\}, \{p, r, s\}\}, \{s\}, \{p, v\})$ is finitely representable as $(p \land s)$.

We now show how CBBC computes the projected merge of the same multi belief change scenario ($\{\{p, \neg r\}, \{p, r, s\}\}, \{s\}, \{p, v\}$).

1. Find the common atoms appearing in at least two of the knowledge bases $(K_j)_{i \in J}$ to be merged.
$KCA = \{p, r\}$

2. Find the common atoms that any knowledge base $K_j$ shares with any formula in $IC_e$ or with any formula in $IC_c$.
$CA = \{p, s\}$

3. Create a conjunction $K$ of all formulas in $K_j^d$ obtained from $K_j$ by superscripting any atom $b$ appearing in $K_j$ with $j$ if $b \in KCA \cup CA$.
$K = (\land K_1^1) \land (\land K_2^2) = (p^1 \land \neg r^1) \land (p^2 \land r^2 \land s^2)$

4. For each formula $\phi \in IC_e$, create a BDD for the conjunction of $K$, the formulas in $IC_e$, and $\phi$.
$KIC_eIC_eBDD_1 = getAndBDD(getBDD(K), getBDD(IC_e), getBDD(p))$
$KIC_eIC_eBDD_2 = getAndBDD(getBDD(K), getBDD(IC_e), getBDD(v))$

5. Find all maximal equivalence sets $EQ_i \subseteq \{b^d \equiv b \mid b \in KCA \cup CA, d \in J, \text{and } b \in Atoms(K_d)\}$ such that each $getBDD(EQ_i)$ conjoined with each $KIC_eIC_eBDD_k$ does not yield $B_0$.
$EQ_1 = \{p^1 \equiv p, p^2 \equiv p, s^2 \equiv s, r^1 \equiv r\}$
$EQ_2 = \{p^1 \equiv p, p^2 \equiv p, s^2 \equiv s, r^2 \equiv r\}$

6. For each $EQ_i$, create a projected belief change extension by doing the following substeps:
(a) replace each atom $b^j$ appearing in $K$ with $\neg b$ if $(b^j \equiv b) \in EQ_i$, or with $b$ otherwise;
(b) create a conjunction $E_i$ of $K$ and the formulas in $IC_e$.

For $EQ_1$, we have $K = (p \land \neg r \land p \land \neg r \land s) = (p \land \neg r \land s)$ after step (a); $E_1 = (p \land \neg r \land s) \land s$, which is logically equivalent to $(p \land \neg r \land s)$ after step (b). Thus, $\nabla_{c1} (\{ \{p, \neg r\}, \{p, r, s\} \}, \{s\}, \{p, v\}) = E_1$ is finitely representable as $(p \land \neg r \land s)$, where $c_1(\{1, 2\}) = 1$.

For $EQ_2$, we have $K = (p \land \neg r \land p \land r \land s) = (p \land r \land s)$ after step (a); $E_2 = (p \land r \land s) \land s$, which is logically equivalent to $(p \land r \land s)$ after step (b). Thus, $\nabla_{c2} (\{ \{p, \neg r\}, \{p, r, s\} \}, \{s\}, \{p, v\}) = E_2$ is finitely representable as $(p \land r \land s)$, where $c_2(\{1, 2\}) = 2$.

7. The resulting knowledge base is the deductive closure of either the disjunction of all projected belief change extensions for skeptical change, or one projected belief change extension for choice change.

$\nabla (\{ \{p, \neg r\}, \{p, r, s\} \}, \{s\}, \{p, v\})$ is finitely representable as $(p \land s)$.

5.5 Using CBBC

CBBC is available as a Java command-line tool, complete with an interactive task menu. Via the menu, users can import a belief change scenario from files, specify the type (skeptical or choice) of belief change desired, and obtain the resulting knowledge base(s). Under the assumption that users have no preferred EQ set (and thus no preferred belief change extension), CBBC computes choice belief change in the same way as it does skeptical belief change.

CBBC has two notable features. First, it alerts users to any syntactically ill-formed or inconsistent input formulas. Section 5.5.2 elaborates on CBBC’s consistency checks and syntactical checks on input formulas. Second, CBBC automatically simplifies formulas where applicable. Let $T$ and $F$ stand for a tautology and a contradiction, respectively. Specifically, these basic simplifications include removing double negation symbols, $F$ disjuncts, and $T$ conjuncts; replacing a disjunction with a $T$ disjunct by $T$; and replacing a conjunction with a $F$ conjunct by $F$.

The Java code and Javadocs of CBBC are accessible from http://www.cs.sfu.ca/
Welcome to the CBBC Command-Line Version.

CBBC Command-Line Task Menu
(1) Specify the path of the KB input file.
(2) Specify the path of the entailment-based integrity constraints input file.
(3) Specify the path of the consistency-based integrity constraints input file.
(4) Specify the type of change (default=skeptical change).
(5) Specify the type of merge (default=symmetric merge).
(6) Execute the belief change scenario.
(7) Exit CBBC

Please select a task by entering its number.

Figure 5.1: CBBC's Interactive Task Menu

cl/software/CBBC/CBBC.zip. Included with the source code are a Makefile for compiling
CBBC, shell scripts for running CBBC, and a ReadMe file with compilation and execution
instructions. The ReadMe file supplements the user manual included here in Section 5.5.

To facilitate our ensuing discussion in Sections 5.5.1 and 5.5.2, we adopt the acronyms
KB, EB IC, and CB IC to denote a knowledge base, an entailment-based integrity constraint,
and a consistency-based integrity constraint, respectively.

5.5.1 Syntax

CBBC accepts almost all alphanumerical strings for atom names. The only exceptions are
the symbols in the following comma-separated list: ', +, &, ^, ~, =, >, ( and ). The
reserved symbols T and F stand for T and I, respectively.

More complex formulas can be constructed from formulas a and b using connectives.

¬a  for the negation of a

(a&b) for the conjunction of a and b

(a+b) for the disjunction of a and b

(a>b) for a implies b

(a=b) for a is logically equivalent to b
A top-level formula with a binary connective (\&, +, \textgreater, or \textasciitilde) must be enclosed in parentheses so that CBBC can parse it correctly. Parentheses within a formula, however, are optional and are used only to enforce parsing preference. For example, \((a\&b+c)\) is a syntactically well-formed input formula and is different from \((a\&(b+c))\), whereas a top-level formula like \(a+b\) is syntactically ill-formed.

Input file formats for belief change scenarios vary according to the list (KB, EB IC, or CB IC) to which formulas are to be added. A KB file to be imported should precede each knowledge base by a line "KB :" (without the double quotes) and list each formula in the knowledge base on a separate line. Each EB IC is listed on a separate line in an EB IC input file. For a CB IC input file, each line is construed as a CB IC independent of the other CB ICs.

As an example, Table 5.1.1 shows some syntactically well-formed input files.

<table>
<thead>
<tr>
<th>KB</th>
<th>EB IC</th>
<th>CB IC</th>
</tr>
</thead>
</table>
| KB :
(a\&b&v)   | (a&b+c)        | y     |
| (\textasciitilde b\textasciitilde u) | (x&(y+z)) | \textasciitilde y |
| KB :
(a>(t=d))  |                |       |

Table 5.1: Example Input Files

5.5.2 Input Checks

CBBC performs both syntactical and consistency checks on all input formulas.

Syntactical Checks

With regard to the syntax detailed in Section 5.5.1, CBBC alerts users to syntactically ill-formed input formulas by displaying informative error messages.

Consider, as examples, the following syntactically ill-formed input strings and the corresponding error messages generated by CBBC.

- a': The illegal literal a' has the reserved symbol '.
- a~: The illegal literal a~ has the reserved symbol ~.
• a>: The illegal literal a> has a reserved logical symbol.

• a): The illegal literal a) contains a parenthesis.

• (a+b&): (a+b&) has no matching right operand for &.

• (+a): (+a) has no matching left operand for +.

• (a: a has no matching closing parenthesis.

• (a&(b): (a&(b) has one or more unmatched parentheses.

Consistency Checks
To preempt inconsistent belief change scenarios (i.e. those whose sole belief change extension is $\mathcal{L}_P$), CBBC prohibits certain kinds or combinations of input formulas that result in inconsistent belief change scenarios. This preventive measure accords well with the consistency checks in function CheckPreliminaryConsistency in Section 5.1.

Users are advised to avoid the following inconsistent belief change scenarios; sample error messages, where applicable, are italicized.

1. a contradiction in a source knowledge base:
   No error message; formula not added.

2. a contradiction as an EB IC:
   No error message; formula not added.

3. a contradiction as a CB IC:
   No error message; formula not added.

4. a contradiction in the set of all EBICs:
   The conjunction of EB ICs is a contradiction!

5. a conflict between EBICs and any CB IC:
   The CB IC indexed 0 is inconsistent with the conjunction of EB ICs (indexing starts at 0)!
CHAPTER 5. IMPLEMENTATION CBBC

The error message in 5. is generated, for example, for a multi belief change scenario with p and q as the EB ICs, \(\neg p\) as the first (indexed 0) CB IC, and \((a+b)\) as the second (indexed 1) CB IC. Since indexing of formulas in a knowledge base starts at 0 in CBBC, the \(i\)-th formula in a knowledge base has the index \((i-1)\).

The aforementioned consistency checks 1.-5. correspond to the consistency checks on input in function \textit{CheckPreliminaryConsistency} in Section 5.1. Specifically, 1. corresponds to C3; 2. and 4. to C2; lastly, 3. and 5. to C4.

5.6 Experiments

Experiments in this section were all conducted in the environment CYGWIN_NT-5.1 1.5.16 on one machine with an Intel Pentium IV CPU of 1.41 GHz, 128 MB of RAM, and the Microsoft XP Professional operating system. These experimental results for both skeptical symmetric merge and skeptical projected merge provide a run-time comparison of the four BDD libraries (BuDDy [47], CUDD [59], CAL [53], and JDD [61]) to which CBBC has access via JavaBDD [63].

For our skeptical symmetric merge experiments, we considered ten multi belief change scenarios indexed \(i = 1, 2, ..., 10\) accordingly. The input multi belief change scenario indexed \(i\) is of the form \((\{K_1, K_2\}, \emptyset, \emptyset)\), where \(K_1 = \bigwedge_{n = 1}^{i} (p_n \land q_n)\) and \(K_2 = \bigwedge_{n = 1}^{i} (\neg p_n \lor \neg q_n)\) involving propositional atoms \(p_1, p_2, ..., p_i\) and \(q_1, q_2, ..., q_i\). These input multi belief change scenarios, each with an exponential (i.e. exponential in the number of clauses in \(K_2\)) number of determining EQ sets, were chosen to facilitate run-time comparisons among the four aforementioned BDD libraries.

Table 5.2 presents our skeptical symmetric merge experimental results. Each row indexed \(i\) comprises four experimental results of the multi belief change scenario indexed \(i\). Column \(i\) lists the index; Column \(EQs\) the number of determining EQ sets; Column \(Poss. EQs\) the total number of possible equivalence sets; Column \(SAT Chks\) the number of satisfiability checks involved in finding all the determining EQ sets; Column \(Skipped\) the number of satisfiability checks avoided thanks to \textit{maxConsEQSets} and \textit{minInconsEQSets} (see Section 5.2). The remaining eight columns show four mutually independent sets of results; Each set consists of \(SAT\): the time in seconds spent on satisfiability checks involved in the search for all determining EQ sets and \(Total\): the total run-time in seconds when one of the four aforementioned BDD libraries was used for all the satisfiability checks. Precisely stated, \(Total\)
CHAPTER 5. IMPLEMENTATION CBBC

### Table 5.2: Skeptical Symmetric Merge Experimental Results

<table>
<thead>
<tr>
<th>i</th>
<th>EQs</th>
<th>Poss. EQs</th>
<th>Chks</th>
<th>Skipped</th>
<th>BuDDy SAT</th>
<th>BuDDy Total</th>
<th>CUDD SAT</th>
<th>CUDD Total</th>
<th>CAL SAT</th>
<th>CAL Total</th>
<th>JDD SAT</th>
<th>JDD Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.0</td>
<td>0.02</td>
<td>0.0</td>
<td>0.01</td>
<td>0.0</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0.0</td>
<td>0.02</td>
<td>0.0</td>
<td>0.02</td>
<td>0.0</td>
<td>0.03</td>
<td>0.0</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>64</td>
<td>24</td>
<td>0</td>
<td>0.0</td>
<td>0.05</td>
<td>0.0</td>
<td>0.05</td>
<td>0.0</td>
<td>0.05</td>
<td>0.0</td>
<td>0.06</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>256</td>
<td>97</td>
<td>0</td>
<td>0.02</td>
<td>0.13</td>
<td>0.0</td>
<td>0.11</td>
<td>0.02</td>
<td>0.15</td>
<td>0.02</td>
<td>0.12</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>1024</td>
<td>350</td>
<td>0.01</td>
<td>0.201</td>
<td>0.09</td>
<td>0.01</td>
<td>0.19</td>
<td>0.03</td>
<td>0.21</td>
<td>0.03</td>
<td>0.21</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>4096</td>
<td>217</td>
<td>0.03</td>
<td>0.431</td>
<td>0.1</td>
<td>0.01</td>
<td>0.421</td>
<td>0.03</td>
<td>0.46</td>
<td>0.04</td>
<td>0.45</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>2144</td>
<td>454</td>
<td>0.03</td>
<td>1.873</td>
<td>0.17</td>
<td>0.01</td>
<td>1.613</td>
<td>0.17</td>
<td>1.762</td>
<td>0.06</td>
<td>1.663</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>216</td>
<td>962</td>
<td>0.07</td>
<td>11.436</td>
<td>0.38</td>
<td>0.09</td>
<td>11.236</td>
<td>0.38</td>
<td>11.096</td>
<td>0.06</td>
<td>10.926</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>218</td>
<td>2060</td>
<td>0.22</td>
<td>101.916</td>
<td>1.292</td>
<td>0.11</td>
<td>106.583</td>
<td>1.292</td>
<td>100.314</td>
<td>0.491</td>
<td>99.152</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>220</td>
<td>5219</td>
<td>1.102</td>
<td>1151.535</td>
<td>11.313</td>
<td>0.451</td>
<td>866.666</td>
<td>11.313</td>
<td>871.934</td>
<td>3.125</td>
<td>875.609</td>
</tr>
</tbody>
</table>

is the total run-time for steps A2-A9, inclusive, of algorithm `ComputeSymmetricBCE` from Section 5.1, except that all determining `EQ` sets are found and all corresponding consistent symmetric belief change extensions created. On the other hand, `SAT` only accounts for the total time spent on (for all `θ ∈ KICeICc`, `getAndBDD(e, {θ}, In) ≠ B0`) in `ComputeSymmetricBCE`'s step A6, repeated sufficiently many times to cover all the equivalence sets examined.

Table 5.2 shows that on average for our ten multi belief change scenarios, `CUDD < BuDDy < JDD < CAL` with respect to `SAT` time, while `CUDD < JDD < CAL < BuDDy` with respect to `Total` time. Admittedly, these relative rankings are likely to be particular to our chosen multi belief change scenarios; by no means do they conclusively establish the relative preferences of the four aforementioned BDD libraries in all applications.

For our skeptical projected merge experiments, we considered 13 multi belief change scenarios indexed `i = 1, 2, ..., 13` accordingly. The input multi belief change scenario indexed `i` is of the form `{K1, K2}, Φ, Φ`, where `K1 = ∨_{n = 1}^{i} p_n` and `K2 = ∨_{n = 1}^{i} ¬p_n` involving propositional atoms `p_1, p_2, ..., p_i`. These input multi belief change scenarios, each with an exponential (i.e. exponential in the number of atoms in `K_1`) number of determining `EQ` sets, were chosen to facilitate run-time comparisons among the four aforementioned BDD libraries.

Table 5.3 presents our skeptical projected merge experimental results, using exactly the same format as Table 5.2. Precisely stated, `Total` is the total run-time for steps B2-B10,
inclusive, of algorithm \textit{ComputeProjectedBCE} from Section 5.1, except that all determining \textit{EQ} sets are found and all corresponding consistent projected belief change extensions created. On the other hand, \textit{SAT} only accounts for the total time spent on (for all $\theta \in KIC_e IC_c, getAndBDD(e, \{\theta\}, In) \neq B_0$) in \textit{ComputeProjectedBCE}'s step B7, repeated sufficiently many times to cover all the equivalence sets examined.

Table 5.3 reveals that on average for our 13 multi belief change scenarios, BuDDy < CUDD < JDD < CAL with respect to \textit{SAT} time, while BuDDy < JDD < CUDD < CAL with respect to \textit{Total} time. These relative rankings differ from those derived earlier from Table 5.2. The discrepancy suggests that the rankings are dependent on the particular structure of the chosen multi belief change scenarios, underscoring the fact that at present, no conclusive evidence exists as to the optimal BDD library to use in all applications.

In all rows of both Tables 5.2 and 5.3, the sum of \textit{SAT Chks} and \textit{Skipped} is less than \textit{Poss. EQs}. As a result of \textit{maxConsEQSets} and \textit{minInconsEQSets} (see Section 5.2), not all the possible equivalence sets need be generated or considered. Let $ceq$ denote the equivalence set being currently examined by CBBC, and $B$ be the multi belief change scenario in question. While searching for determining \textit{EQ} sets, CBBC dynamically generates further equivalence sets for consideration only if:

- either $ceq$ includes all the possible atomic equivalences but is an inconsistent equivalence set for $B$,
or $ceq$ is a consistent, not necessarily maximal, equivalence set for $B$.

As illustrated in Section 2.3, the chosen variable ordering has a significant impact on the size of an OBDD constructed, and dynamic variable reordering (DVO) is a size-minimizing technique whereby a BDD library automatically determines appropriate points at which to pause processing, adjust the variable ordering, and then continue processing [56]. Some well-known DVO heuristics are discussed in Section 6.3, where we show experimental results of projected merge using BuDDy [47] with various DVO heuristics via JavaBDD [63].

Although DVO is available in BuDDy, CUDD, and CAL, the current version 1.0b2 of JavaBDD has yet to implement the interface functions to access the corresponding reordering functions in CUDD and CAL. As well, JDD [61] does not have DVO but assumes that users themselves would give a variable reordering if so desired. Given these DVO (or the lack thereof) incongruities, no experiments in Tables 5.2 and 5.3 employed DVO. Nor did we specify a variable ordering as our main focus in CBBC is simply to implement symmetric merge and projected merge. Admittedly, the investigation of optimal BDD variable orderings in Delgrande and Schaub's framework [15, 16, 17] could warrant more attention in the future.

The current version 1.0b2 of JavaBDD [63] recommends the use of BuDDy over the other three BDD libraries, because the interface functions linking to BuDDy have been fully implemented whereas some of those linking to the other three libraries are still incomplete. Accordingly, in Chapter 6, where we compare the implementation of belief revision in CBBC to that in other related or similar systems, we describe our CBBC experiments conducted using only BuDDy via JavaBDD.
Chapter 6

Related Work

In this chapter, we compare CBBC with such related or similar systems as COBA 2.0 [14], QUIP [19], and BReLS [46]. COBA 2.0 [14] is a predecessor of CBBC concentrating on belief revision and contraction incorporating integrity constraints. QUIP [19] is a logical reasoning system with a uniform implementation of several nonmonotonic reasoning formalisms including belief revision as defined in Delgrande and Schaub’s framework [15]. BReLS [46] is a framework in which revision, update, and merging of knowledge bases can be jointly expressed. Experiments complementing our comparisons are also presented.

6.1 Experiment Preliminaries

As expounded in Section 3.4.2, similarity exists between projected merge and belief revision in Delgrande and Schaub’s framework [15, 16, 17] upon which CBBC is based. Specifically, Theorem 6 in Section 3.4.2 yields that \((K_1 + K_2) = \nabla (\{K_1, \top\}, K_2, \emptyset)\). This leads naturally to a comparison of belief revision in both COBA 2.0 [14] and QUIP [19] to projected merge with respect to entailment-based integrity constraints in CBBC.

All the experiments included in this chapter were conducted on one machine with an Intel Pentium IV CPU of 2.80 GHz, 256 MB of RAM, and the Red Hat Enterprise Linux 3.0 (with compiler gcc version 3.2.3) operating system.

We considered input belief change scenarios each with an index \(i\) ranging from 1 to 20, inclusive. The belief change scenario indexed \(i\) is of the form \((\{K_1\}, \{\alpha\}, \emptyset)\), where \(K_1 = \bigwedge_{n=1}^{i} (p_n \land r_n)\) and \(\alpha = \bigwedge_{n=1}^{i} (\neg p_n \lor \neg r_n)\) involving propositional atoms \(p_1, p_2, ..., p_i\) and \(r_1, r_2, ..., r_i\). These belief change scenarios, each with an exponential
CHAPTER 6. RELATED WORK

(i.e. exponential in the number of clauses in $\alpha$) number of determining $EQ$ sets, were selected to facilitate run-time comparisons among CBBC, COBA 2.0, and QUIP.

The command-line version of COBA 2.0 can be downloaded from http://www.cs.sfu.ca/~cl/software/COBA/coba2.html, and CBBC from http://www.cs.sfu.ca/~cl/software/CBBC/CBBC.zip. The implementation of BReLS is available via a CGI interface at http://www.dis.uniroma1.it/~liberato/brels/. Lastly, we are indebted to Stefan Woltran et al. (authors of QUIP) for making the QUIP code and user guide available to us for testing and comparison purposes.

6.2 COBA 2.0

COBA 2.0 [14] is a Java implementation of Delgrande and Schaub’s framework [15] focusing only on belief revision and contraction incorporating integrity constraints. Another distinguishing characteristic of COBA 2.0 is that it uses the SAT solver Berkmin in the SAT4J library [6] for all its satisfiability checks.

There are similarities between COBA 2.0 and CBBC, both founded on Delgrande and Schaub’s framework. First, COBA 2.0’s algorithm for computing a belief change extension as shown in [14] consists of four main stages: checking preliminary consistency of the input formulas, expressing the knowledge base in a disjoint language, non-deterministically finding a determining $EQ$ set, and finally re-expressing the knowledge base in its original language according to $EQ$. Consequently, CBBC’s algorithms $ComputeSymmetricBCE$ and $ComputeProjectedBCE$ from Section 5.1 bear much resemblance to this algorithm. Second, COBA 2.0 also uses $maxConsEQSets$ and $minInconsEQSets$ for the same purposes and in the same way that CBBC uses them as detailed in Section 5.2.

Table 6.1 presents COBA 2.0’s skeptical belief revision experimental results. Each row indexed $i$ comprises the experimental results of the input belief change scenario indexed $i$. Column $i$ lists the index; Column $\#EQs$ the number of determining $EQ$ sets; Column $\#Poss EQs$ the total number of possible equivalence sets; Column $\#Sat$ the number of satisfiability checks involved in finding all the determining $EQ$ sets; Column $\#Sup$ the number of satisfiability checks avoided thanks to $minInconsEQSets$; Column $\#Sub$ the number of satisfiability checks avoided thanks to $maxConsEQSets$. Column $SAT$ indicates the time in seconds spent on satisfiability checks involved in the search for all determining $EQ$ sets, while $TOTAL$ shows COBA 2.0’s total run-time (including $SAT$ time) in seconds.
CHAPTER 6. RELATED WORK

Table 6.1: COBA 2.0 Belief Revision Experimental Results

<table>
<thead>
<tr>
<th>i</th>
<th>#EQs</th>
<th>#Poss EQs</th>
<th>#Sat</th>
<th>#Sup</th>
<th>#Sub</th>
<th>COBA 2.0 SAT TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.099 0.102</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>16</td>
<td>11</td>
<td>0</td>
<td>3</td>
<td>0.116 0.22</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>$2^5$</td>
<td>24</td>
<td>6</td>
<td>14</td>
<td>0.047 0.062</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>$2^8$</td>
<td>50</td>
<td>37</td>
<td>53</td>
<td>0.097 0.126</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>$2^{10}$</td>
<td>100</td>
<td>158</td>
<td>184</td>
<td>1.6   1.734</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>$2^{12}$</td>
<td>197</td>
<td>549</td>
<td>600</td>
<td>4.231 4.429</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>$2^{14}$</td>
<td>537</td>
<td>1742</td>
<td>1775</td>
<td>12.469 12.762</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>$2^{16}$</td>
<td>1241</td>
<td>5517</td>
<td>5508</td>
<td>33.812 34.591</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>$2^{18}$</td>
<td>3108</td>
<td>17701</td>
<td>16845</td>
<td>97.162 101.28</td>
</tr>
<tr>
<td>10</td>
<td>$2^{10}$</td>
<td>$2^{20}$</td>
<td>5515</td>
<td>37293</td>
<td>53895</td>
<td>193.162 210.857</td>
</tr>
<tr>
<td>11</td>
<td>$2^{11}$</td>
<td>$2^{22}$</td>
<td>12307</td>
<td>114045</td>
<td>165309</td>
<td>459.714 562.283</td>
</tr>
<tr>
<td>12</td>
<td>$2^{12}$</td>
<td>$2^{24}$</td>
<td>52955</td>
<td>346476</td>
<td>479136</td>
<td>1990.507 2620.188</td>
</tr>
</tbody>
</table>

Table 6.1: COBA 2.0 Belief Revision Experimental Results

Table 6.2 presents our experimental results of skeptical projected merge using BuDDy [47] without dynamic variable reordering (DVO) via JavaBDD [63]. Table 6.2 uses exactly the same format as Table 6.1.

For each belief change scenario indexed i, Tables 6.1 and 6.2 share the same #EQs and #Poss EQs; however, Tables 6.1 and 6.2 differ in #Sat, #Sup, and #Sub in most rows. Although COBA 2.0 and CBBC generate the same ordered list L (see Section 5.2) of bit-strings (representing possible equivalence sets to consider) for each belief change scenario indexed i, the respective actual equivalence sets in COBA 2.0 and CBBC corresponding to the same bit-string are likely to be different. In CBBC, the order of the actual atomic equivalences (each represented by a bit in a specific bit position) is determined by one traversal through a Hashtable of common atoms (see CASet commonAtoms in Section 5.2); this order is similarly designated by one iteration over a HashSet of common atoms in COBA 2.0. However, both HashSet and Hashtable lack a predictable iteration order; that is, each iteration over the same HashSet or Hashtable may impose a different order, with no guarantee of reflecting the order in which the common atoms were inserted. This is, nevertheless, reasonable as we have no special knowledge of any optimal order of atomic equivalences to examine for any given belief change scenario.

Although SAT and TOTAL in both Tables 6.1 and 6.2 exhibit an exponential growth, CBBC has significantly lower SAT and TOTAL time than does COBA 2.0. See the four rightmost columns in Table 6.3 for a juxtaposition of CBBC’s run-time and COBA 2.0’s run-time. The substantially higher TOTAL time of COBA 2.0 is attributed to the requisite
conjunctive normal form (CNF) transformation of any input formula to its SAT solver using de Morgan’s and distributive laws [14]. As an example of the exponential blow-up in formula size associated with CNF transformation, any CNF formula logically equivalent to 

\[ \bigvee_{i=1}^{n} (p_i \land r_i) \]

has at least \(2^n\) clauses [11]. In contrast, with a BDD library via JavaBDD [63], CBBC directly constructs BDDs from input formulas for satisfiability checks without any structural transformation of the formulas. The bottleneck due to the required CNF transformation of formulas for satisfiability checks by SAT solvers is, in fact, our primary motivation for deciding satisfiability with a BDD library in CBBC.

### 6.3 QUIP

In [19, 20], Egly, Eiter, Tompits, and Woltran propose the automated inference tool QUIP for solving such propositional nonmonotonic reasoning approaches as logic-based abduction, autoepistemic logic, circumscription, default logic, and disjunctive stable model semantics. QUIP first translates a given nonmonotonic problem of one of the aforementioned types into quantified Boolean formulas (QBF) using polynomial-time reductions given in [20], and then computes the solution by using the publicly available bddlib [48] as its QBF-evaluator.

Additionally, [18] axiomatizes belief revision as defined in Delgrande and Schaub’s framework [15] by means of QBFs. The polynomial-time constructible reduction of belief revision into QBFs lends itself naturally to QUIP’s framework [18]. In particular, [18] formalizes
belief revision in QUIP via the following theorem, assuming that each of $K$, $R$, and $C$ in a given belief change scenario $B = (K, R, C)$ is finitely representable as a consistent conjunction of its constituent formulas.

**Theorem 13** [18] Let $B = (K, R, C)$ be a belief change scenario in $\mathcal{L}_p$, and $V = \{v_1, v_2, \ldots\}$ be the set of all variables appearing in $B$. Also let $V_{eq}$ denote the set of new variables $\{v_{eq} \mid v_i \in V\}$, and let $\mathcal{M}[B]$ abbreviate $K' \land \land 1$ to $|V|$, $v_{eq} \in V_{eq}, v_i \in V, v_i' \in V' (v_{eq} \supset (v_i \equiv v_i')) \supset R$. Consider the following QBF:

$$F_{ext}[B] = \exists V \forall V' (\mathcal{M}[B] \land C) \land \land \land \land (\neg v_{eq} \supset (\neg \exists V \forall V'((v_i \equiv v_i') \land \mathcal{M}[B] \land R))).$$

Then $B$ has a consistent belief change extension iff $F_{ext}[B]$ is satisfiable. Moreover, the satisfying truth assignments of the free variables $V_{eq}$ of $F_{ext}[B]$ are in a one-to-one correspondence to the consistent belief change extensions of $B$.

Table 6.3 shows the total run-time (in seconds) of QUIP juxtaposed with the earlier runtime results of COBA 2.0 and CBBC from Tables 6.1 and 6.2. An NA result in Table 6.3 indicates that the particular experiment with which the result is associated was either not completed or not carried out. Notably, COBA 2.0’s experiment on the belief change scenario indexed 13 was aborted after 7200 seconds, and so too was CBBC’s experiment on the input belief change scenario indexed 14.

For each successive input belief change scenario in Table 6.3, QUIP’s run-time increases by an approximate factor of two. However, QUIP’s run-time is still considerably lower than that of CBBC and that of COBA 2.0. The primary reason is that to perform belief revision on a given belief change scenario, QUIP passes the corresponding polynomial-sized QBF to its QBF-evaluator just once; in contrast, when identifying $EQ$ sets from among the exponentially many possible equivalence sets under consideration, CBBC and COBA 2.0 feed at most an exponential number of polynomial-sized formulas to their respective consistency checkers.

There are also two possible reasons for QUIP’s relatively shorter runtime that justify further empirical investigation in the future. First, QUIP’s implementation (excluding bddlib) consists of only 2000 lines of C code, and C may render QUIP faster and more memory-efficient than the Java implementations COBA 2.0 and CBBC. Second, the QBF evaluator bddlib, which accepts arbitrary input QBFs, may be inherently more efficient than the BDD solvers considered in CBBC via JavaBDD [63] and the SAT solver in COBA 2.0.
Lastly, we conclude our experimental comparisons with Table 6.4, which shows CBBC’s experimental results for the 12 input belief change scenarios indexed \( i = 1, 2, \ldots, 12 \) using BuDDy \[47\] with six different dynamic variable ordering (DVO) techniques via JavaBDD. Columns titled \textit{SAT} and \textit{TOTAL} are defined in the same way for each respective DVO technique, and in the same way as they are in Table 6.2 for CBBC employing BuDDy without DVO via JavaBDD. On the 12 input belief change scenarios considered, using BuDDy with different DVO techniques or without DVO via JavaBDD yields closely comparable run-time.

As described in Section 5.6, DVO is a size-minimizing technique whereby a BDD library automatically determines appropriate points at which to pause BDD construction, adjust the variable ordering, and then continue processing \[56\]. The DVO techniques tested in Table 6.4 are briefly summarized here. Win2, using a sliding window of size two, considers swapping two adjacent variables and keeps the swap (and thus the new variable order) if the swap does not result in more BDD nodes; Win2 then repeats this for all pairs of adjacent variables \[56\]. Win3, except for its use of a sliding window of size three, proceeds in the same way as Win2. Sift focuses on finding the optimum position for a variable under the
assumption that all other variables stay fixed [56]. Each technique with suffix Ite in its name (e.g. Win2Ite) is the same as its non-iterative version (e.g. Win2) except that the process is repeated until no further progress is achieved [47]. Readers interested in further details are referred to [56, 47].

### 6.4 BReLS

In [46], Liberatore and Schaerf present BReLS, a framework in which belief revision, update, and merging of knowledge bases can be jointly expressed. Belief revision and merging are as defined earlier in Chapter 2; belief update is given in [46] as the integration of two fully reliable pieces of information each referring to a different time point (indicating that some changes may have happened between the two time points).

To obtain a comparison with CBBC, we focus on BReLS’s belief revision and merging operators, centered on the notion of the preference of static models in a distance-based semantics. A static model, representing the state of the world at a given time point, is a truth assignment to a set of propositional atoms. Letting \( M \) be a static model and \( \phi \) be a propositional formula, [46] defines the distance between \( M \) and \( \phi \) as \( \text{dist}(M, \phi) = \min\{\text{hamm}(M, N) \mid N \in \text{mod}(\phi)\}, \subseteq \), where \( \text{hamm}(M, N) \) is the Hamming distance between models \( M \) and \( N \) (i.e. the number of atoms on whose truth values they disagree), and \( \text{mod}(\phi) \) is the set of all models of \( \phi \). Also, source\((i) \) with \( i \in \mathbb{Z}^+ \) indicates the degree \( i \) of reliability of a propositional formula; the higher \( i \) is, the more
reliable the associated source is.

[46] provides the following definitions regarding the preference of static models. Given a knowledge base composed of propositional formulas \( K_1 \) and \( K_2 \) of the same degree of reliability, the preference of a static model \( M \) is given by \( \text{pref}(M) = \text{dist}(M, K_1) + \text{dist}(M, K_2) \). Given a knowledge base comprised of propositional formulas \( K_1, K_2, ..., K_n \) of differing degrees of reliability, the preference of a static model \( M \) is an array whose \( i \)th element is defined as \( \text{pref}(M)[i] = \sum_{\phi \in \{K_1, K_2, ..., K_n\}} \text{has reliability } \phi \cdot \text{dist}(M, \phi) \). Additionally, the ordering between two static models \( M_1 \) and \( M_2 \) is as follows, where \( P \) is the largest degree of reliability of formulas in a given knowledge base:

\[
M_1 \prec M_2 \text{ iff } \\
\exists i (1 \leq i \leq P) \text{ such that } (\text{pref}(M_1)[i] < \text{pref}(M_2)[i]) \\
\text{and } \forall j (i < j \leq P) \ (\text{pref}(M_1)[j] = \text{pref}(M_2)[j]).
\]

The ordering guarantees that if \( M_1 \) has a shorter distance than \( M_2 \) does to the formulas with the maximal reliability, then \( M_1 \) is preferred. In other words, a model’s distance to formulas of reliability \( P - 1 \) matters only when the competing models under consideration have the same distance to formulas of reliability \( P \).

Belief revision is expressed in BReLS as follows, where \( K \) is a knowledge base represented by a conjunction of its constituent formulas and \( \alpha \) is a propositional formula for revision:

\[
KB = \left\{ \text{source}(1) : K, \text{source}(2) : \alpha \right\}
\]

While both \( K \) and \( \alpha \) hold at the same time point, we place more credence in \( \alpha \) given its higher degree of reliability. The result of revising \( K \) by \( \alpha \), denoted by \( \text{mod}(K \star \alpha) \), is the set of static models of \( KB \). The revision operator \( \star \) is alleged in [46] to satisfy all the AGM revision postulates (R1)-(R8) enumerated in Section 2.1. Additionally, [15] notes the correspondence of \( \star \) to Dalal’s approach [13].

Let \( K_1 \) and \( K_2 \) be two knowledge bases each represented by a conjunction of its constituent formulas. Under the assumption that both \( K_1 \) and \( K_2 \) have the same degree of reliability at the same time point, the merging of \( K_1 \) and \( K_2 \) is expressed in BReLS as follows:

\[
KB = \left\{ \text{source}(1) : K_1, \text{source}(1) : K_2 \right\}
\]
The result of merging $K_1$ and $K_2$, denoted by $mod(K_1 \triangleleft_{LS} K_2)$, is the set of static models of $KB$. The merging operator $\triangleleft_{LS}$ is claimed in [46] to satisfy the basic arbitration postulates (LS1)-(LS6) listed in Section 2.2. However, one deficiency of $\triangleleft_{LS}$ is that it fails to obey the essential arbitration property (LS7). To enforce (LS7) as well as (LS1)-(LS6), [46] recasts the arbitration of $K_1$ and $K_2$ as follows:

$$KB = \begin{cases} 
  \text{source}(2) : & K_1 \lor K_2, \\
  \text{source}(1) : & K_1, \\
  \text{source}(1) : & K_2
\end{cases}$$

The result of the arbitration of $K_1$ and $K_2$, denoted by $mod(K_1 \triangledown_{LS} K_2)$, is the set of static models of $KB$.

The following example illustrates the connection between the preferences of static models and the resulting models of a belief revision.

$$KB = \begin{cases} 
  \text{source}(1) : & p \land q, \\
  \text{source}(2) : & \neg p \lor \neg q
\end{cases}$$

Given that only two distinct propositional atoms $p$ and $q$ appear in $KB$, there are four representative consistent static models to consider, namely, $M_1 = \{p, q\}$, $M_2 = \{p, \neg q\}$, $M_3 = \{q, \neg p\}$, and $M_4 = \{\neg p, \neg q\}$. $M_1$ is representative of all static models satisfying $p$ and $q$; $M_2$ all static models satisfying $p$ and $\neg q$; $M_3$ all static models satisfying $q$ and $\neg p$; $M_4$ all static models satisfying $\neg p$ and $\neg q$. We calculate each of their preferences separately as follows.

$$
\begin{align*}
\text{pref}(M_1)[1] &= \text{dist}(M_1, (p \land q)) = 0 \\
\text{pref}(M_1)[2] &= \text{dist}(M_1, (\neg p \lor \neg q)) = 1 \\
\text{pref}(M_2)[1] &= \text{dist}(M_2, (p \land q)) = 1 \\
\text{pref}(M_2)[2] &= \text{dist}(M_2, (\neg p \lor \neg q)) = 0 \\
\text{pref}(M_3)[1] &= \text{dist}(M_3, (p \land q)) = 1 \\
\text{pref}(M_3)[2] &= \text{dist}(M_3, (\neg p \lor \neg q)) = 0 \\
\text{pref}(M_4)[1] &= \text{dist}(M_4, (p \land q)) = 2 \\
\text{pref}(M_4)[2] &= \text{dist}(M_4, (\neg p \lor \neg q)) = 0
\end{align*}
$$

Their relative order is, therefore, that $M_2 \prec M_4 \prec M_1$ and $M_3 \prec M_4 \prec M_1$, with $M_2$ and $M_3$ being assigned the same level of preference. Consequently, the most preferred static models $M_2$ and $M_3$ with respect to $KB$ are the static models of $KB$. 
While both $\Delta_{LS}$ and $\nabla_{LS}$ can be extended to integrate more than two knowledge bases, the current C implementation of BReLS [46] can accommodate only up to eight different propositional atoms appearing in any given belief revision, merging, or arbitration problem. In addition to obtaining the models of a given problem, users can query any propositional formula to determine whether it is valid in the problem (i.e. whether it holds in all models of the problem). We omit detailed run-time experiments for BReLS because of its current inability to scale to larger input belief change scenarios, such as those with index $i > 4$ considered in Sections 6.2 and 6.3. Nevertheless, this elucidating review of BReLS is conducive to our understanding of distance-based approaches to belief change, which stand in contrast to CBBC's consistency-based approaches to belief change.
Chapter 7

Conclusion

We have presented a BDD-based system implementation, CBBC, of the consistency-based belief change framework proposed by Delgrande and Schaub [15, 16, 17]. The generality of this framework manifests itself in the unified manner in which it handles belief revision, multiple contractions, and knowledge base merging with respect to integrity constraints. Moreover, the belief revision, contraction, and merging operators defined in the framework have good formal properties as shown by their connections to the theoretical foundations in existing belief change literature.

CBBC concentrates on knowledge base merging with respect to integrity constraints, addressing two consistency-based approaches to merging in particular. In symmetric merge, the common information in the source knowledge bases is collected to produce a resulting belief set retaining as much as consistently possible of the contents of the source knowledge bases. In projected merge, source knowledge bases are projected onto a target knowledge base which we desire to augment; in the simplest case, the target knowledge base could consist solely of T.

CBBC, implemented in Java's object-oriented paradigm, is portable to all platforms with a Java Virtual Machine and can be downloaded from http://www.cs.sfu.ca/~cl/software/CBBC/CBBC.zip. Via the Java interface JavaBDD [63], CBBC employs one of four well-known BDD libraries (BuDDy [47], CUDD [59], CAL [53], and JDD [61]) as its consistency checker. As well, the finite and vocabulary-restricted representation of merging results, whose correctness is proved in this thesis, has been incorporated into CBBC.

Experimental results comparing CBBC with related systems COBA 2.0 [14] and QUIP [19] reveal an exponential growth in the run-time of all three systems. Furthermore, dynamic
variable ordering in the BDD solver BuDDy has seemingly no effect on CBBC’s run-time for the belief change scenarios used in our experiments. With respect to our belief revision and projected merge experimental results in Chapter 6, CBBC is substantially faster than COBA 2.0, whereas QUIP is considerably faster than CBBC. This indicates that there exists ample room for improvement on CBBC and leads to some proposed future directions of our work detailed below.

Our proposed future directions, along with their motivations, are manifold as enumerated below.

1. A current development in COBA 2.0 [14] is the replacement of the conjunctive normal form (CNF) translation of input formulas with structure preserving normal form (SPNF) translation [51]. Avoiding the exponential increase in the size of a translated formula associated with CNF translation, SPNF translation yields a satisfiable equivalent formula polynomial in the length of the input formula. We can explore substituting a SAT solver for our current BDD solver in CBBC and using the SPNF translation to convert input formulas to a form suitable for the SAT solver.

2. We can write our own Java interface to link CBBC to BDD libraries written in C. Although JavaBDD provides a uniform interface to BuDDy, CUDD, CAL, and JDD, its current version 1.0b2 has yet to implement the interface functions to dynamic variable ordering. As well, in order to minimize the sizes of constructed BDDs, we can investigate ways to achieve optimal variable ordering in the context of CBBC’s consistency-based belief change framework.

3. While QUIP is a viable tool for belief revision, CBBC has the additional, distinct ability to perform knowledge base merging incorporating integrity constraints. To emulate the relatively desirable run-time behavior of QUIP, we may re-implement CBBC in C and adopt bddlib [48] as our underlying consistency checker. Additionally, given the availability of many BDD libraries implemented in C, we can devise further tests to identify the optimal BDD libraries to use for consistency checking in CBBC.

4. We can present CBBC as an applet with an easily navigable user interface. This will obviate the current requirement that users install the Java Development Kit and compile the CBBC’s source code before running CBBC. Instead, without recompiling and rebuilding the code, users will be able to interact with the applet in a Web browser.
providing that they have installed the Java Virtual Machine on their computers and that they have configured their Web browser to enable the display of Java applets.

1., 2., and 3. are of the same priority, followed by 4., which can be pursued after a decidedly improved implementation has been obtained. Admittedly, 2. and 3. are likely to be disjoint endeavors since the former maintains a Java implementation, while the latter involves a complete re-write of CBBC as a C implementation. We are optimistic that proposed future directions 1.-3. will afford us more insight into the run-time behavior of our system, with 4. enhancing the usability and appeal of our system to people interested in belief change.

We summarize the notable contributions of this thesis. First, to the best of our knowledge, CBBC is the first and currently the only published consistency-based system for knowledge base merging with respect to integrity constraints. Second, for the sake of feasibility, we propose considerations for finite and vocabulary-restricted representation of knowledge base merging. Our formal proofs establish both their correctness and their logical equivalence to the original semantic definitions of symmetric and projected merge given in [16, 17]. Third, our adoption of a BDD solver in CBBC offers a refreshing and more efficient alternative to the conventional use of a SAT solver for consistency checking purposes, as evidenced by the relatively shorter run-time of CBBC than that of COBA 2.0 in Table 6.3 from Section 6.3.

Lastly, from our empirical evaluations of CBBC, COBA 2.0, and QUIP, we conclude that while much of the aforementioned proposed future work on CBBC is motivated by QUIP's commendable run-time behavior, directly extending QUIP to address symmetric merge and projected merge would be a promising avenue of research. As we have observed in Sections 6.3 and 3.5, not only does QUIP already provide an evidently efficient implementation of consistency-based belief revision, but QUIP also constructs polynomial-sized QBFs whose structures are faithful to the computational complexity of the EXT, CHOICE, and Skeptical problems of a given belief change scenario. In stark contrast, both the CNF transformation in COBA 2.0 and the BDD representation in CBBC are used solely for consistency checking purposes and impart no such structural insight. As a result of the generality of Delgrande and Schaub's consistency-based belief change framework, the proposed extension of QUIP would likely be a relatively easier, and thus more practicable, undertaking than future improvement on CBBC and COBA 2.0.
Appendix A

Proofs

We show the proofs of Theorems 9, 10, 11, and 12 in this appendix.

A.1 Proof of Theorem 9

Let \( B = (\{K_1, K_2\}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_p \), and \((EQ_i)_{i \in I}\) be the family of all sets of equivalences determining the symmetric belief change extensions of \( B \). Also, let \( \uparrow K_1 \uparrow_i \) and \( \uparrow K_2 \uparrow_i \) be as given by the substitutions (S1)-(S6) with respect to a given \( EQ_i \) as described in Section 4.1. We aim to show that \( \Delta (\{K_1, K_2\}, R, C) = \bigvee_{i \in I} (\uparrow K_1 \uparrow_i \lor \uparrow K_2 \uparrow_i) \land (\bigwedge R) \).

Let \( E_i = \{ \alpha \mid \{\alpha^1, \alpha^2\} \subseteq Cn(K_1^i \cup K_2^i \cup R^1 \cup R^2 \cup EQ_i) \} \) be a consistent symmetric belief change extension of \( B \) with the determining set \( EQ_i \) of equivalences. We wish to show that \( E_i \equiv (\uparrow K_1 \uparrow_i \lor \uparrow K_2 \uparrow_i) \land (\bigwedge R) \).

Supposing \( \beta \) is an arbitrary propositional formula in \( E_i \), we obtain the following result.

\[ \beta \in E_i \]

\[ \iff \text{(by Definition 7) } Cn(K_1^i \cup K_2^i \cup R^1 \cup R^2 \cup EQ_i) \vdash \beta^1 \land \beta^2 \]

\[ \iff \text{(by definition of } Cn(\cdot)) \]

\[ K_1^i \cup K_2^i \cup R^1 \cup R^2 \cup EQ_i \vdash \beta^1 \text{ and } \]

\[ K_1^i \cup K_2^i \cup R^1 \cup R^2 \cup EQ_i \vdash \beta^2. \] \hspace{1cm} \text{(A.1)}

We next show that (A.1) iff

\[ \uparrow K_1 \uparrow_i \land (\bigwedge R) \vdash \beta, \] \hspace{1cm} \text{(A.3)}
As an intermediate step, we show that (A.1) iff

\[ \uparrow K_2 \uparrow \land (\bigwedge R) \vdash \beta. \]  

(A.4)

We show each part of the iff in turn.

(Only-if Part:) We show that any model \( M \) of \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \) is also a model of \( K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1 \). From substitution of logical equivalents (via \( EQ_i \)), we have that \( K^2 \cup EQ_i \vdash K_2 \sigma_{2,1}^{EQ_i} \) and that \( R^2 \cup EQ_i \vdash R_1 \) as \( EQ_i \) determines a consistent symmetric belief change extension of the consistent multi belief change scenario \( B \). Thus, we get that \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \vdash (K_1 \lor K_2 \sigma_{2,1}^{EQ_i} \cup R_1) \) and so \( M \) is also a model of \( K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1 \). Consequently, (A.1) implies (A.5).

(If Part:) We prove the contrapositive that \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \nvdash \beta \) implies \( K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1 \nvdash \beta \). Suppose that \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \nvdash \beta \); that is, there exists some model \( M \) which satisfies \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \) but falsifies \( \beta \). From substitution of logical equivalents (via \( EQ_i \)), we have that \( K^2 \cup EQ_i \vdash K_2 \sigma_{2,1}^{EQ_i} \) and that \( R^2 \cup EQ_i \vdash R_1 \) as \( EQ_i \) determines a consistent symmetric belief change extension of the consistent multi belief change scenario \( B \). Thus, we get that \( K_1 \cup K_2 \cup R_1 \cup R_2 \cup EQ_i \vdash (K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1) \), and so \( M \), which falsifies \( \beta \), also satisfies \( K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1 \). The existence of such a model \( M \) attests to \( K_1 \cup K_2 \sigma_{2,1}^{EQ_i} \cup R_1 \nvdash \beta \). Consequently, (A.5) implies (A.1).

Analogous to (A.1) iff (A.5), we obtain that (A.2) iff

\[ K_1^{\sigma_{1,2}^{EQ_i}} \cup K_2^{R_2} \vdash \beta^2. \]  

(A.6)

Furthermore, via mere manipulation of syntax by dropping superscripts in (A.5) and in (A.6), we get that (A.5) iff

\[ nosup(K_1 \cup K_2^{EQ_i}) \cup nosup(R_1) \vdash \beta, \]  

(A.7)

and that (A.6) iff

\[ nosup(K_1^{\sigma_{1,2}^{EQ_i}} \cup K_2^{R_2}) \cup nosup(R^2) \vdash \beta. \]  

(A.8)
We then observe that in accordance with the substitutions (S1)-(S6) with respect to \( EQ_i \) as described in Section 4.1, (A.7) iff (A.3), just as (A.8) iff (A.4). Moreover, by the basic semantic definition of propositional logic, we get that [both(A.3) and (A.4)] iff \\
\[ (\triangleright K_1 \downarrow_i \lor \triangleright K_2 \downarrow_i) \land (\land R) \vdash \beta. \]

By the transitivity of iff, we have established that \( E_i \equiv (\triangleright K_1 \downarrow_i \lor \triangleright K_2 \downarrow_i) \land (\land R) \) corresponding to some arbitrarily chosen \( EQ_i \) of \( B = (\{K_1, K_2\}, R, C) \). Therefore, we conclude that \( \Delta (\{K_1, K_2\}, R, C) = \bigvee_{i \in I} (\triangleright K_1 \downarrow_i \lor \triangleright K_2 \downarrow_i) \land (\land R). \)

### A.2 Proof of Theorem 10

Given the similarity (noted in Section 3.4.2) between belief revision and projected merge in Delgrande and Schaub's framework [15, 16, 17], this proof is modeled after that of the corresponding Theorem 5.1 in [15].

Let \( B = (\{K_1, K_2\}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_P \), and \( \{EQ_i\}_{i \in I} \) be the family of all sets of equivalences determining the projected belief change extensions of \( B \). Also, let \( \downarrow KB \downarrow_i \) denote \((\land K_1^\uparrow_i) \land (\land K_2^\uparrow_i) \) having undergone the substitutions (P1) and (P2) with respect to a given \( EQ_i \) as described in Section 4.1. We aim to show that \\
\[ \forall (\{K_1, K_2\}, R, C) = \bigvee_{i \in I} (\downarrow KB \downarrow_i) \land (\land R) \] by making use of Lemmas 1 and 2.

**Lemma 1** If \( EQ_i \) determines a consistent projected belief change extension of \((K_j)_{j \in J}, R, C \) in \( \mathcal{L}_P \), then for \( (p^j \equiv p) \in \overline{EQ_i} \), there is some \( \phi \in C \cup \{\bot\} \) such that \( \bigcup_{j \in J} K_j^\uparrow \cup R \cup \{\neg \phi\} \cup EQ_i \vdash (p^j \equiv p) \), where \( \overline{EQ_i} = \{p^j \equiv p \mid p \in P \text{ and } j \in J\} \setminus EQ_i \).

**Proof 1** Let \( EQ_i \subseteq \{q^j \equiv q \mid j \in J \text{ and } q \in P \} \) be a maximal set of equivalences determining some consistent projected belief change extension of \((K_j)_{j \in J}, R, C \). Assume that \( \overline{EQ_i} \neq \emptyset \), and let \( (p^j \equiv p) \in \overline{EQ_i} \). By the maximality of \( EQ_i \), we have that \( \bigcup_{j \in J} K_j^\uparrow \cup R \cup \{\neg \phi\} \cup EQ_i \cup \{p^j \equiv p\} \vdash \bot \) for some \( \phi \in C \cup \{\bot\} \). That is, \( \bigcup_{j \in J} K_j^\uparrow \cup R \cup \{\neg \phi\} \cup EQ_i \vdash \neg(p^j \equiv p) \), or equivalently \( \bigcup_{j \in J} K_j^\uparrow \cup R \cup \{\neg \phi\} \cup EQ_i \vdash (\neg p^j \equiv p) \).

Note that the use of \( J \) in Lemma 1 is more general and certainly encompasses the case of interest, namely \( J = \{1, 2\} \), in our proof of Theorem 10.

**Lemma 2** Let \( E_i \) be a consistent projected belief change extension of \(\{K_1, K_2\}, R, C \) with the determining set \( EQ_i \) of equivalences. Then, we have \\
\[ \vdash (\bigwedge_{p^j \equiv p \in EQ_i, j \{1,2\}} (p \equiv p^j) \land (p \equiv p^j)) \supset ((\bigwedge K_1^\uparrow \land \bigwedge K_2^\uparrow) \equiv \downarrow KB \downarrow_i). \]
Proof 2 Let $M$ be a model of $\bigwedge_{p=p^j \in EQ_i, j \in \{1,2\}} (p \equiv p^j) \wedge \bigwedge_{p=p^j \not\in EQ_i, j \in \{1,2\}} (p \equiv \neg p^j)$. $\Downarrow KB \downarrow_i$ is the same as $(\bigwedge K_1) \land (\bigwedge K_2)$ except that for every $p^j \in P_{EQ_i}$ with $j \in \{1,2\}$, where $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ mentions $p^j$, $\Downarrow KB \downarrow_i$ has $\neg p$, and that for every $p^j \in P_{EQ_i}$ with $j \in \{1,2\}$, where $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ mentions $p^j$, $\Downarrow KB \downarrow_i$ has $p$.

1. For $p^j \in P_{EQ_i}$, we have that $M$ assigns the same truth value to $p^j$ in $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ as to $p$ in $(\bigwedge K_1) \land (\bigwedge K_2)$. This means that $M$ assigns the same truth value to $p^j$ in $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ as to $p$ in $\Downarrow KB \downarrow_i$.

2. For $p^j \in P_{EQ_i}$, we have that $M$ assigns the opposite truth value to $p^j$ in $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ than it does to $p$ in $(\bigwedge K_1) \land (\bigwedge K_2)$. This means that $M$ assigns the same truth value to $p^j$ in $(\bigwedge K_1^j) \land (\bigwedge K_2^j)$ as to $\neg p$ in $\Downarrow KB \downarrow_i$.

Hence, $M$ is a model of $(\bigwedge K_1^j \land \bigwedge K_2^j)$ iff $M$ is a model of $\Downarrow KB \downarrow_i$, concluding our proof of Lemma 2. 

We now need to show that for $E_i = Cn(K_1^i \cup K_2^i \cup R \cup EQ_i) \cap L_p$,

1. $Cn(K_1^i \cup K_2^i \cup R \cup EQ_i) \cap L_p \vdash KB \downarrow_i \land (\land R)$, and

2. $\{\Downarrow KB \downarrow_i \land (\land R)\} \vdash \phi$ for every $\phi \in Cn(K_1^i \cup K_2^i \cup R \cup EQ_i) \cap L_p$.

We prove each part in turn.

1. From Lemma 2, we have

$$\vdash (\bigwedge_{p=p^j \in EQ_i, j \in \{1,2\}} (p \equiv p^j) \wedge \bigwedge_{p=p^j \not\in EQ_i, j \in \{1,2\}} (p \equiv \neg p^j)) \supset ((\bigwedge K_1^j \land \bigwedge K_2^j) \equiv \Downarrow KB \downarrow_i).$$

Hence,

$$\{K_1^i\} \cup \{K_2^i\} \cup EQ_i \cup EQ_i \vdash \Downarrow KB \downarrow_i \tag{A.9}$$

Since $K_1^i \cup K_2^i \cup R \cup EQ_i \vdash p \equiv \neg p^j$ for every $(p \equiv p^j) \in EQ_i$ by Lemma 1, we obtain from (A.9) that $K_1^i \cup K_2^i \cup R \cup EQ_i \vdash \Downarrow KB \downarrow_i \land (\land R)$. By the definition of $Cn(\cdot)$, this means that $\Downarrow KB \downarrow_i \land (\land R) \in Cn(K_1^i \cup K_2^i \cup R \cup EQ_i)$. Since also $\Downarrow KB \downarrow_i \land (\land R) \in L_p$, we get $\Downarrow KB \downarrow_i \land (\land R) \in Cn(K_1^i \cup K_2^i \cup R \cup EQ_i) \cap L_p$. Therefore, $Cn(K_1^i \cup K_2^i \cup R \cup EQ_i) \cap L_p \vdash \Downarrow KB \downarrow_i \land (\land R)$.
2. Assume that $\phi \in Cn(K_1 \cup K_2 \cup R \cup EQ_1) \cap \mathcal{L}_p$. Then $\phi \in \mathcal{L}_p$ and $K_1 \cup K_2 \cup R \cup EQ_1 \models \phi$. From monotonicity of classical logic, it follows that $K_1 \cup K_2 \cup R \cup EQ_1 \cup \mathcal{L}_p \models \phi$. Lemma 2, along with (A.9), yields that $\{\downarrow KB \downarrow_i \} \cup R \models \phi$, or equivalently $\{\downarrow KB \downarrow_i \wedge (\wedge R)\} \models \phi$ as required.

Having shown that $E_i \equiv \{\downarrow KB \downarrow_i \wedge (\wedge R)\}$ corresponding to some arbitrarily chosen $EQ_i$ of $B = \{(K_1, K_2), R, C\}$, we conclude that $\forall (\{K_1, K_2\}, R, C) = \bigvee_{i \in I} (\downarrow KB \downarrow_i) \wedge (\wedge R)$.

A.3 Proof of Theorem 11

Let $B = \{(K_1, K_2), R, C\}$ be a multi belief change scenario in $\mathcal{L}_p$. We aim to show that $\Delta (\{K_1, K_2\}, R, C) \equiv \Delta \text{[KCA]}(\{K_1, K_2\}, R, C)$ by making use of Lemmas 3 and 4.

Lemma 3 If $EQ$ determines a consistent symmetric belief change extension of $((K_j)_{j \in J}, R, C)$ in $\mathcal{L}_p$, then for $(p^d \equiv p^e) \in \overline{EQ}$, there is some $\phi \in C^J \cup \{\bot\}$ such that $\bigcup_{j \in J} K_j^I \cup R^I \cup \{\neg \phi\} \cup EQ \models (\neg p^d \equiv p^e)$, where $\overline{EQ} = \{p^d \equiv p^e \mid p \in \mathcal{P}, d \neq e, \text{ and } d, e \in J\} \setminus EQ$.

Proof 3 Let $EQ \subseteq \{q^d \equiv q^e \mid d, e \in J, d \neq e, \text{ and } q \in \mathcal{P}\}$ be a maximal set of equivalences determining some consistent symmetric belief change extension of $((K_j)_{j \in J}, R, C)$. Assume that $\overline{EQ} \neq \emptyset$, and let $(p^d \equiv p^e) \in \overline{EQ}$. By the maximality of $EQ$, we have that $\bigcup_{j \in J} K_j^I \cup R^I \cup \{\neg \phi\} \cup EQ \models (\neg p^d \equiv p^e)$, where $\overline{EQ} = \{p^d \equiv p^e \mid p \in \mathcal{P}, d \neq e, \text{ and } d, e \in J\} \setminus EQ$.

Note that the use of $J$ in Lemma 3 above and Lemma 4 below is more general and certainly encompasses the case of interest, namely $J = \{1, 2\}$, in our proof of Theorem 11.

Lemma 4 Let $EQ$ be any maximal set of equivalences determining a consistent symmetric belief change extension of $((K_j)_{j \in J}, R, C)$ in $\mathcal{L}_p$. Then $\{p^i \equiv p^j \mid i, j \in J, i \neq j, \text{ and } p \in (\mathcal{P}(K_i) \setminus \text{KCA}) \cup \{p^i \equiv p^j \mid i, j \in J, i \neq j, \text{ and } p \in (\mathcal{P}(R \cup C) \setminus \text{KCA})\} \subseteq EQ$.

Proof 4 Assume otherwise (to derive a contradiction). Thus, there is a symmetric belief change extension of $B$ whose corresponding $EQ$ is such that:

1. $\exists i, j \in J, i \neq j, p \in \mathcal{P}$ where $p \in (\mathcal{P}(K_i) \setminus \text{KCA})$ and $(p^i \equiv p^j) \notin EQ$, or

2. $\exists i, j \in J, i \neq j, p \in \mathcal{P}$ where $p \in (\mathcal{P}(R \cup C) \setminus \text{KCA})$ and $(p^i \equiv p^j) \notin EQ$. 

- For the first case, we have from Lemma 3 that for some \( \phi \in C^J \cup \{ \bot \} \), \( \bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash (\neg p^i \equiv p^j) \), or equivalently \( \bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash (p^i \lor p^j) \lor (\neg p^i \lor \neg p^j) \).

Thus,

\[
\bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash p^i \lor p^j, \text{ and} \tag{A.10}
\]

\[
\bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash \neg p^i \lor \neg p^j. \tag{A.11}
\]

There are two further, possible scenarios to consider.

\((p \notin CA):\) Then \( p^j \notin \mathcal{P}(R^J \cup \{ \neg \phi \}) \) and \( p^i \notin \mathcal{P}(\bigcup_{j \in J} K_j^i) \). These, coupled with the assumption \((p^i \equiv p^j) \notin EQ\), imply that neither \( p^i \) nor \( \neg p^i \) is derivable from the left hand side of \( \vdash \) in (A.10) and (A.11).

Thus, from (A.10) we must have

\[
\bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash p^i. \tag{A.12}
\]

Analogously, from (A.11) we derive

\[
\bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash \neg p^i. \tag{A.13}
\]

However, (A.12) and (A.13) together yield that \( \bigcup_{j \in J} K_j^i \cup R^J \cup \{ \neg \phi \} \cup EQ \vdash \bot \), contradicting the fact that \( EQ \) determines a consistent symmetric belief change extension.

\((p \in CA):\) Since \( p \in (\mathcal{P}(K_i) \setminus KCA) \), neither \( p^i \) nor \( p^j \) appears in any \( K_d^i \) or in any \( R^d \) with \( d \in J \setminus \{i,j\} \). In addition, because \((p^i \equiv p^j) \notin EQ\) and \( p^j \notin \mathcal{P}(K_j^j) \) by assumption, we can simplify (A.10) and (A.11) and derive that

\[
K_i^i \cup R_i^i \cup R^j \cup \{ \neg \phi \} \vdash p^i \land \neg p^j \tag{A.14}
\]

or

\[
K_i^i \cup R_i^i \cup R^j \cup \{ \neg \phi \} \vdash \neg p^i \land p^j. \tag{A.15}
\]

\( \neg \phi \) is a single propositional formula \( \in C^J \cup \{ \bot \} \) and cannot be annotated with both superscripts \( i \) and \( j \); thus, there are three further, possible scenarios for (A.14) to consider.
1. Neither $i$ nor $j$ appears as a superscript in $\neg \phi$. Here, we can simplify (A.14) to $K_i^j \cup R_i^j \cup R_j^i \vdash p^i \land \neg p^j$. Then $\neg p^i \in R_i^j$, meaning $\neg p^i \in R_i^j$ as $R_i^j = \{ \psi^j \mid \psi^j \in R_i^j \}$; thus, $K_i^j \cup R_i^j \cup R_j^i \vdash \bot$.

2. $\{ \neg \phi \} \vdash p^i$. Then $\neg p^i \in R_i^j$, meaning $\neg p^i \in R_i^j$ as $R_i^j = \{ \psi^i \mid \psi^j \in R_i^j \}$; thus, $K_i^j \cup R_i^j \cup R_j^i \cup \{ \neg \phi \} \vdash \bot$.

3. $\{ \neg \phi \} \vdash \neg p^j$. Then there must exist some $\psi \in C^i$ such that $\{ \neg \psi \} \vdash \neg p^i$, which means that $K_i^j \cup R_i^j \cup R_j^i \cup \{ \neg \psi \} \vdash \bot$.

In any scenario for (A.14), we derive that for some arbitrary $\phi$ (or $\psi$) in $C^j \cup \{ \bot \}$, $\bigcup_{j \in J} K_j^j \cup R_j^j \cup \{ \neg \phi \} \cup EQ \vdash \bot$ (or $\bigcup_{j \in J} K_j^j \cup R_j^j \cup \{ \neg \psi \} \cup EQ \vdash \bot$), contradicting the fact that EQ determines a consistent symmetric belief change extension.

Analogously for (A.15), there are three further, possible scenarios to consider.

1. Neither $i$ nor $j$ appears as a superscript in $\neg \phi$. Here, we can simplify (A.15) to $K_i^j \cup R_i^j \cup R_j^i \vdash \neg p^j \land p^i$. Then $p^i \in R_j^i$, meaning $p^i \in R_j^i$ as $R_j^i = \{ \psi^j \mid \psi^j \in R_j^i \}$; thus, $K_i^j \cup R_i^j \cup R_j^i \vdash \bot$.

2. $\{ \neg \phi \} \vdash \neg p^j$. Then $p^j \in R_j^i$, meaning $p^j \in R_j^i$ as $R_j^i = \{ \psi^j \mid \psi^j \in R_j^i \}$; thus, $K_i^j \cup R_i^j \cup R_j^i \cup \{ \neg \phi \} \vdash \bot$.

3. $\{ \neg \phi \} \vdash p^j$. Then there must exist some $\psi \in C^i$ such that $\{ \neg \psi \} \vdash p^i$, which means that $K_i^j \cup R_i^j \cup R_j^i \cup \{ \neg \psi \} \vdash \bot$.

In any scenario for (A.15), we derive that for some arbitrary $\phi$ (or $\psi$) in $C^j \cup \{ \bot \}$, $\bigcup_{j \in J} K_j^j \cup R_j^j \cup \{ \neg \phi \} \cup EQ \vdash \bot$ (or $\bigcup_{j \in J} K_j^j \cup R_j^j \cup \{ \neg \psi \} \cup EQ \vdash \bot$), contradicting the fact that EQ determines a consistent symmetric belief change extension.

For the second case, we derive, as previously, (A.10) and (A.11). Since $p \in (P(R \cup C) \setminus KCA)$ by assumption, neither $p^i$ nor $p^j$ appears in $\bigcup_{j \in J} K_j^j$ or in any $R_d$ with $d \in J \setminus \{i, j\}$. In addition, because $(p^i \equiv p^j) \not\in EQ$ by assumption, we can simplify (A.10) and (A.11) and derive that

\[ R_i^j \cup R_j^i \cup \{ \neg \phi \} \vdash p^i \land \neg p^j \tag{A.16} \]

or

\[ R_i^j \cup R_j^i \cup \{ \neg \phi \} \vdash \neg p^i \land p^j \tag{A.17} \]
\( \neg \phi \) is a single propositional formula in \( C^d \cup \{ \bot \} \) and cannot be annotated with both superscripts \( i \) and \( j \); thus, there are three further, possible scenarios for (A.16) to consider.

1. Neither \( i \) nor \( j \) appears as a superscript in \( \neg \phi \). Here, we can simplify (A.16) to \( R^i \cup R^j \vdash p^i \land \neg p^j \). Then \( \neg p^j \in R^j \), meaning \( \neg p^i \in R^i \) as \( R^i = \{ \psi^i \mid \psi^j \in R^j \} \); thus, \( R^i \cup R^j \vdash \bot \).

2. \( \{ \neg \phi \} \vdash p^i \). Then \( \neg p^j \in R^j \), meaning \( \neg p^i \in R^i \) as \( R^i = \{ \psi^i \mid \psi^j \in R^j \} \); thus, \( R^i \cup R^j \cup \{ \neg \phi \} \vdash \bot \).

3. \( \{ \neg \phi \} \vdash \neg p^j \). Then there must exist some \( \psi \in C^i \) such that \( \{ \neg \psi \} \vdash \neg p^i \), which means that \( R^i \cup R^j \cup \{ \neg \psi \} \vdash \bot \).

In any scenario for (A.16), we derive that for some arbitrary \( \phi \) (or \( \psi \)) in \( C^d \cup \{ \bot \} \), \( \bigcup_{j \in J} K^i_j \cup R^j \cup \{ \neg \phi \} \cup EQ \vdash \bot \) (or \( \bigcup_{j \in J} K^i_j \cup R^j \cup \{ \neg \psi \} \cup EQ \vdash \bot \)), contradicting the fact that \( EQ \) determines a consistent symmetric belief change extension.

Analogously for (A.17), there are three further, possible scenarios to consider.

1. Neither \( i \) nor \( j \) appears as a superscript in \( \neg \phi \). Here, we can simplify (A.17) to \( R^i \cup R^j \vdash \neg p^i \land p^j \). Then \( p^j \in R^j \), meaning \( p^i \in R^i \) as \( R^i = \{ \psi^i \mid \psi^j \in R^j \} \); thus, \( R^i \cup R^j \vdash \bot \).

2. \( \{ \neg \phi \} \vdash \neg p^i \). Then \( p^j \in R^j \), meaning \( p^i \in R^i \) as \( R^i = \{ \psi^i \mid \psi^j \in R^j \} \); thus, \( R^i \cup R^j \cup \{ \neg \phi \} \vdash \bot \).

3. \( \{ \neg \phi \} \vdash p^j \). Then there must exist some \( \psi \in C^i \) such that \( \{ \neg \psi \} \vdash p^i \), which means that \( R^i \cup R^j \cup \{ \neg \psi \} \vdash \bot \).

In any scenario for (A.17), we derive that for some arbitrary \( \phi \) (or \( \psi \)) in \( C^d \cup \{ \bot \} \), \( \bigcup_{j \in J} K^i_j \cup R^j \cup \{ \neg \phi \} \cup EQ \vdash \bot \) (or \( \bigcup_{j \in J} K^i_j \cup R^j \cup \{ \neg \psi \} \cup EQ \vdash \bot \)), contradicting the fact that \( EQ \) determines a consistent symmetric belief change extension.

We now need to show that for any set \( EQ \) of equivalences determining some consistent symmetric belief change extension of \( B = (\{ K_1, K_2 \}, R, C) \), we have that

\[
\{ \alpha \mid \{ \alpha^1, \alpha^2 \} \subseteq \text{Can}(K^1_1 \cup K^2_2 \cup R^1 \cup R^2 \cup EQ) \} \equiv \\
\{ \alpha \mid \text{Can}(K^1_1|\text{KCA}) \cup K^2_2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup EQ|\text{KCA}) \models \alpha^1 \land \alpha^2 \} \\
\cup \{ \alpha \mid \alpha \in \mathcal{L} \cap \text{Can}(K^1_1|\text{KCA}) \cup K^2_2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup EQ|\text{KCA}) \},
\]
APPENDIX A. PROOFS

where $EQ[KCA] = EQ \setminus \{p^1 \equiv p^2 \mid p \notin KCA\}$.

We prove each part in turn.

(Only-if Part:) Suppose $\alpha$ is an arbitrary propositional formula in $\{\alpha \mid \{\alpha^1, \alpha^2\} \subseteq Cn(K_1^1 \cup K_2^1 \cup R_1 \cup R_2 \cup EQ\}\}$. We prove by structural induction on the length of $\alpha$ that $\{\alpha \mid Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA]) = \alpha^1 \land \alpha^2\} \cup \{\alpha \mid \alpha \in \mathcal{L}_p \cap Cn(\bigcup_{j \in J} K_j^1[KCA] \cup R_j^1[KCA] \cup EQ[KCA])\} \vdash \alpha$.

- (Base case: 1) $\alpha$ is an arbitrary propositional atom $q$.
  By our supposition, $\vdash q^1 \land q^2$. There are two subcases 1 a) and 1 b) to consider.
  1 a) $q \notin KCA$: Then it must be that $R \vdash q$, $K_1 \vdash q$, or $K_2 \vdash q$ with the restriction that the last two $\vdash$ expressions cannot both hold as $q \notin KCA$. If $q$ appears in $K_1$, $K_2$, or $R$, it still appears un-superscripted in $K_1^1[KCA]$, $K_2^1[KCA]$, or ($R_1[KCA]$ and $R_2[KCA]$), respectively. Therefore, $\vdash q \in \{\alpha \mid \alpha \in \mathcal{L}_p \cap Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA])\}$. 1 b) $q \in KCA$: Then it must be that $K_1 \vdash q^1$, $K_2 \vdash q^2$, and $(q^1 \equiv q^2) \in EQ$ by Lemma 3 and the assumption that $B$ is a consistent belief change scenario. Since $EQ[KCA] = EQ \setminus \{p^1 \equiv p^2 \mid p \notin KCA\}$, $(q^1 \equiv q^2) \in EQ[KCA]$. Furthermore, we have that $K_1^1[KCA] \vdash q^1$ and $K_2^1[KCA] \vdash q^2$ as $q \in KCA$ appears superscripted in $K_1^1[KCA]$ and in $K_2^1[KCA]$. Therefore, $\vdash q \in \{\alpha \mid \alpha \in \mathcal{L}_p \cap Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA])\} \vdash q^1 \land q^2$, which, in propositional logic, holds iff $\vdash q \in \{\alpha \mid \alpha \in \mathcal{L}_p \cap Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA])\} \vdash \alpha^1 \land \alpha^2$.

- (Inductive Hypothesis) For any arbitrary propositional formula $\beta \in \mathcal{L}_p$ with at most $n$ logical operators, $\beta \in \{\alpha \mid \{\alpha^1, \alpha^2\} \subseteq Cn(K_1^1 \cup K_2^1 \cup R_1 \cup R_2 \cup EQ\}\} \Rightarrow \{\alpha \mid Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA]) = \alpha^1 \land \alpha^2\} \cup \{\alpha \mid \alpha \in \mathcal{L}_p \cap Cn(K_1^1[KCA] \cup K_2^1[KCA] \cup R_1[KCA] \cup R_2[KCA] \cup EQ[KCA])\} \vdash \beta$.
  By the basic semantic definition of propositional logic, any non-atomic propositional formula can be constructed using only logical operators $\neg$ and/or $\land$. Accordingly, there are two inductive cases 2 and 3 to consider.
(Inductive case: 2) \( \alpha \) is of the form \( \neg(q) \), where \( q \) is an arbitrary propositional formula with \( n \) logical operators.

By our supposition, \( Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \vdash \neg(q)^1 \land \neg(q)^2 \). By the basic semantic definition of propositional logic, any non-atomic propositional formula can be constructed using only logical operators \( \neg \) and/or \( \lor \). Accordingly, there are two subcases 2 a) and 2 b) to consider.

2 a) \( \neg(q) \) is of the form \( \neg(\neg(s)) \) where \( s \) is a propositional formula with \( (n - 1) \) logical operators: Note that \( \vdash s \equiv \neg(\neg(s)) \), so by the definition of \( Cn(\cdot) \), we get that \( Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \vdash s^1 \land s^2 \). Since \( s \) has only \( (n - 1) \) logical operators, applying the inductive hypothesis, we obtain that

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash s,
\]

which holds iff

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash \neg(\neg(s))
\]
as desired, where \( \neg(\neg(s)) =_{syn} \neg(q) \).

2 b) \( \neg(q) \) is of the form \( \neg(s \lor t) \) where \( s \) and \( t \) are propositional formulas with \( \lceil \frac{n-1}{2} \rceil \) and \( \lceil \frac{n-1}{2} \rceil \) logical operators, respectively: Note that \( \vdash \neg(s \lor t) \equiv \neg s \land \neg t \), so by the definition of \( Cn(\cdot) \), we get that

\[
Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \vdash \neg s \land \neg t,
\]

and

\[
Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \vdash \neg t \land \neg t.
\]  

Applying the inductive hypothesis to (A.18), we get that

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash \neg s.
\]

Also, applying the inductive hypothesis to (A.19), we get that

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash \neg t.
\]

Consequently, we obtain that

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash \neg s \land \neg t,
\]

which holds iff

\[
\{ \alpha \mid Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \models \alpha^1 \land \alpha^2 \} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap Cn(K_1^1[KCA] \cup K_2^2[KCA] \cup R^1[KCA] \cup R^2[KCA] \cup EQ[KCA]) \} \vdash \neg(\neg(s \lor t))
\]
as desired, where \( \neg(\neg(s \lor t)) =_{syn} \neg(q) \).
(Inductive case: 3) \( \alpha \) is of the form \( q \land r \), where \( q \) and \( r \) are each an arbitrary propositional formula with \( \frac{n}{2} \) logical operators.

By our supposition, \( \text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \vdash q^1 \land q^2 \land r^1 \land r^2 \). By the basic semantic definition of propositional logic, we get that

\[
\text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \vdash q^1 \land q^2, \tag{A.20}
\]

\[
\text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \vdash r^1 \land r^2. \tag{A.21}
\]

Applying the inductive hypothesis to (A.20), we get that \( \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \vdash q \). Analogously, applying the inductive hypothesis to (A.21), we get that \( \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \vdash r \). Therefore, we establish that \( \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \vdash q \land r \) as desired.

We have shown that for any propositional atom \( \alpha \) or any propositional non-atomic formula \( \alpha \) constructed using only logical operators \( \neg \) and/or \( \land \), \( \alpha \in \{ \alpha_1, \alpha_2 \} \subseteq \text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \rightarrow \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \vdash \alpha \).

(If Part (one of two):) Let \( \alpha \) be an arbitrary propositional formula in \( \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \) \( \subseteq \text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \rightarrow \{ \alpha \mid \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \vdash \alpha^1 \land \alpha^2\} \cup \{ \alpha \mid \alpha \in \mathcal{L}_P \cap \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \vdash \alpha \).

By our supposition, \( \text{Cn}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) ) \) \( \vdash \alpha^1 \land \alpha^2 \). Assume, for the sake of deriving a contradiction, that \( \text{Cn}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \not\vdash \alpha^1 \land \alpha^2 \). This assumption implies that there exists some truth assignment \( M : \{ \text{P}(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \cup \text{P}(\alpha^1) \cup \text{P}(\alpha^2) \} \rightarrow \{ \top, \bot \} \) such that \( M \) satisfies \( (K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup \text{EQ}) \) but falsifies \( \alpha^1 \land \alpha^2 \).

Since \( \text{EQ} \supseteq \text{EQ}|\text{KCA} \) and \( \text{Domain}(M) \supseteq \{ \text{P}(K_1^1|\text{KCA}) \cup K_2^2|\text{KCA} \cup R^1|\text{KCA} \cup R^2|\text{KCA} \cup \text{EQ}|\text{KCA}) \} \setminus \text{P} \), we can augment truth assignment \( M \) to obtain truth assignment \( M' \)
which satisfies \((K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\). Let's specify truth assignment \(M' : [\text{Domain}(M) \cup \mathcal{P} \cap \mathcal{P}(K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])] \rightarrow \{\top, \bot\}\) as follows:

\[
M'(x) = \begin{cases} 
M(x) & \text{if } x \in \text{Domain}(M) \\
M(y) & \text{if } x \notin \text{Domain}(M), \ x \in \mathcal{P}, \ y \in \{x_1, x_2\}, \\
\text{undefined} & \text{otherwise} 
\end{cases}
\]

\(M'\) assigns a truth value to each atom appearing in \((K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\) since \(\text{Domain}(M') \supseteq \mathcal{P}(K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\). Moreover, by Lemma 4, \(M'\) is a satisfying truth assignment for \((K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\) because it preserves the truth values assigned by \(M\); that is, \(M'(p) = M(p^1) = M(p^2)\) as \((p^1 \equiv p^2) \in EQ\) by Lemma 4 for any atom \(p \in \text{Domain}(M') \cap P\), while \(M'(p^i) = M(p^i)\) for any \(i \in \{1, 2\}\) and any atom \(p^i \in \text{Domain}(M)\). Since \(M'(^1) = M(\alpha^1)\) and \(M'(^2) = M(\alpha^2)\), we get that \(M'(\alpha^1 \land \alpha^2) = M(\alpha^1 \land \alpha^2) = \bot; \ M', \) which satisfies \((K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\), falsifies \((\alpha^1 \land \alpha^2)\). This implies that \(\text{CN}(K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\) \(\not\models \alpha^1 \land \alpha^2\), yielding a contradiction. Thus, the assumption that \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) \(\not\models \alpha^1 \land \alpha^2\) must be wrong, and we get that \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) \(\models \alpha^1 \land \alpha^2\), which holds, in propositional logic, if \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) \(\vdash \alpha^1 \land \alpha^2\), meaning that \(\alpha \in \{\alpha \mid \{\alpha^1, \alpha^2\} \subseteq \text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\}\).

(If Part (two of two):) Suppose \(\alpha\) is an arbitrary propositional formula in \(\{\alpha \mid \alpha \in \mathcal{L}_P \cap \text{CN}(K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\}\). We prove that \(\alpha \in \{\alpha \mid \{\alpha^1, \alpha^2\} \subseteq \text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\}\).

To derive a contradiction, assume that \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) \(\not\vdash \alpha^1 \land \alpha^2\). However, according to how the annotation [KCA] is defined and by Lemma 4, \(\text{CN}(K_1^1[KCA] \cup K_2^2[KCA] \cup R_1^1[KCA] \cup R_2^2[KCA] \cup EQ[KCA])\) \(\vdash \alpha\) implies that

\[
\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)) \vdash \alpha^1 \lor \alpha^2, \quad \text{and} \quad (A.22)
\]

\[
\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)) \vdash \alpha^1 \equiv \alpha^2. \quad (A.23)
\]

(A.22), along with the assumption that \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) \(\not\vdash \alpha^1 \land \alpha^2\), yields that there exists a model \(M\) of \(\text{CN}(K_1^1 \cup K_2^2 \cup R_1 \cup R_2 \cup EQ)\) that satisfies \(\alpha^1 \neq \alpha^2\). This,
however, contradicts (A.23). Thus, the assumption that \( Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \not\vdash \alpha^1 \land \alpha^2 \) must be wrong, and we establish that \( Cn(K_1^1 \cup K_2^2 \cup R^1 \cup R^2 \cup EQ) \vdash \alpha^1 \land \alpha^2 \) as desired.

Therefore, for any arbitrary \( EQ \) set (whose index is arbitrarily chosen by some selection function \( c \)) of \( B \), \( \Delta_c ((K_j)_{j \in J}, R, C) \equiv \Delta_c [KCA] ((K_j)_{j \in J}, R, C) \). Furthermore, for \( B \), there is a correspondence between the consistent symmetric belief change extension determined by (the arbitrarily chosen) \( EQ \) and that determined by the corresponding set \( EQ[KCA] \). Consequently, we have that \( \Delta ((K_j)_{j \in J}, R, C) \equiv \Delta [KCA] ((K_j)_{j \in J}, R, C) \), concluding the proof of Theorem 11.

A.4 Proof of Theorem 12

Given the similarity (noted in Section 3.4.2) between belief revision and projected merge in Delgrande and Schaub's framework [15, 16, 17], this proof is modeled after that of the corresponding Theorem 5.4 in [15].

Let \( B = ((K_j)_{j \in J}, R, C) \) be a multi belief change scenario in \( \mathcal{L}_P \). We aim to show that \( \nabla ((K_j)_{j \in J}, R, C) \equiv \nabla [W] ((K_j)_{j \in J}, R, C) \) by making use of Lemma 5.

**Lemma 5** Let \( EQ \) be any maximal set of equivalences determining a consistent projected belief change extension of \( ((K_j)_{j \in J}, R, C) \) in \( \mathcal{L}_P \). Then \( \{p^j \equiv p \mid p \in (\mathcal{P}(K_j) \setminus [KCA \setminus CA]) \mbox{ and } j \in J\} \cup \{p^j \equiv p \mid p \in (\mathcal{P}(R \cup C) \setminus CA) \mbox{ and } j \in J\} \subseteq EQ \).

**Proof 5** Assume otherwise (to derive a contradiction). Thus, there is a projected belief change extension of \( B \) whose corresponding \( EQ \) is such that:

1. \( \exists j \in J, p \in \mathcal{P} \mbox{ where } p \in (\mathcal{P}(K_j) \setminus KCA \setminus CA) \mbox{ and } (p^j \equiv p) \not\in EQ \), or
2. \( \exists j \in J, p \in \mathcal{P} \mbox{ where } p \in (\mathcal{P}(R \cup C) \setminus CA) \mbox{ and } (p^j \equiv p) \not\in EQ \).

- For the first case, we have from Lemma 1 that for some \( \phi \in C \cup \{\perp\} \), \( \bigcup_{j \in J} K_j^j \cup R \cup \{\neg \phi\} \cup EQ \vdash (\neg p^i \equiv p) \), or equivalently \( \bigcup_{j \in J} K_j^j \cup R \cup \{\neg \phi\} \cup EQ \vdash (p^j \lor p) \land (\neg p^j \lor \neg p) \).

Thus,

\[
\bigcup_{j \in J} K_j^j \cup R \cup \{\neg \phi\} \cup EQ \vdash p^j \lor p, \tag{A.24}
\]
\[ \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash \neg p^j \lor \neg p. \]  
(A.25)

We have that \( p \not\in \mathcal{P}(R \cup \{\neg \phi\}) \) by assumption, and clearly \( p \not\in \mathcal{P}(EQ) \) and \( p \not\in \mathcal{P}(\bigcup_{j \in J} K^j_j) \). That is, neither \( p \) nor \( \neg p \) appears on the left hand side of \( \vdash \) in (A.24) and (A.25).

Thus, from (A.24) we must have

\[ \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash p^j. \]  
(A.26)

Analogously, from (A.25) we derive

\[ \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash \neg p^j. \]  
(A.27)

However, (A.26) and (A.27) together yield that \( \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash \bot \), contradicting the fact that \( EQ \) determines a consistent projected belief change extension.

- For the second case, we derive, as previously, (A.24) and (A.25). Since \( p \in (\mathcal{P}(RUC) \setminus CA) \) by assumption, \( p^j \not\in \mathcal{P}(\bigcup_{j \in J} K^j_j) \). As well, \( p^j \not\in \mathcal{P}(R \cup \{\neg \phi\}) \) and \( p^j \not\in \mathcal{P}(EQ) \). Analogous to the first case, we obtain a contradiction via \( \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash p \) and \( \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \vdash \neg p \).

We now need to show that for any set \( EQ \) of equivalences determining some consistent projected belief change extension of \( B = ((K_j)_{j \in J}, R, C) \), we have that for every \( \phi \in C \cup \{\bot\} \),

\[ Cn(\bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ) \cap L_P = Cn(\bigcup_{j \in J} K^j_j[W] \cup R \cup \{\neg \phi\} \cup EQ[W]) \cap L_P \]

where \( EQ[W] = EQ \setminus \{p^j \equiv p \mid p \not\in W \text{ or } p \not\in \mathcal{P}(K_j)\} \). Thus, there is a one-to-one correspondence between sets \( EQ \) and \( EQ[W] \) determining belief change extensions of \( B \).

We prove each part in turn.

\textbf{(Only-if Part:) } We show that any model of \( \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \) is also a model of \( \bigcup_{j \in J} K^j_j[W] \cup R \cup \{\neg \phi\} \cup EQ[W] \). Let \( M \) be any model of \( \bigcup_{j \in J} K^j_j \cup R \cup \{\neg \phi\} \cup EQ \). From substitution of equivalent formulas (here in \( EQ \)) and Lemma 5, we get that \( \bigcup_{j \in J} K^j_j \cup EQ \vdash \bigcup_{j \in J} K^j_j[W] \). Thus, \( M \), being a model of \( \bigcup_{j \in J} K^j_j \cup EQ \), is also a model of \( \bigcup_{j \in J} K^j_j[W] \). Since \( EQ[W] \subseteq EQ \) and \( M \) is a model of \( EQ \), \( M \) is also a model of \( EQ[W] \). Thus, \( M \) is a model of \( \bigcup_{j \in J} K^j_j[W] \cup R \cup \{\neg \phi\} \cup EQ[W] \).
(If Part:) We show that for arbitrary $\delta \in \mathcal{L}_P$, any proof of $\bigcup_{j \in J} K_j^2[W] \cup R \cup \{-\phi\} \cup EQ[w] \vdash \delta$ can be transformed into a proof of $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ \vdash \delta$. Let $\delta_1, ..., \delta_n = \delta$ be a proof of $\delta$ from $\bigcup_{j \in J} K_j^2[W] \cup R \cup \{-\phi\} \cup EQ[w]$. We construct a proof of $\delta$ from $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ$ as follows.

For $\delta_k, 1 \leq k \leq n$, we have the six following cases.

1. $\vdash \delta_k$: We leave $\delta_k$ unchanged.
2. $\delta_k \in \bigcup_{j \in J} K_j^2[W]$: It follows easily that $\bigcup_{j \in J} K_j^2 \cup EQ \vdash \delta_k$. We replace $\delta_k$ by a proof (sequence of formulas) of $\delta_k$ from $\bigcup_{j \in J} K_j^2 \cup EQ$.
3. $\delta_k \in R$: We leave $\delta_k$ unchanged.
4. $\delta_k = \{-\phi\}$: We leave $\delta_k$ unchanged.
5. $\delta_k \in EQ[w]$: We leave $\delta_k$ unchanged.
6. $\delta_k$ results from $\delta_i, \delta_j, 1 \leq i, j \leq k$ by an application of modus ponens: Since, by induction hypotheses, $\delta_i$ and $\delta_j$ are logical consequences of $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ$ and $\delta_j$ is $\delta_i \supset \delta_k$, we obtain $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ \vdash \delta_k$ by modus ponens.

Hence, we obtain a sequence of formulas where each formula is

1. a tautology (case 1),
2. a premiss drawn from the set $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ$ (cases 2-5), or
3. obtained from previous formulas in the sequence by an application of modus ponens (case 6).

Hence, we have shown that $\bigcup_{j \in J} K_j^2 \cup R \cup \{-\phi\} \cup EQ \vdash \delta$.

Therefore, for any arbitrary $EQ$ set (whose index is arbitrarily chosen by some selection function $c$) of $B$, $\nabla_c ((K_j)_{j \in J}, R, C) \equiv \nabla_c [w] ((K_j)_{j \in J}, R, C)$. Furthermore, for $B$, there is a one-to-one correspondence between the consistent projected belief change extension determined by (the arbitrarily chosen) $EQ$ and that determined by the corresponding set $EQ[w]$. Consequently, we have that $\nabla ((K_j)_{j \in J}, R, C) \equiv \nabla [w] ((K_j)_{j \in J}, R, C)$, concluding the proof of Theorem 12.
Bibliography


