Formalizing a Deductively Open Belief Space

CSE Technical Report 2000-02

Frances L. Johnson and Stuart C. Shapiro

Department of Computer Science and Engineering, Center for Multisource Information Fusion, and Center for Cognitive Science
State University of New York at Buffalo
226 Bell Hall, Buffalo, NY 14260-2000
flj0@cse.buffalo.edu shapiro@cse.buffalo.edu
January 24, 2000

Abstract

A knowledge representation and reasoning system must be able to deal with contradictions and revise beliefs. There has been much research in belief revision in the last decade, but this research tends to be either in the Coherence camp (AGM) or the Foundations (TMS) camp with little crossover. Most theoretical postulates on belief revision and belief contraction assume a deductively closed belief space - something that is computationally hard (or impossible) to produce in an implementation. This makes it difficult to analyze implemented belief revision systems using the theoretical postulates. This paper offers a formalism that describes a deductively open belief space (DOBS). It then uses this formalism to alter the AGM integrity constraints for a DOBS. A DOBS uses a base set of hypotheses, but only deduces beliefs from that base as the result of specific queries. Thus, it can grow over time even if the base remains static, and can never be referred to as consistent - only either inconsistent or “not known to be inconsistent.” This work and future alterations to the traditional postulate formalisms will better enable system postulate comparisons.

Most formalized postulates for belief revision and belief contraction come from theorists (as opposed to implementers), who assume a deductively closed belief space (DCBS). This is something that is computationally hard (or impossible) to produce in an implementation, which makes it difficult to compare the operations of implemented belief revision systems with the theoretical postulates.

Our research began with the goal of altering the AGM postulates – and others (Hansson 1993) – for a deductively open belief space (DOBS), a belief space that builds up its explicit beliefs gradually. We quickly realized that the first step was to formalize the DOBS followed by altering the AGM integrity constraints upon which the postulates were formed. This paper offers our DOBS formalism and a DOBS version of the integrity constraints. For this paper, we will assume that the belief revision system is complete and uses Classical Propositional Logic. The next section contains a brief overview of AGM terminology and Integrity Constraints for belief revision (Alchourron, Gärdenfors, and Makinson 1985). The sections following define a DOBS and its belief change operations, and propose a DOBS version of the Integrity Constraints. In the final section, we present a discussion of our findings and issues that we plan to explore in the future. Most important is how these DOBS integrity constraints will help us to formulate postulates for DOBS belief change operations.

Introduction

A knowledge representation and reasoning system must be able to deal with contradictions and revise beliefs. There has been much research in belief revision (Martins 1991; Gärdenfors 1992; Martins 1992a; Martins 1992b; Gärdenfors and Rott 1995; Friedman and Halpern 1996), but this research tends to be either in the Coherence camp or the Foundations (TMS) camp with little crossover. Foundations theory states that justifications should be maintained – requiring that all believed propositions must be justified; and, conversely, those that lose their justification should no longer be believed. By contrast, coherence theory focuses on whether the belief space is consistent – i.e. whether a belief coheres with the other beliefs in the current belief space without regard to its justification.

Terminology and DCBS Integrity Constraints

As mentioned above, the system discussed is assumed to be complete and using Classical Propositional Logic. Propositions may also be referred to as sentences. When we refer to a proposition as a belief, we will be specifically referring to a proposition that is currently believed by the system. A proposition is “believed” if the system accepts it (asserts that it is true, considers it an asserted belief). It is unasserted if and only if it is not accepted – this is not the
same as believing its negation. For the purposes of this paper, an inconsistency refers to a pair of contradictory propositions, $P$ and $\neg P$, as opposed to their conjunction, $P \land \neg P$. A belief space is a set of believed propositions.

Gärdenfors and Rott (Gärdenfors and Rott 1995) list four “integrity constraints” or “rationality postulates” for belief revision that are the basic guidelines for developing postulates for belief change:

1. A knowledge base should be kept consistent whenever possible;
2. If a proposition can be derived from the beliefs in the knowledge base, then it should be included in that knowledge base (deductive closure);
3. There should be a minimal loss of information during belief revision;
4. If some beliefs are considered more important or entrenched than others, then belief revision should retract the least important ones.

Constraints 3 and 4 can conflict with each other, so properly weighting and combining them is an open question for both theorists and implementers. For example: How do you choose between retraction of many weak beliefs vs. one strong belief? The deductive closure of a DCBS gives it a decided advantage over the DOBS, which does not have access to its implicit beliefs. All a DOBS can do is minimize the loss of what information it does have.

This paper focuses on constraints 1 and 2. Constraint 1 is implementable depending on your interpretation of the phrase “whenever possible.” We will alter it to clarify what it means in a DOBS system. Constraint 2 as stated precludes the very notion of a DOBS, so we need to define some DOBS terms that can be used to rewrite constraint 2 for a DOBS.

<table>
<thead>
<tr>
<th>Expansion: Addition of a proposition, $p$, to a belief space, $K$, such that $K \cup {p} = Cn(K \cup {p})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contraction: Retraction of a proposition, $p$, from a belief space, $K$, such that $K - p \vdash p$</td>
</tr>
</tbody>
</table>
| Revision: Consistent addition of a proposition, $p$, to a belief space, $K$, such that $K \cup \{p\}$ is consistent, (equivalent to $Cn((K \cup \{p\}) \cup \{p\})$).

Figure 1: Table representing the AGM belief change operations.

When discussing deductive closure, we use the AGM definition of a consequence operation, $Cn$, where $Cn(K)$ denotes the deductive closure of a belief space, $K$, and $K$ is a deductively closed belief space (DCBS) under $Cn$ if $K = Cn(K)$ (Alchourron, Gärdenfors, and Makinson 1985). The AGM belief change operations are shown in Figure 1.

**Why a Deductively Open Belief Space (DOBS)**

The integrity constraints and belief revision operations mentioned above assume a deductively closed belief space (DCBS) within a language $L$. For classical propositional logic, this is an infinite belief space. Even if only one of every set of logically equivalent propositions is included, to make the belief space finite, its size is on the order of $2^n$ sentences, where $n$ is the number of atomic propositions in the language $L$ (e.g., over 4 trillion sentences if $n = 5$). This makes implementation computationally hard if not impossible.

Forming postulates about how a deductively closed belief space would be altered by the various belief change operations is helpful in establishing theoretical guidelines for belief revision. To compare how well these postulates are satisfied by an implemented belief revision system, however, they must be altered to fit the broad constraints of an implemented system: it must function within a finite (and reasonable) amount of time and use a finite memory and reasoning space. Since even a finite belief space that is deductively closed can become unmanageable, implementations must consider using a Deductively Open Belief Space (DOBS).

**Defining the DOBS**

A DOBS, by definition, is a belief space that is not guaranteed to contain all the possible inferences from the beliefs it holds (only some subset of them) or to know all the possible ways that its beliefs can be derived (only those derivations that it has already performed). A DOBS begins as an empty set to which hypotheses can be added. The beliefs derived from those hypotheses, however, are added gradually over time, as it considers them and discovers them derivable — not all at once.

The entire Belief State is represented by a knowledge base, KB. The DOBS is the Belief Space of the knowledge base, BS(KB). Given a propositional language $L$, consisting of all the well-formed propositions formed from some set of proposition letters, a belief state is defined as:

$$KB = \text{def} \prec HYPS, DERS, B, A, J \succ$$

where

$$HYPS \prec L, DERS \prec Cn(HYPS), B \prec HYPS, A \subseteq \{\langle p, false \rangle\}, \text{ and } J \subseteq \{\langle q, true \rangle\}$$

where

$$p \models HYPS \cup DERS \wedge$$

$$q \models DERS \wedge$$
HYPS represents all the hypotheses ever introduced into KB. DERS represents all the propositions ever derived from HYPS, and A and J are the record of just those derivations—in the styles of an ATMS (A) and a JTMS (J). B represents the set of currently believed hypotheses. All propositions in the knowledge base are represented in A by at least one pair. All propositions in DERS are represented in J by at least one pair.

Since a DOBS can have propositions that are derivable but not, yet, derived, we introduce the concept of a proposition, \( \alpha \), being known to be derivable from a set of propositions, \( \alpha \). This is denoted as \( \alpha \vdash_{KB} p \) and is defined by the rules below.

1. A hypothesis is known to derive itself:
   \[ p \vdash HYPS : \{ p \} \vdash_{KB} p. \]
2. A justification set, \( js \), for a proposition, \( p \), is known to derive \( p \):
   \[ \langle p, js \rangle \vdash_{J} js \vdash_{KB} p. \]
3. An origin set for a proposition, \( p \), is known to derive \( p \):
   \[ \alpha \vdash \alpha \vdash_{A} os \vdash_{KB} p. \]
4. \( \alpha \vdash_{KB} p \) is transitive:
   \[ \forall q \left( q \vdash_{\alpha} p \land \alpha \vdash_{KB} q \right) \vdash_{KB} p. \]
5. A superset of a set that derives a proposition also derives that proposition:
   \[ \alpha \vdash_{KB} p \land \alpha \vdash \beta \vdash_{KB} \beta \vdash_{KB} p. \]

A proposition \( p \) can be an element of both HYPS and DERS if it is both asserted as a hypothesis and known to be derivable from some \( \alpha \vdash_{HYPS} \) where \( p \in \alpha \).

D is the set of derived propositions that are currently believed, and BS(KB) is the set of all currently believed propositions (the DOBS):

\[
\begin{align*}
D(KB) & \equiv_{def} \{ \alpha | \alpha \subset DERS \land B \vdash_{KB} \alpha \} \\
BS(KB) & \equiv_{def} B \cup D
\end{align*}
\]

In other words, KB represents all the propositions that exist in the system along with a record of how they were derived, and BS(KB) represents only those propositions that are currently believed. Although a DCBS "forgets" what it no longer believes, its "omniscient" deductive closure allows it to instantly remember anything that is re-believed. The DOBS must keep track of the disbelieved propositions and derivations to avoid having to repeat earlier derivations when disbelieved propositions are returned to the belief space. The diagram in Figure 2 below, shows most of these concepts.

For shorthand purposes, BS(KB) and D(KB) can be written as BS and D respectively when their KB is clear from the context. The information that \( p \vdash HYPS \cup DERS \) can be written in a shorthand version as \( p \in KB \).

This is not to be confused with \( p \in BS \), though the latter implies the former.

**KB-Closure and K-consistency**

Because we are removing the omniscience of a DCBS and its consequence operation, we must "remember" as much as possible about our DOBS, including propositions that we no longer believe. Once a base set of hypotheses, B, is chosen, the closure of B is limited by KB (i.e. by its derivation records in A and J). We call this closure \( Cn_{KB} \), and it is defined below:

\[
Cn_{KB}(B) \equiv_{def} \{ \alpha | B \vdash_{KB} \alpha \} = BS(KB)
\]
A DOBS is inconsistent if a contradiction has been \textit{or can be} derived from its beliefs. A DOBS, BS(KB) or \text{Cn}_{KB}(B), is \textit{k-inconsistent} iff \( \not\exists p \in \text{Cn}_{KB}(B) \land \not\not p \in \text{Cn}_{KB}(B) \). If a DOBS is \textit{not} k-inconsistent, then we will call it \textit{k-consistent} – there are no inconsistencies in \text{Cn}_{KB}(B), it is \textit{not known to be inconsistent}. This means a DOBS can be both inconsistent and k-consistent at the same time: For example, \( B = \{ A, P, P = \not A \} \), but \( -A \) has not yet been derived.

It has been suggested that this is unacceptable, causing a lack of confidence in the information provided by the DOBS system. There are no uncertainties in a real world implementation, and you can never know for sure if you have correct information… Only a system’s best guess. Our goal is to try to make that best guess as good as possible.

It is always the case that, for a given KB = \( \langle \text{HYPS}, \text{DERS}, B, A, J \rangle \), \( A \not\not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \not \no
Query-addition (+_0)

If a one-step query, Q(p, KB), is successful, a new proposition, p, is added by query-addition (+_0) to DERS. The actual query-addition refers to the method of adding pairs to A and J, thus altering them to form an A’ and J’.

For the query to be successful (in one step), it had to find some js s.t. js \cup \{p, q\} \subseteq BS. The tuple \langle p, js \rangle is then query-added to J: J +_0 \langle p, js \rangle = _{def} (J \cup \{\langle p, js \rangle \})

Warning: It is possible to store this derivation information in A, as a \langle p, os \rangle pair, by forming an origin set, os, for \langle p, q \rangle from the union of the origin sets for each of the elements in js. For example: js = \{q, q\rangle \}, where the origin sets for q and q \rightarrow p are \{q \rightarrow p\} and \{q \rightarrow p\}, respectively. The os for p would then be \{q \rightarrow p, q \rightarrow p\}. But there are three things to beware of:
1. If an element of js has more than one origin set, then there will be multiple origin sets for p. (e.g., if q also has the origin set \{q \rightarrow s\}, then p would have a second os: \{q \rightarrow s, q \rightarrow p\}.)

2. To avoid duplication and foster minimalism of the origin sets, use the following definition:
A +_0 \langle p, os \rangle = _{def}
(A \cup \{\langle p, os \rangle \}) - \{\langle p, os \rangle\}, for all os s.t. os : os.
This is to guarantee minimalism and a lack of duplication of the origin set, os, for any given proposition in DERS: i.e.
\forall p, os. \exists p, \alpha \rightarrow A := \exists \beta (p, \beta \rightarrow A \wedge \beta \in \alpha)\).
Continuing the above example: if p is later derived directly from \{q \rightarrow p\}, its new os would be \{q \rightarrow p\}, which would replace the os \{q \rightarrow p, q \rightarrow p\}.

3. If you use A exclusively, you need to consider your algorithms carefully or your KB-closure might be a subset of the KB-closure when js was also used. Continuing our example: After retracting \{q \rightarrow p\} and \{q \rightarrow s, p, q \not\subseteq BS\}, the addition of q as a hypothesis would restore p to BS if the js \{q, q \rightarrow p\} was referenced. But, if only A was available for derivation records, p \not\subseteq BS until a new query is made.

When a query makes recursive queries, the above process is iterated, continually adjusting KB along the way through query-addition. After each query-addition, BS is regenerated (using the KB-Closure operation). It is possible, therefore, to query-add derivable propositions to KB (and, therefore, BS) without ever deriving the initial proposition that was queried for, A.

Query Postulates
Whether it succeeds or fails, the query, Q(A, KB), is a function Q that takes a proposition A and a knowledge base KB = \langle HYPS, DERS, B, A, J \rangle as arguments. It returns an altered knowledge base KB’ = \langle HYPS, DERS’, B, A’, J’ \rangle that has the following properties (where BS and BS’ are used to represent BS(KB) and BS(KB’) respectively):

Q0) If KB is a knowledge base, then Q(A, KB) is a knowledge base
Q1) If A \subseteq BS, then KB’ = KB
Q2) If B \models A, then A \subseteq BS’
Q3) If B \models A, then A \subseteq BS’
Q4) If B is k-inconsistent and A \subseteq BS, then DERS’ = DERS = BS’ = BS = \{A\} (since anything follows from a contradiction)
Q5) If KB’ \models KB and A \subseteq KB’, then\exists p, \beta s.t. \langle p, \beta \rangle \rightarrow J’ \wedge \beta \subseteq BS, and either p = A, or p was derived in a successful attempt to derive A
Q6) If KB’ \models KB and A \subseteq KB’, then\exists p, \beta s.t. \langle p, \beta \rangle \subset J’ \wedge \beta \subseteq BS, and p was derived in an unsuccessful attempt to derive A
\footnote{Q2} – deriving A is not guaranteed if the system is incomplete.
\footnote{Q4} does not apply if the logic is paraconsistent.

The initial KB, KB_0, is the tuple \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle. Note that HYPS, DERS, A, and J increase monotonically with the exception of the recommended minimalism constraint of query-addition to A.

Integrity Constraints for a DOBS

Now that we have formalized a DOBS, we can assess the key changes necessary to adjust the list of integrity constraints (Gärdenfors and Rott 1995) so that they can be used as belief revision guidelines for a DOBS. Alterations are in **bold italics**. Additions or clarifications are in *plain italics*:

1. A knowledge base should be kept **k-consistent** whenever possible.
2. If a proposition can be derived from the beliefs in the knowledge base using the derivations currently known, then it should be included in that knowledge base (kb-closure). Likewise, if a proposition is not in the knowledge base, but can be derived from the beliefs in the knowledge base, it should be produced if queried for.
3. There should be a minimal loss of *the known* information during belief revision.
4. If some beliefs are considered more important or entrenched than others, then belief revision should retract the least important ones.

Constraint 1 suggests that a system should activate belief revision as soon as an inconsistency is detected. Constraint 2 recommends that we avoid re-deriving a proposition from a set of propositions from which it has already been derived. It furthermore suggests that we strive for a system that is as complete as possible, so that, from a set of hypotheses, the user can expect to derive any proposition that would be included in the deductive closure of that set.

Constraint 3 reminds us that we are forced to analyze the system based on the knowledge we "know," a subset of the total knowledge that a DCBS has. Fortunately, this means there is less information to analyze during a retraction and, since we queried to get this information, it is more likely to be of interest to us. A might bother us with belief revision decisions about obscure information than our query-generated DOBS. However, more information does imply better choices about how to minimize information loss, so we return to the issue of constraint 2 and the need to build our belief space as quickly as possible.

Lastly, constraint 4 seems to try no adjustment for compliance with the needs and restrictions of a DOBS. A DCBS (with its deductive closure) might make a connection between some seemingly unimportant proposition and some important information, whereas the limited knowledge of a DOBS might not have made (or derived) that connection. In this sense, a DOBS is still less "reliable" than the theoretical DCBS. All the more reason to focus on constraint 2 – attempting to improve the quality of completeness – in hopes that we can query for those important pieces of information and still make well-informed choices.

### Discussion and Future Work

We have analyzed the concepts of a DOBS and presented a formalism that is flexible enough to be useful to both the coherence theory researchers as well as those working in the foundations camp. The term k-consistent enables a more direct reference to the DOBS state of "not knowing whether a DOBS is consistent." The detailed formalism offers a guideline for obtaining derivation information for a DOBS, using both the ATMS and/or JTMS style; and the DOBS integrity constraints offer implementers basic concepts for optimizing their belief revision systems.

The next step in this research is to formulate belief revision postulates specific to a DOBS, using the altered integrity constraints. We hope to offer a DOBS version of not only the AGM postulates, but also Hansson's base contraction postulates (Hansson, 1993) and postulates proposed for ranked beliefs. We also hope to continue to provide brief comments regarding postulate adherence for paracomplete logics and for incomplete systems. Implemeneters will then, be able to analyze how well their systems meet the standards of the postulates. This is a "pressing issue" (Hansson 1999, p. 367) for those doing belief revision research in Computer Science.

### References


