COBA 2.0: A Consistency-Based Belief Change System

James P. Delgrande[1], Daphne H. Liu[1], Torsten Schaub[2] ⋆, and Sven Thiele[2]

[1] School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
[2] Institut für Informatik, Universität Potsdam, August-Bebel-Str. 89, D-14482 Potsdam, Germany

Abstract. We describe COBA 2.0, an implementation of a consistency-based framework for expressing belief change, focusing here on revision and contraction, with the possible incorporation of integrity constraints. This general framework was first proposed in [1]; following a review of this work, we present COBA 2.0's high-level algorithm, work through several examples, and describe our experiments. A distinguishing feature of COBA 2.0 is that it builds on SAT-technology by using a module comprising a state-of-the-art SAT-solver for consistency checking. As well, it allows for the simultaneous specification of revision, multiple contractions, along with integrity constraints, with respect to a given knowledge base.

1 Introduction

Given a knowledge base and a sentence for revision or contraction, the fundamental problem of belief change is to determine what the resulting knowledge base contains. The ability to change one's knowledge is essential for an intelligent agent. Such change in response to new information is not arbitrary, but rather is typically guided by various rationality principles. The best known of these sets of principles was proposed by Alchourron, Gardenfors, and Makinson [2], and has come to be known as the AGM approach.

In this paper, we describe COBA 2.0, an implementation of a consistency-based approach to belief revision and contraction. The general methodology was first proposed in [1]. In this approach, the AGM postulates for revision are effectively satisfied, with the exception of one of the "extended" postulates. Similarly the contraction postulates are satisfied with the exception of the controversial recovery postulate and one of the extended postulates. Notably the approach is syntax independent, and so independent of how a knowledge base and sentence for belief change is represented. COBA 2.0 implements this approach, and in a more general form. Thus a single belief change operation will involve a single knowledge base and (possibly) a sentence for revision, but along with (possibly) a set of sentences for contraction; as well integrity constraints are handled, and in a straightforward fashion.

In Section 2, we give background terminology, notation, and implementation considerations. Section 3 presents COBA 2.0's high-level algorithm, in addition to working through two examples. Section 4 discusses COBA 2.0's features, syntax, and input checks, while Section 5 describes our experiments evaluating COBA 2.0 against a comparable solver. Lastly, Section 6 concludes with a summary.

⋆ Affiliated with Computing Science at Simon Fraser University and IIIS at Griffith University.

2 Preliminaries

To set the stage, we informally motivate our original approach to belief revision; contraction is motivated similarly, and is omitted here given space considerations. First, the syntactic form of a sentence doesn't give a clear indication as to which sentences should or should not be retained in a revision. Alternately, one can consider interpretations, and look at the models of K and α. The interesting case occurs when $K \cup \{\alpha\}$ is unsatisfiable because K and α share no models. Intuitively, a model of $K \dot{+} \alpha$ should then contain models of α, but incorporating "parts" of models of K that don't conflict with those of α. That is, we will have $Mod(K \dot{+} \alpha) \subseteq Mod(\alpha)$, and for $m \in Mod(K \dot{+} \alpha)$ we will want to incorporate whatever we can of models of K.

We accomplish this by expressing K and α in different languages, but such that there is an isomorphism between atomic sentences of the languages. In essence, we replace every occurrence of an atomic sentence p in K by a new atomic sentence p', yielding knowledge base K' and leaving α unchanged. Clearly, under this relabelling, the models of K' and α will be independent, and $K' \cup \{\alpha\}$ will be satisfiable (assuming that each of K, α are satisfiable). We now assert that the languages agree on the truth values of corresponding atoms wherever consistently possible. So, for every atomic sentence p, we assert that $p \equiv p'$ whenever this is consistent with $K' \cup \{\alpha\}$ along with the set of equivalences obtained so far. We obtain a maximal set of such equivalences, call it EQ, such that $K' \cup \{\alpha\} \cup EQ$ is consistent. A model of $K' \cup \{\alpha\} \cup EQ$ then will be a model of α in the original language, wherein the truth values of atomic sentences in K' and α are linked via the set EQ. A candidate "choice" revision of K by α consists of $K' \cup \{\alpha\} \cup EQ$ re-expressed in the original language. General revision corresponds to the intersection of all candidate choice revisions. The following section gives an example, once we have given a formal summary of the approach.

2.1 Formal Preliminaries

We deal with propositional languages and use the logical symbols $\top$, $\bot$, $\neg$, $\vee$, $\wedge$, $\supset$, and $\equiv$ to construct formulas in the standard way. We write $\mathcal{L}_{\mathcal{P}}$ to denote a language over an alphabet $\mathcal{P}$ of propositional letters or atomic propositions. Formulas are denoted by the Greek letters α, β, α_1, ... Knowledge bases, identified with belief sets or deductively-closed sets of formulas, are denoted by K, K_1, ... So $K = Cn(K)$, where $Cn(\cdot)$ is the deductive closure in classical propositional logic of the formula or set of formulas given as argument. Given an alphabet $\mathcal{P}$, we define a disjoint alphabet $\mathcal{P}'$ as $\mathcal{P}' = \{p' \mid p \in \mathcal{P}\}$. For $\alpha \in \mathcal{L}_{\mathcal{P}}$, α' is the result of replacing in α each proposition $p \in \mathcal{P}$ by the corresponding proposition $p' \in \mathcal{P}'$ (and hence an isomorphism between $\mathcal{P}$ and $\mathcal{P}'$). This definition applies analogously to sets of formulas.

A *belief change scenario* in $\mathcal{L}_{\mathcal{P}}$ is a triple $B = (K, R, C)$ where K, R, and C are sets of formulas in $\mathcal{L}_{\mathcal{P}}$. Informally, K is a belief set that is to be modified so that the formulas in R are contained in the result, and the formulas in C are not. An extension determined by a belief change scenario is defined as follows.

Definition 1 (Belief Change Extension). *Let $B = (K, R, C)$ be a belief change scenario in $\mathcal{L}_{\mathcal{P}}$, 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 $Cn(K' \cup R \cup EQ) \cap \mathcal{L}_{\mathcal{P}}$ *is a* belief change extension *of* B. *If there is no such set* EQ*, then* B *is* inconsistent *and* $\mathcal{L}_{\mathcal{P}}$ *is defined to be the sole* (inconsistent) belief change extension *of* B.

In Definition 1, "maximal" is with respect to set containment, and the exclusive use of "$\{\perp\}$" is to take care of consistency if $C = \emptyset$. Definition 1 provides a very general framework for specifying belief change. Next, we can restrict the definition to obtain specific functions for belief revision and contraction.

Revision and Contraction. For a given belief change scenario, there may be more than one consistent belief change extension. We can thus use a *selection function* c that, for any set $I \neq \emptyset$, has as value some element of I.

Definition 2 (Revision). *Let* K *be a knowledge base,* α *a formula, and* $(E_i)_{i\in I}$ *the family of all belief change extensions of* $(K, \{\alpha\}, \emptyset)$*. Then, we define*

1. $K \dot{+}_c \alpha = E_i$ *as a* choice revision *of* K *by* α *with respect to some selection function* c *with* $c(I) = i$.
2. $K \dot{+} \alpha = \bigcap_{i\in I} E_i$ *as the* (skeptical) revision *of* K *by* α.

Definition 3 (Contraction). *Let* K *be a knowledge base,* α *a formula, and* $(E_i)_{i\in I}$ *the family of all belief change extensions of* $(K, \emptyset, \{\alpha\})$*. Then, we define*

1. $K \dot{-}_c \alpha = E_i$ *as a* choice contraction *of* K *by* α *with respect to some selection function* c *with* $c(I) = i$.
2. $K \dot{-} \alpha = \bigcap_{i\in I} E_i$ *as the* (skeptical) contraction *of* K *by* α.

A *choice* change represents a feasible way in which a knowledge base can be revised or contracted to incorporate new information. On the other hand, the intersection of all choice changes represents a "safe," *skeptical* means of taking into account all choice changes.

Table 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 maximal consistent EQ set(s); the last column gives the results of the revision, as a finite representation of $Cn(K \dot{+} \alpha)$. For $\{p \wedge q\} \dot{+} (\neg p \vee \neg q)$, there

K'	α	EQ	$K \dot{+} \alpha$
$p' \wedge q'$	$\neg q$	$\{p \equiv p'\}$	$p \wedge \neg q$
$\neg p' \equiv q'$	$\neg q$	$\{p \equiv p', q \equiv q'\}$	$p \wedge \neg q$
$p' \vee q'$	$\neg p \vee \neg q$	$\{p \equiv p', q \equiv q'\}$	$p \equiv \neg q$
$p' \wedge q'$	$\neg p \vee \neg q$	$\{p \equiv p'\}, \{q \equiv q'\}$	$p \equiv \neg q$

Table 1. Skeptical Revision Examples

are two maximal consistent EQ sets $\{p \equiv p'\}$ and $\{q \equiv q'\}$ and thus two corresponding choice extensions $Cn(p \wedge \neg q)$ and $Cn(\neg p \wedge q)$, respectively. Table 2 lists four skeptical contraction examples.

K'	α	EQ	$K\dot{-}\alpha$
$p' \land q'$	q	$\{p \equiv p'\}$	p
$p' \land q' \land r'$	$p \lor q$	$\{r \equiv r'\}$	r
$p' \lor q'$	$p \land q$	$\{p \equiv p', q \equiv q'\}$	$p \lor q$
$p' \land q'$	$p \land q$	$\{p \equiv p'\}, \{q \equiv q'\}$	$p \lor q$

Table 2. Skeptical Contraction Examples

The general approach, with $|C| > 1$, can be employed to express *multiple contraction* [3], in which contraction is with respect to a set of (not necessarily mutually consistent) sentences. Therefore, we can use the belief change scenario $(K, \emptyset, \{\alpha, \neg\alpha\})$ to represent a *symmetric contraction* [4] of α from K. Refer to [1] for a discussion of the formal properties of these belief revision and contraction operators.

Integrity Constraints. Definition 1 allows for simultaneous revision and contraction by sets of formulas. This in turn leads to a natural and general treatment of integrity constraints. To specify a belief change incorporating a set of *consistency-based* integrity constraints [5, 6], IC_c, and a set of formulas as entailment-based constraints [7], IC_e, one can specify a belief change scenario by $(K, R \cup IC_e, C \cup \overline{IC_c})$, where K, R, and C are as in Definition 1, and $\overline{IC_c} = \{\neg\phi \mid \phi \in IC_c\}$. See [1] for details.

2.2 Implementation Considerations

Finite Representation. Definitions 1–3 provide an abstract characterization of revision and contraction, yielding in each case a deductively-closed belief set. It is proven in [1] that the same (with respect to logical equivalence) operators can be defined so that they yield a knowledge base consisting of a finite formula. Consider $K \dot{+} \alpha$. Via Definitions 1 and 2, we determine maximal sets EQ where $\{K'\} \cup \{\alpha\} \cup EQ$ is consistent. For each such EQ set, we carry out the substitutions:

- for $p \equiv p' \in EQ$, replace p' with p in K',
- for $p \equiv p' \notin EQ$, replace p' with $\neg p$ in K'.

It is shown that following this substitution, the resulting knowledge base and input formula is logically equivalent to some choice revision; the disjunction of all such resulting knowledge bases and input formula is equivalent to the skeptical revision.

For contraction (where $C \neq \emptyset$), we need to substitute into the resulting K all possible combinations of truth value assignments for all elements in $P_{\overline{EQ}}$. Again, see [1] for details.

Limiting Range of EQ. The range of EQ can be limited to "relevant" atoms. Intuitively, if an atomic sentence appears in a knowledge base K but not in the sentence for revision α, or vice versa, then that atomic sentence plays no part in the revision process. The same intuition extends to contraction. It was proven in [1] that for computing a belief change extension of a belief change extension $B = (K, R, C)$, we need consider only those atoms common to K and to $(R \cup C)$. That is, if $Atoms(X)$ is the set of atoms in set of formulas X, then in Definition 1 for forming K' and the set EQ we can limit ourselves to considering atoms in $Atoms(K) \cap (Atoms(R) \cup Atoms(C))$.

3 Algorithm

The results at the end of the last section lead to an algorithm for computing a belief change extension for an arbitrary belief change scenario. After presenting our algorithm, we will work through two example belief change scenarios.

Given a set K of formulas in $\mathcal{L}_{\mathcal{P}}$, and sets Rev, IC_e, Con, and IC_c of formulas in $\mathcal{L}_{\mathcal{P}}$ for revision, entailment-based integrity constraints, contraction, and consistency-based integrity constraints, respectively, algorithm $ComputeBCE$ returns a formula whose deductive closure is a belief change extension of the belief change scenario $B = (K, Rev \cup IC_e, Con \cup \overline{IC_c})$, where $\overline{IC_c} = \{\neg\phi \mid \phi \in IC_c\}$.

Algorithm $ComputeBCE$ invokes the following auxiliary functions:

$Atoms(S)$ Returns the set of atoms appearing in any formula in set of formulas S.

$Prime(K, CA)$ For set of formulas K and set of atoms CA, returns K but where every atom $p \in A$ is replaced by p'.

$Initialize(K', R, Con, IC_c)$ Given a formula K' and sets R, Con, IC_c of formulas, returns a set of formulas of form $(K' \wedge (\bigwedge R) \wedge \neg\phi \wedge \psi)$, for each $\phi \in (Con \cup \{\bot\})$ and $\psi \in (IC_c \cup \{\top\})$.

$Replace(K', p', p)$ Returns K' with every occurrence of atom p' replaced by p.

$ForgetOutEquiv(K', Out)$ Input: formula K' and a set Out of equivalences of atoms
Output: K' with every atom p such that $(p' \equiv p) \in Out$ is "forgotten":

```
1. If $Out = \emptyset$, then return $K'$.
2. $OutAtoms := \{p \mid (p' \equiv p) \in Out\}$.
3. $TA := PowerSet(OutAtoms)$.
  //$TA$ is the set of all truth assignments to $OutAtoms$.
4. $KDisj := \bot$.
5. For each truth assignment $\pi \in TA$ {
     $TempK := K'$.
     $KDisj := KDisj \vee Substitute(TempK, \pi)$. }
     //$Substitute$ returns $\pi$ applied to $TempK$.
6. Return $KDisj$.
```

```
**Algorithm** $ComputeBCE(K, Rev, IC_e, Con, IC_c)$
Let $R = Rev \cup IC_e$ and $C = Con \cup \overline{IC_c}$.
1. If $R \vdash \bot$ or $K \vdash \bot$, then return $\bot$.
2. If (for any $\psi \in IC_c$, $R \cup \{\psi\} \vdash \bot$), then return $\bot$.
3. If (for any $\phi \in Con$, $R \cup \{\neg\phi\} \vdash \bot$), then return $\bot$.
4. If (for any $\phi \in Con$ and any $\psi \in IC_c$
        $\{\neg\phi\} \cup \{\psi\} \vdash \bot$), then return $\bot$.
5. $CA := Atoms(K) \cap (Atoms(R) \cup Atoms(C))$.
6. $K' := Prime(K, CA)$.
7. $KR\overline{C} := Initialize(K', R, Con, IC_c)$.
8. $In := Out := \emptyset$.
9. For each $e \in \{p' \equiv p \mid p \in CA\}$ {
        If ( for any $\theta \in KR\overline{C}$   we have   $e \cup \{\theta\} \vdash \bot$ )
```

```
            Then $Out := Out \cup \{e\}$.
            Else $In := In \cup \{e\}$. }
10. For each $e \in In$:      $K' := Replace(K', p', p)$.
11. For each $e \in Out$:      $K' := Replace(K', p', \neg p)$.
12. If ($Con \neq \emptyset$) Then $K' := ForgetOutEquiv(K', Out)$.
13. Return $K' \wedge (\bigwedge Rev)$.
```

Algorithm $ComputeBCE$ generates a belief change extension in non-deterministic polynomial (NP) time; i.e., an extension can be computed by a deterministic polynomial Turing machine using the answers given by an NP oracle. For this purpose, we currently use the SAT-solver called Berkmin in the SAT4J library [8]. The solver performs the consistency checks in lines 1 through 4 and within the for loop in Line 9. Before passing any formula to the solver, we convert it first to conjunctive normal form (CNF). The CNF formula, once created, is saved with its corresponding formula so that conversions are not done repetitively.

The selection function (for the "preferred" EQ set) is left implicit in Line 9 of Algorithm $ComputeBCE$; it is realized by the particular order chosen when treating the atoms in CA. In COBA 2.0, however, we create an ordered (in ascending cardinality) list L of all $2^{|CA|}$ possible subsets of $\{p' \equiv p \mid p \in CA\}$. To help streamline the search for EQ sets and minimize memory usage, we represent each equivalence by a single bit so that it is included in an EQ set e iff its corresponding bit is 1 in e's bit-string. Furthermore, the ordered list L can accommodate our subsequent search for maximal EQ sets, whether the search be breadth-first or depth-first. On average, the running time and memory usage of breadth-first search is comparable to that of depth-first search, although in our experience neither is consistently superior.

3.1 Examples

We illustrate how COBA 2.0 computes belief change extensions by working through two examples. The examples include belief revision and contraction.

Revision. Consider revising a knowledge base $K = \{p, q\}$ by a formula $\alpha = \neg p \vee \neg q$. We show how COBA 2.0 computes $K \dot{+} \alpha$:

1. Find the common atoms between the knowledge base and the revision formula.
   $CA = \{p, q\}$
2. Create a new formula K' from K by priming the common atoms appearing in K.
   $K' = (p' \wedge q')$
3. Find all maximal equivalence sets $EQ = \{b' \equiv b \mid b \in CA\}$ such that $\{K'\} \cup \{\alpha\} \cup EQ$ is satisfiable.
   $EQ_1 = \{p' \equiv p\}$
   $EQ_2 = \{q' \equiv q\}$
4. For each EQ_i, create a belief change extension by (a) unpriming in K' every primed atom p' if $(p' \equiv p) \in EQ_i$, (b) replacing every primed atom p' with $\neg p$ if $(p' \equiv p) \notin EQ_i$, and finally (c) conjoining K' with the revision formula.
   $K \dot{+}_{c_1} \{\alpha\} = (p \wedge \neg q) \wedge (\neg p \vee \neg q) \equiv (p \wedge \neg q)$
   $K \dot{+}_{c_2} \{\alpha\} = (\neg p \wedge q) \wedge (\neg p \vee \neg q) \equiv (\neg p \wedge q)$

5. The resulting knowledge base is the deductive closure of either the disjunction of all belief change extensions for *skeptical* change, or one belief change extension for *choice* change.
   $K \dot{+} \{\alpha\} = Cn((p \wedge \neg q) \vee (\neg p \wedge q))$

Contraction. Consider contracting a knowledge base $K = \{p \vee q\}$ by a formula $\alpha = p \vee q$. We show how COBA 2.0 computes $K \dot{-} \alpha$:

1. Find the common atoms between the knowledge base and the contraction formula.
   $CA = \{p, q\}$
2. Create a new formula K' from K by priming the common atoms appearing in K.
   $K' = (p' \vee q')$
3. Find all maximal equivalence sets $EQ = \{b' \equiv b \mid b \in CA\}$ such that $\{K'\} \cup \{\neg\alpha\} \cup EQ$ is satisfiable.
   $EQ_1 = \{\}$
4. For each EQ_i, create a belief change extension by (a) unpriming in K' every primed atom p' if $(p' \equiv p) \in EQ_i$, (b) replacing every primed atom p' with $\neg p$ if $(p' \equiv p) \notin EQ_i$, and finally (c) taking the disjunction of all possible substitutions of $\top$ or $\perp$ into those atoms in K' that are in CA but whose corresponding equivalences are not in EQ_i.
   $K \dot{-}_{c_1} \{\alpha\} = (\top)$
5. The resulting knowledge base is the deductive closure of either the disjunction of all belief change extensions for $skeptical$ change, or one belief change extension for $choice$ change.
   Here, there is only one resulting knowledge base for skeptical change and for choice change: $K \dot{-} \{\alpha\} = Cn((\neg\perp \vee \neg\perp) \vee (\neg\perp \vee \neg\top) \vee (\neg\top \vee \neg\perp) \vee (\neg\top \vee \neg\top)) = Cn(\top)$

4 Implementation

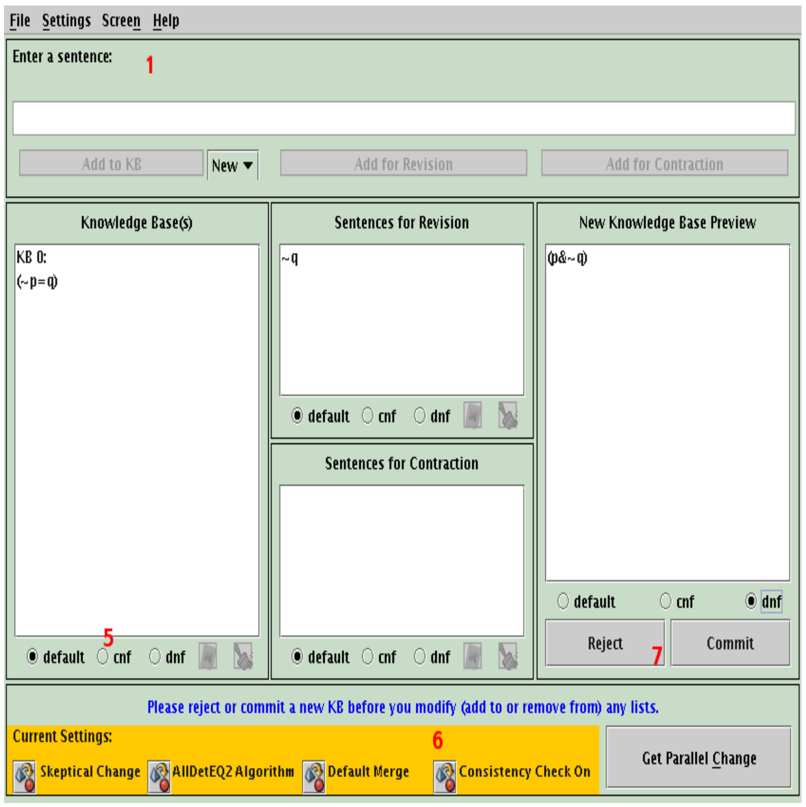
In this section, we describe the COBA 2.0 implementation. We discuss features, syntax, and syntactic and consistency checks on input formulas.

4.1 Features

COBA 2.0 is available as an interactive Java applet, complete with a menu, text boxes, buttons, and separate panels for belief change, integrity constraints, and snapshots. Via the menu, users can import belief change scenarios from files, specify the type (skeptical or choice) of belief change desired, and obtain a resulting knowledge base.

Users may also

1. enter belief change scenarios in text boxes,
2. view logs of the changes made to the knowledge base (KB) list, the entailment-based integrity constraints (EB IC) list, and the consistency-based integrity constraints (CB IC) list,
3. revert to an older KB, EB IC, or CB IC snapshot,



KB 0:
(~p=q)

~q

(p&~q)

Please reject or commit a new KB before you modify (add to or remove from) any lists.

Current Settings:
Skeptical Change AllDetEQ2 Algorithm Default Merge Consistency Check On

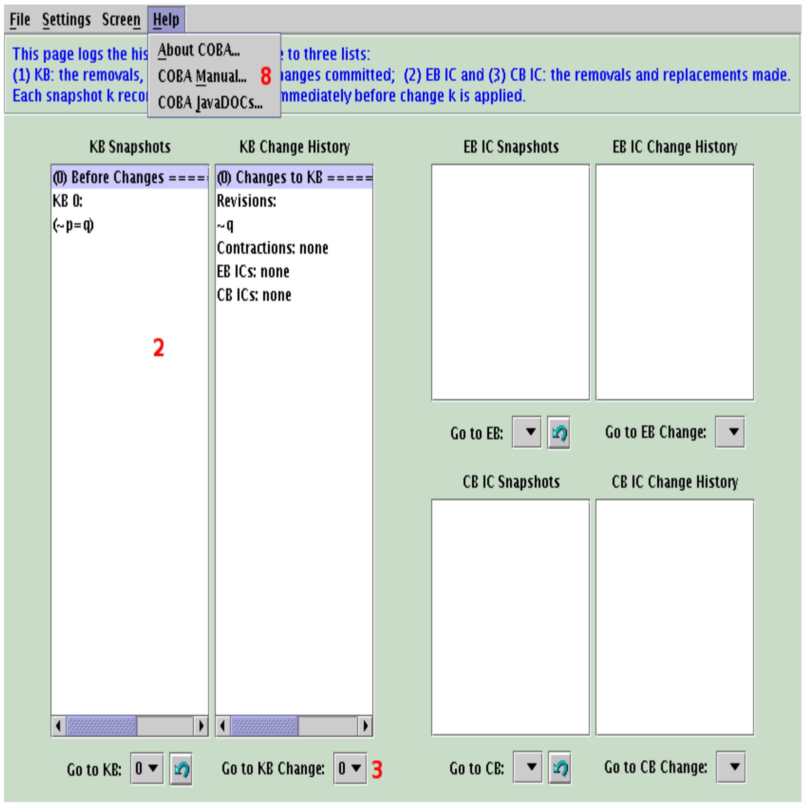
Fig. 1. COBA 2.0's Main Screen

4. save any list to an output file,
5. view formulas in CNF or DNF,
6. turn off the various consistency checks,
7. preview, and then reject or commit, a resulting knowledge base, and
8. view the user manual and JavaDocs in external browser windows (if the applet is running in an html document).

COBA 2.0 automatically simplifies formulas where applicable, for example, eliminating occurrences of $\top$ and $\bot$ in subformulas. COBA 2.0 also automatically informs users of any syntactically ill-formed input formulas. The consistency checks in 6. above and the syntax checks are elaborated on in Subsection 4.3. The applet, user manual, Java code, and Javadocs of COBA 2.0 are accessible from [9].

4.2 Syntax

COBA 2.0 accepts almost all alphanumerical strings for atom names. The exceptions are the symbols in the following list: `'`, `+`, `&`, `^`, `~`, `=`, `>`, (and). Note that `T` and `F` stand for $\top$ and $\bot$, respectively.



This page logs the his… to three lists:
(1) KB: the removals, … hanges committed; (2) EB IC and (3) CB IC: the removals and replacements made.
Each snapshot k reco… mmediately before change k is applied.

(0) Before Changes ====
KB 0:
(~p=q)

(0) Changes to KB =====
Revisions:
~q
Contractions: none
EB ICs: none
CB ICs: none

Fig. 2. COBA 2.0's History Screen

More complex formulas can be built 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 equivalent to B

A top-level formula with a binary connective (`&, +, >,` or `=`) must be enclosed in parentheses. Parentheses within a formula, however, are optional and are used only to enforce precedence. For example, `(a&b+c)` is a valid input sentence and is different from `(a&(b+c))`, whereas a top-level sentence like `a&b` is syntactically ill formed.

Encoding Input Files. Input file formats (for belief change scenarios) vary according to the list (KB, Revision, Contraction, EB IC, or CB IC) to which formulas are to be added. Any KB file to be loaded should precede each knowledge base by a line "KB :" (without the double quotes) and list each formula on a separate line. Each formula is listed on a separate line in any Revision and EB IC input files. For any contraction and

CB IC input files, each line is interpreted as an independent formula for contraction and as a CB IC, respectively.

Consider an example contraction file. While the formula $\boxed{\texttt{(p\&\~{}q)}}$ means that $(p\&\neg q)$ is to be removed from the consequences of the resulting knowledge base, $\boxed{\begin{array}{l}\texttt{p}\\ \texttt{\~{}q}\end{array}}$ listed on two separate lines means that both p and $\neg q$ are to be dropped from the consequences of the resulting knowledge base.

As an example, the next table shows some valid input files.

KB	Rev	Cont	EB IC	CB IC

`KB :`
`(p&q&r)`
`(~q+~s)`
	`q`
	`~p`
		`p`
		`~q`
			`(a&b+c)`
			`(x&(y+z))`
				`d`
				`~d`

4.3 Input Checks

COBA 2.0 performs syntax and consistency checks on all input formulas. The former checks are always enforced, while the latter checks are optional but carried out by default. See below for details.

Syntax Checks. With regard to the syntax detailed earlier in Subsection 4.2, COBA 2.0 informs users of ill-formed input formulas. Thus, for example, the following ill-formed input strings would be flagged with an appropriate message: `q)`, `q+`, `p^`, `p'`, `(p`, `(p&(q)`, `(p+q&)`, and `(+q)`.

Consistency Checks. To preempt inconsistent belief change scenarios, COBA 2.0 prohibits certain kinds of input formulas that result in inconsistent belief change scenarios. This preemptive measure accords well with the consistency checks in lines 1 through 4 of Algorithm $ComputeBCE$ in Section 3. Automatic consistency checks on input formulas, although carried out by default, can be optionally disabled by users wishing to speed up computations. One caveat is that, if these checks are disabled, `F` might be obtained as the resulting knowledge base.

Let $(\bigwedge Rev)$ denote the conjunction of all formulas in Rev for revision, $(\bigwedge EBIC)$ the conjunction of all entailment-based integrity constraints. The following inconsistent belief change scenarios should be avoided; sample error messages, where applicable, are italicised.

1. a contradiction in Rev: *The conjunction of revisions is inconsistent!*
2. a contradiction in $EBIC$: *The conjunction of EB ICs is a contradiction!*
3. a contradiction as a KB, revision, or EB IC formula: No error message; sentence not added.
4. a tautology as a contraction formula: No error message; sentence not added.
5. a contradiction as a CB IC formula: No error message; sentence not added.
6. conflict between $(\bigwedge Rev)$ and $(\bigwedge EBIC)$: *The conjunction of revisions is inconsistent with the conjunction of EB ICs!*
7. conflict between $(\bigwedge Rev)$ and contraction formulas: *The contraction indexed 0 is inconsistent with the conjunction of revisions (indexing starts at 0)!*

8. conflict between $(\bigwedge Rev)$ and CB IC formulas: *The CB IC indexed 1 is inconsistent with the conjunction of revisions (indexing starts at 0)!*
9. conflict between $(\bigwedge EBIC)$ and contraction formulas: *The contraction indexed 6 is inconsistent with the conjunction of EB ICs (indexing starts at 0)!*
10. conflict between $(\bigwedge EBIC)$ and CB IC formulas: *The CB IC indexed 3 is inconsistent with the conjunction of EB ICs (indexing starts at 0)!*
11. conflicting pairs of CB IC formulas and contraction formulas: *The contraction indexed 2 is inconsistent with the CB IC indexed 0 (indexing starts at 0)!*

The aforementioned consistency checks correspond to the consistency checks on input in Algorithm $ComputeBCE$ from Section 3. Specifically, 1, 2, 3, and 6 correspond to the checks $(R \vdash \bot)$ and $(K \vdash \bot)$ in Line 1 of $ComputeBCE$; 5, 8, and 10 to the check ($R \cup \{\psi\} \vdash \bot$, for any $\psi \in IC_c$) in Line 2 of $ComputeBCE$; 4, 7, and 9 to the check ($R \cup \{\neg\phi\} \vdash \bot$, for any $\phi \in Con$) in Line 3 of $ComputeBCE$; lastly, 11 to the check ($\{\neg\phi\} \cup \{\psi\} \vdash \bot$, for any $\phi \in Con$ and any $\psi \in IC_c$) in Line 4 of $ComputeBCE$.

5 Experiments

It has been shown that skeptical revision and contraction in our approach are Π_2^P-hard problems [1]. In [10] is was shown how the approach could be encoded using quantified Boolean formulas (QBF). This allows us to compare COBA 2.0 with an implemented version of the approach using the quantified Boolean formula solver QUIP [11].

For comparing the implementations, we created knowledge bases and revision sentences made up of randomly generated 3-DNF formulas, and converted each to a QBF. We also devised an experimental prototype of COBA 2.0 which performs structural transformation (by replacing sub-formulas with new atoms) instead of the CNF conversion of formulas (for consistency checks). Experiments were then conducted on QUIP, and on both the stable version (the applet) and the experimental prototype of COBA 2.0.

Preliminary experimental results reveal that most of COBA 2.0's run-time is attributed to its structural or CNF conversion of formulas and to its consistency checks. The run-time of all three implementations shows an exponential growth rate. QUIP, however, is relatively faster than both versions of COBA 2.0. The experimental prototype seems to be more than two orders of magnitude faster than the stable version of COBA 2.0, and this observation suggests that structural transformation be done in lieu of CNF conversion in our future implementation.

6 Conclusion

We have presented COBA 2.0, an implementation of a consistency-based approach for belief change incorporating integrity constraints. Operators for belief revision and contraction incorporating integrity constraints are readily defined in a general framework that satisfies the majority of the AGM postulates, notably independence of syntactic representation As demonstrated by COBA 2.0, the framework is easily implementable, for the results of our operators are finite and vocabulary-restricted belief change can be performed instead. Examples of how COBA 2.0 computes belief change are detailed in Section 3.

Our preliminary experiments show that our stable version (the applet) still has much potential for improvement. To this end, we devised an experimental prototype (with structural transformation in lieu of CNF conversion) that seems to be more than two orders of magnitude faster than the stable version (with CNF conversion). Hence, we are optimistic that COBA 2.0 can be improved to achieve a similar run-time behaviour as the monolithic QUIP system.

To our knowledge, COBA 2.0 is the most general belief change system currently available, capable of computing arbitrary combinations of belief revision and contraction that (possibly) incorporate consistency-based and entailment-based integrity constraints. Moreover, COBA 2.0's general framework is easily extensible to consistency-based merging operators as detailed in [12], and currently we are refining our implementation so as to accommodate the merging of knowledge bases. The only comparable system is described in [13]. However, it is based on another approach to belief change, relying on stratified knowledge bases.

The applet, user manual, Java code, and Javadocs of COBA 2.0 are all accessible at [9].

References

1. Delgrande, J., Schaub, T.: A consistency-based approach for belief change. Artificial Intelligence **151** (2003) 1–41
2. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet functions for contraction and revision. Journal of Symbolic Logic **50** (1985) 510–530
3. Fuhrmann, A.: Relevant Logics, Modal Logics, and Theory Change. PhD thesis, Australian National University, Australia (1988)
4. Katsuno, H., Mendelzon, A.: On the difference between updating a knowledge base and revising it. In Gärdenfors, P., ed.: Belief Revision, Cambridge University (1992) 183–203
5. Kowalski, R.: Logic for data description. In Gallaire, H., Minker, J., eds.: Logic and Data Bases. Plenum (1978) 77–103
6. Sadri, F., Kowalski, R.: A theorem-proving approach to database integrity. In Minker, J., ed.: Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann (1987) 313–362
7. Reiter, R.: Towards a logical reconstruction of relational database theory. In Brodie, M., Mylopoulos, J., Schmidt, J., eds.: On Conceptual Modelling. Springer (1984) 191–233
8. A satisfiability library for java. (`http://www.sat4j.org`)
9. COBA 2.0. (`http://www.cs.sfu.ca/˜cl/software/COBA/coba2.html`)
10. Delgrande, J., Schaub, T., Tompits, H., Woltran, S.: On computing belief change operations using quantified boolean formulas. Journal of Logic and Computation **14** (2004) 801–826
11. Egly, U., Eiter, T., Tompits, H., Woltran, S.: Solving advanced reasoning tasks using quantified Boolean formulas. In: Proceedings of the AAAI National Conference on Artificial Intelligence (2000) 417–422
12. Delgrande, J., Schaub, T.: Consistency-based approaches to merging knowledge bases. Journal of Applied Logics. To appear.
13. Benferhat, S., Kaci, S., Berre, D., Williams, M.A.: Weakening conflicting information for iterated revision and knowledge integration. Artificial Intelligence. **153** (2004) 339–371