DEPENDENCY-DIRECTED RECONSIDERATION: AN ANYTIME ALGORITHM FOR HINDSIGHT KNOWLEDGE-BASE OPTIMIZATION

by

Frances Laird Johnson

January 11, 2006

A dissertation submitted to the
Faculty of the Graduate School of
the State University of New York at Buffalo
in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy

Department of Computer Science and Engineering

Acknowledgements

As I think of those who have supported and inspired me over the years, I am overwhelmed by their number. Space limitations and a somewhat legendary scatterbrain memory assure that I will be revising this list for years to come, so I apologize in advance to those who are not mentioned below.

Without question, this research is a testimony to the guidance, patience and perseverance of my advisor, Stuart Shapiro. He is a very rare and unique researcher—a versatile and ever publishing scholar who also loves working with implementations and is an excellent programmer. This made Stu the perfect advisor for my dissertation. I am especially grateful to him for holding the reins tight enough to guide me to this destination yet loose enough to choose my own path. Stu's ethics and professionalism meet the highest standards, and I only hope that I can follow his example.

William Rapaport and Carl Alphonce joined Stu to form my committee. Bill has always been a reliable source of knowledge, grammar, vocabulary, support, and humor. There were many times when Stu and I would settle some point of confusion with "ask Bill, he'll know." Carl's input was an excellent complement to Bill and Stu's, and this combination improved the readability of my work. My thanks also go to Erwin Segal whose feedback from a background outside that of computer science was very helpful; relocation forced him to step down from my committee.

There is no way that I can sufficiently thank Sven Ove Hansson—who agreed to receive a 300-page dissertation during the winter holidays (having never met me or, as far as I know, read my work), and who returned a favorable report with revision suggestions within two weeks! His report will be a permanent part of my collection of treasured correspondence.

I am grateful for the friendship and coding skills of Michael Kandefer, whose assistance not only helped me to refine my implementation, but it freed me to complete this dissertation within the timeframe required.

Ken Regan, Helene Kershner and Jodi Reiner provided a measure of support that was essential to my successful survival and the completion of this work. Together, they gave me (and the graduate school) confidence that I would finish. I am especially grateful to Helene for her consistent support and advice throughout my entire time at UB.

I had an especially bad time in 2005-2006, physically, and traditional medicine failed me. Steve Grande, Joe Cerceo, and John Zilliox kept me out of a wheelchair and, possibly, a hospice bed. This dissertation would have never been completed without their healing knowledge.

The interest and feedback of the members of the SNePS Research Group was invaluable in the development of my thesis. Those not already mentioned include Josephine Anstey, Debra Burhans, Bharat Bushan, Alistair Campbell, Albert Goldfain, Haythem Ismail, Jean-Pierre Koenig, João Martins, David Pierce, and John Santore. I especially commend João for his initial contribution to the development of the SNePS belief revision system. And, I will forever be grateful that I had Haythem and John as my long-term office-mates. They were always there for me, and they are truly exemplary people.

Other UB professors and staff and fellow students (past and present) who have helped, encouraged and advised me during this journey are: Eloise Benzel, Jan Chomicki, Chris Egert, Sally Elder, Margaret Evans, Joann Glinski, Elle Heffner, Bharat Jayaraman, Kathy Pohlman, Bina Ramamurthy, Kenneth Smith, Matthew Stock, Jaynee Straw, Marianne Sullivan, Len Talmy, Pallavi Tambay, Lynne Terrana, and Deborah Walters. Thank you all. Special thanks go to James Llinas who jump-started my belief revision research through funding from a group headed by Joseph Karakowski, who later supported this work (both professionally and financially) for several years.

In addition to those at UB, many researchers outside the UB community have provided valuable feedback and insight. I am especially grateful to Samir Chopra, Doug Lenat, Thomas Meyer, Leora Morgenstern, Renata Wassermann, and Mary-Anne Williams.

To the teachers from my past, without the foundation you laid, I would not be the person I am, today. From my earlier years, I especially thank Bill ”Doc” Chase, Debbie Coates, Mary Louise Kelley, Bobbette Mason, and Betty Richardson. And to Linda Brinkerhoff, Jeannette Neal, and Alison Girod, of Erie Community College, your excellence as teachers inspired me to continue on at the Ph.D. level.

My full support structure of friends and family is too large to include here, but I must mention a select few. To Howard, Bud, Rose, Letty, Margie, Ted, and, especially, Dad—I love you and miss you all. Mirjana, you have been such a help and inspiration to me. Wick and Donna, your advice and encouragement and constant support for both me and Tom has been invaluable. Carol, your unconditional faith in me is an ever-present source of strength. You are my bestest friend in the whole wide world. Clint and Tillie, you have lightened my load in so many ways; thank you for your support. Mom, you were my first teacher and I still have so much to learn from you; I hope I will continue to become more like you every day. To my girls, Elizabeth and Katherine: you are my favorite teachers, and I love you more than tongue can tell. And, Tom, you are my everything; the magnitude of your faith in me and love for me amazes me every day. You’re the best.

For my family

Contents

List of Figures

List of Tables

Abstract

Dependency-directed reconsideration (DDR) is an efficient, anytime algorithm for performing the knowledge-base optimizing operation of reconsideration in an implemented computational system.

Any computational system that stores and reasons with information must have some means for making changes to that information. These changes include (1) adding new information, (2) changing existing information, and (3) removing information—which can be used to resolve contradictions (called *consistency maintenance*). Many fields of computer science use revision techniques—examples include artificial intelligence (AI), robotics, diagnosis, databases, security, and constraint satisfaction problems; and techniques may be applicable across disciplines.

This research takes an AI approach by referring to data as beliefs, and it assumes that there is a distinction between *base beliefs* (information that enters the system from some outside source) and *derived beliefs* (information produced solely by reasoning from some set of base beliefs and dependent upon those base beliefs). The belief *state* of such a system includes its *currently believed* base beliefs (its *belief base*) as well as the base beliefs that have been disbelieved. Information is rarely entered into a system all at once, and changing the order of the incoming information can alter the resulting state of the system in the case where consistency maintenance is ongoing. Assuming that the optimal base is the base that would result from deferring consistency maintenance until all incoming data is assimilated, any deviation from this optimal base is a *negative effect of operation order*, and the resulting state is sub-optimal.

The two main contributions of this research are (1) the introduction and formalization of the hindsight,

belief-base-optimizing operation of *reconsideration* and (2) the development of *dependency-directed reconsideration (DDR)*—an efficient, anytime algorithm for performing reconsideration in an implemented system.

Reconsideration optimizes the base of a sub-optimal belief state by eliminating the negative effects of operation order. There no longer needs to be a choice between having a consistent base ready for reasoning and having an optimal base (by delaying consistency maintenance until all information is gathered). Reconsideration also improves the performance of belief change operations for finite belief bases.

DDR optimizes an already consistent base by using a queue that contains a small, yet relevant, subset of the removed base beliefs. Processing the beliefs on the queue in decreasing order of preference (or credibility) results in the optimization of the base in a computationally friendly way: (1) a consistent base is always available for reasoning; (2) DDR optimizes the most important parts of the base first; (3) the base is always improving as DDR progresses; and (4) DDR can be performed in multiple stages that can interleave with the addition of (and reasoning with) new information. A measure of confidence in an existing base can be determined by the top element in the DDR queue—the lower its credibility, the higher the confidence in the base. DDR works for monotonic logics, but it is *not* restricted to classical propositional logic or ideal reasoning systems—it works for first-order predicate logic and relevance logics, and it works for systems whose reasoning is not guaranteed to be complete.

Chapter 1

Introduction

1.1 Belief Change

It is an agent's prerogative to change its mind. A person's mind contains thoughts or beliefs, and those beliefs must be able to be changed, updated, adjusted, corrected, or, in a word, *revised*. Likewise, many things can be represented by a data set—such as the mind of a single agent, the beliefs of a group of agents, input from sensors, a database for storing information, or a modeling of the world. How to represent these things in a computational way is a continual challenge for knowledge engineers; but, whatever a set represents and however its elements are constructed, there needs to be some means for revising those elements.

The general meaning of the term *revision* encompasses all types of changes to a data set:

1. adding new data to the set
2. changing existing data in the set
3. removing data from the set

These changes include altering the data in a set to eliminate contradictions, which occur when data conflict.

Many areas of computer science need data-revision techniques—examples include artificial intelligence, robotics, diagnosis, databases, security, and constraint satisfaction problems. Although the techniques used to revise data might differ because of the method of representing the data and the data structure in which they are stored, these differences can be superficial, and techniques from one area might be useful in other areas and should be shared across disciplines.

The computational systems we use are becoming more and more complex. As a result, there is an issue of *how* to represent information in a computer so that it can be recalled efficiently. This is addressed by the research field of *knowledge representation* (KR). Systems can also manipulate their stored information to produce new information, so KR is expanded to knowledge representation *and reasoning* (KR&R). Because that body of knowledge can be used to devise plans of action for an embodied cognitive agent (or robot), we see the field expand further to knowledge representation, reasoning, *and action*. Since there seems to be no end to this expansion, I will settle on the KR&R designation to refer to any reasoning system that can reason about and revise its information—regardless of the end goals of the system (e.g., controlling a robot, advising troop deployment, system analysis, etc.).

Because I will be focusing on revising the beliefs in a KR&R system and comparing my revision techniques with research performed by the belief-change community, I will typically refer to the general concept of data revision as *belief change*.

1.2 Notation and Terminology

Before discussing the motivations for this research, I need to present some basic assumptions, notation, and terminology.

For the majority of this dissertation, I will assume the KR&R system uses propositional logic whose propositions are denoted by lowercase letters $(p,q,r,\ldots)$. The propositional language $\mathcal{L}$ is closed under the truth functional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$.

In actuality, the research presented will also work with more complex logics, such as first-order predicate logic and relevance logic [Anderson & Belnap1975, Anderson, Belnap, & Dunn1992]—cf. Sections 3.3.3 (page 97) and 5.6 (page 208).

I will refer to propositions as beliefs, interchangeably. Sets of beliefs are denoted by uppercase letters $(A, B, C, \ldots)$, and there will also be specific cases where uppercase letters are designated for other kinds of sets or sequences. Since I will be discussing sets of beliefs, here is a brief review of the terminology I will use:

- $p \in A$ states that the belief p is an element of the set A (also referred to as, "p is in A"). Example: $r \in \{p, q, r, s\}$.
- $A \subseteq B$ states that the set A is a *subset* of the set B: every element in A is also in B. Note that A may be equivalent to B. Example: $\{1,3\} \subseteq \{1,2,3\} \subseteq \{1,2,3\}$.
- $A \subsetneq B$ states that the set A is a *proper subset* of the set B: every element in A is also in B, *and* there are elements in B that are *not* in A. Example: $\{1,3\} \subsetneq \{1,2,3\}$, but $\{1,2,3\} \not\subsetneq \{1,2,3\}$.
- $A \not\subseteq B$ states that the set A is *not* a subset of the set B: some element in A is *not* in B.

  Example: $\{1,3\} \not\subseteq \{1,2,5\} \not\subseteq \{1,5\}$.
- $B \setminus A$ is the operation of *set subtraction* (also written as $B - A$) which results in the largest subset of the set B that does not contain any elements that are in the set A. Note that elements of A that are not in B have no effect on the outcome (i.e., A need not be a subset of B). Example: $\{1,2,3,4\} \setminus \{1,3,5\} = \{2,4\}$.

If p is true whenever the beliefs in A are true, then A entails p—denoted as $A \models p$. If p can be derived (through a series of inference steps) from some set A, then it is said that A derives p—denoted as $A \vdash p$.

In a sound system, any belief that is derived is entailed. In a complete system, any belief that is entailed can be derived.

The deductive closure of the set A is written as $Cn(A)$, and it is the set of all beliefs that are derivable from A: $Cn(A) =_{def} \{p \mid A \vdash p\}$.

Some set A is consistent if $A \nvdash \perp$, where $\perp$ denotes logical contradiction. In other words, A is consistent if for every $p \in Cn(A), \neg p \notin Cn(A)$, where $\neg p$ is the *negation* of p, or "not p". Note: For monotonic logics, if $A \vdash \perp$ and $A \subseteq A'$, then $A' \vdash \perp$. Similarly, if $A \nvdash \perp$ and $A' \subseteq A$, then $A' \nvdash \perp$. If $A \vdash \perp$, then A is *inconsistent*.

Inconsistent sets are referred to as *nogoods* in the assumption-based truth maintenance system (ATMS) literature [de Kleer1986, Forbus & de Kleer1993]. A kernel for p is, by definition, a set that minimally derives the belief p [Hansson1999].

Definition 1.2.1 The set A is kernel for p (also called a p-kernel) if and only if $A \vdash p$ and $(\forall A' \subsetneq A) : A' \nvdash p$.

There are two main approaches to belief revision that will be discussed in depth in Chapter 2: the *coherence approach* and the *foundations approach*. The *foundations* approach to belief change [Doyle1979, Gärdenfors1992, Hansson1993] distinguishes between (1) a finite set of beliefs called the belief *base* (where the beliefs have independent standing) and (2) the set of derivable (non-base) beliefs—each requiring the support of the base beliefs from which it is derived to be believed. The deductive closure of the base B is the *belief space* [Martins & Shapiro1983] or *theory* [Alchourrón, Gärdenfors, & Makinson1985] of B, and it is infinite.[1] The foundations approach is also called *base belief change*.

In contrast to the foundations approach, where base beliefs support derived beliefs, the *coherence* approach [Alchourrón, Gärdenfors, & Makinson1985] to belief revision does *not* distinguish between a set of base beliefs and beliefs that are derived through reasoning. All beliefs have equal status in the belief space, and they can support each other. The coherence approach defines belief change in terms of the entire belief

[1]Some of the literature also refers to a theory as a *belief set* that is distinguished from a finite set of beliefs called a belief base. I will not use this terminology, because I prefer to refer to a set of beliefs as a "set" or a "belief set" without the requirement that it be deductively closed.

space, which is a deductively closed set of beliefs (a theory). This set is infinite, which makes implementing a coherence approach to belief change a challenge. Another term for the coherence approach is *theory belief change*.

Any implemented system must, inherently, manipulate and reason from a *finite* set of beliefs. These beliefs can be some arbitrary finite representation of of a coherence theory (some A, where $Cn(A)$ equals the theory) as discussed in [Nebel1989], or they can represent the explicit core beliefs from which a cognitive agent reasons, which follows the foundations approach. I feel that the latter is more accurate description of the reasoning systems in use today—where information (input; base beliefs) is entered in to a system from which further reasoning produces inferences (the derived beliefs). Errors in the inferences point to errors in the input information (or, possibly, the reasoning algorithms). These core beliefs are typically referred to as base beliefs, but they are also called hypotheses in [Martins & Shapiro1983, Martins & Shapiro1988]. I will refer to them as base beliefs throughout this dissertation.

An implemented system may not be able to reason completely. This could be due to due to limitations on time, logic, implementation, and/or memory. In this case, there may be derivable beliefs that have not, yet, been derived. *Explicit* beliefs are the beliefs that an agent (or system) is *aware* of having. These include the base beliefs, initially, as well as beliefs the agent has derived from those base beliefs. *Implicit* beliefs are beliefs that are derivable from explicit beliefs, but have *not, yet, been derived*; thus, implicit beliefs and explicit beliefs form two separate, non-intersecting sets whose union equals he deductive closure of the base.[2] Note that an explicit belief may be additionally derivable in some way that has not, yet, been detected by the system. In this case, the belief is explicit, but it has an implicit derivation.

The difference between explicit and implicit beliefs can be seen in Figure 1.1—where the base B and

[2]We define these terms while choosing not to engage in the discussion that there are some implicit beliefs that do *not* follow from other beliefs—beliefs that a person *knows* without being told or having to reason. Because those beliefs (if they truly exist) would need to be explicit base beliefs in an implemented system, I merely distinguish between explicit and implicit beliefs.

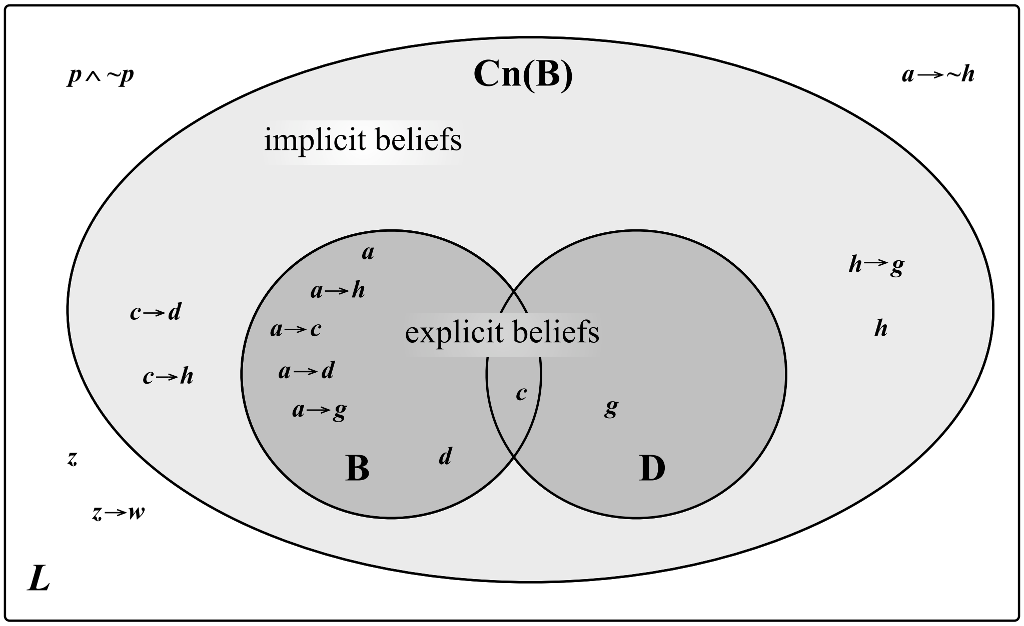

Figure 1.1: A diagram showing explicit vs. implicit beliefs. Explicit beliefs are the beliefs explicitly written in the darkly shaded area consisting of the belief base B and the explicitly derived beliefs D: $B \cup D$. The implicit beliefs are any beliefs that would be in the lightly shaded area (and *not* in the darkly shaded area) made up of $Cn(B) \setminus (B \cup D)$—this set is infinite, so only a small subset is included in this diagram. In this example, the system has discovered that the base belief c is also derivable; but, the only other belief it has derived is g.

explicitly derived beliefs D are in the darkly shaded area, and the area for implicit beliefs is lightly shaded inside $Cn(B)$. I further refer to the makeup of a system (the details that define it, such as the current belief base) as the *state* of the system—also called its *belief state* or *knowledge state*.

Implemented systems strive to be sound, but are rarely complete—only systems that have relatively small bases and simple logics can hope for completeness. This dissertation centers on accepting that a system is incomplete and being ready to adjust in the case where an increase in knowledge (either through acquiring new knowledge or deriving new inferences) exposes decisions based on lesser knowledge as sub-optimal (in hindsight).

A system that cannot reason completely cannot guarantee that a set A of base beliefs underlying some derived belief p is a p-kernel; because, it cannot guarantee that p is *minimally* derived from A (i.e. some proper subset of A may implicitly derive p). However, any sets believed to minimally derive p are called p's *origin sets* [Martins & Shapiro1983, Martins & Shapiro1988]—e.g., if A explicitly derives p, and no $A' \subsetneq A$ explicitly derives p, then A is an origin set for p.

A *minimally inconsistent* subset of a set A is a subset of A that is inconsistent and contains no *proper* subset that is inconsistent: i.e., S is a minimally inconsistent subset of A if and only if $S \subseteq A$, $S \vdash \perp$, and $(\forall S' \subsetneq S) : S' \nvdash \perp$. Therefore, a minimally inconsistent set could also be called a $\perp$-kernel (or falsum-kernel). A nogood is not *necessarily* minimal.

A *base* proposition (or belief) is "believed" if the system accepts it without need for derivational support (considers it an *asserted* base belief; has it in the belief base). It becomes "disbelieved" if it is removed from the base (unasserted); this is *not* the same as believing its negation. A derived belief p is "believed" (is in the belief *space*) if it is derivable from the current base (i.e., only if there is some p-kernel in the current base). A belief may be both a base belief and a derived belief.

Although I will introduce belief change operations in detail in the next chapter, I include a brief description of some key terminology here. *Expansion* is the operation of adding a belief (to a base or theory) with no regard to consistency maintenance (essentially, the union of that belief with the existing beliefs—followed by deductive closure if a theory). *Contraction* of a belief base or theory by some belief results in the removal or elimination (also called the *retraction*) of that belief from the belief *space*—resulting in a new set of beliefs from which the retracted belief is *not* derivable. *Consolidation* is an operation for removing existing contradictions from a belief *base*, resulting in a base that is a consistent subset of the original base.

In this dissertation, the term "belief" does *not* refer to a weakened assertion or a non-fact—as opposed to facts that might be designated as untouchable during operations of belief change. I assume that *all* beliefs are potentially removable (with the *possible* exception of tautologies) .

1.3 Motivations

1.3.1 Implementing KR&R Requires a Departure from the Ideal

As computers are used to store more and more information, there is a strong need to improve the way that information is both represented and used. Whether modeling real world situations or concepts in an agent's mind, the information that is to be stored in (and reasoned about by) a computer must be converted into a representation that is, in most cases, a simplified version of the information.

There are exceptions to the above statement. Storing a social security number (SSN) in a database, for example, does not alter it in any way. But consider that a SSN may be just one of many items of data stored in the database, and a set of that data (SSN, date of birth, name, address, etc.) represents a person. Even though the SSN is accurately represented, the person is not—and *cannot* be—fully represented. So, we must first accept that the representations of information cannot be perfect.

The next reality check is that the ways that computers are programmed to use this information are also not perfect. Although there are many cases where computers can improve on the mental computations of humans by manipulating information faster and more accurately than we can, there are also reasoning techniques that humans use that cannot be (or have not yet been) captured in a KR&R system.

Even when the beneficial aspects of human reasoning are implemented in a computational system, that implementation is a representation of the way that humans reason and inherently must be a simplification of the process. If you doubt that, try writing down exactly how to ride a bike or cook a favorite recipe, then observe someone following your directions. There will always be some subtle nuance of the process that cannot be captured on paper. How much harder is it, then, to program the subtle ways that we reason—

especially when we often are unaware of *what* we are doing, much less *how* we are doing it.

Just as a system cannot attain perfection when representing and reasoning with information, it may also fall short of perfection when *revising* a collection (or set) of information—adding, changing, and removing information, as well as detecting and dealing with contradictions.

The underlying motivation for this research is the goal of improving the performance of a real world, non-ideal, implemented system. This required the development of new terminology to discern the subtle differences in non-ideal states, new operations of belief change, and the development of an *anytime algorithm*—an algorithmic design concept that is favored in implemented systems. The anytime algorithm was introduced in [Dean & Boddy1988]; and the many aspects and benefits of an *anytime* algorithm will be discussed later (in Section 4.5.1)— though, essentially, it provides an ever-improving result as it progresses, even if it does not run to completion.

1.3.2 Specific Observations Motivating this Research

Belief change research often assumes an ideal reasoning agent

Much of the research on belief change defines the belief change operations as they would be used by an *ideal agent* – an agent that has an unlimited memory and is capable of instantaneous and perfect reasoning. The belief change operations defined in such cases describe an ideal goal that a real-world (implemented) system can strive to approach in spite of its limitations—limitations such as (1) reasoning takes time, (2) memory is not infinite, and (3) the system's reasoning capabilities may be imperfect.

Although these limitations prevent ideal reasoning, the terminology used when defining the ideal operations is not sensitive to these limitations. For example, the term "inconsistent" would apply equally to a system whose belief space contains a *known* contradiction and a system that is unaware of some implicit contradiction. There is a need for enhancing the terminology to encompass the more subtle states of a non-ideal system.

The Limited Effects of Removing a Belief

In many cases, humans change their minds without realizing how that change will affect their belief space. In the specific case of removing some belief p (typically, because a stronger belief conflicted with it), there may exist previously retracted beliefs that were removed solely because they conflicted with p. In hindsight, the removal of p eliminates the need for the removal of those other beliefs, and they could possibly return to the belief space without raising a contradiction.

The research in this dissertation formalizes and refines an algorithm for just this kind of hindsight repair. It has a belief base optimizing effect that mimics the process of sending a jury to the jury room to review *all* the information from a trial in order to produce an optimal decision on the case. This hindsight repair is also closely related to the topics mentioned in the next two sections: adherence to the postulate of Recovery and the negative effects of operation order.

Coherence vs. Foundations Theory and the Recovery Postulate

There are various postulates used to describe belief change operations for the two belief change approaches: foundations and coherence. The coherence postulates presented in the most cited belief change reference, [Alchourrón, Gärdenfors, & Makinson1985], also contain the most controversial postulate—the Recovery postulate for contraction. Although these postulates will be discussed in detail in Chapter 2, I mention the Recovery, because it motivates this research.

The Recovery postulate states that if an operation to retract some belief p is immediately followed by an operation of expansion by p, then the belief space existing *prior* to those two operation should be contained in the belief space that results *from* the two operations. Essentially, it says that expansion by a belief should undo the contraction by that same belief—in the sense that any beliefs removed from the theory due to the contraction are returned to the theory. The new theory is a proper superset of the original theory *only* if the belief being added in expansion was not in the original theory.

There are two key elements that are both needed for a system's adherence to Recovery. One is that the contraction operation remove as little as possible to achieve the retraction—e.g., removing *all* beliefs would be one way to perform contraction, but it would be too extreme a reduction of the belief space. The second requirement is deductive closure both before and after the belief change operations—and this is why Recovery holds for theory contraction of the coherence approach but not for the belief base contraction used in the foundations approach.

The background needed for an example of this is too extensive for this chapter. There is a full discussion of the controversy surrounding the Recovery postulate and a defense of Recovery are presented in Chapter 2, Section 2.3. I feel that the foundations approach would benefit from an improved adherence to Recovery.

Non-prioritized Addition and the Effects of Operation Order

A majority of the belief-change literature requires prioritized addition—where a belief added to a base or theory (*along with* attention to consistency maintenance) is *required* to survive the addition and be in the resulting belief space; the belief being added has the *highest priority*. Prioritized-addition is fine for recency-dependent information, where the most recent information rules out all contradictory information (example: the traffic light is red vs. the light is green). In this case, operation order is essential to obtaining the correct belief space.

Recency-dependence has multiple levels of granularity, however, and there are cases where prioritized-addition is inappropriate. One example is when information is collected from multiple sources with varying credibilities, and the order the information enters the system has no relationship to the credibility of the source. In this case, non-prioritized addition would be more appropriate—where information is retained based on its credibility (more so than its recency), and new information that conflicts with existing beliefs is not *guaranteed* to be in the newly revised belief space. The system might choose to disallow the acceptance of some weaker new belief—preferring, instead, to retain the conflicting beliefs that are considered more credible and thus maintain its current belief space (although, depending on the system, the belief *state* might change to reflect that the new information was considered and rejected).

It would seem that the order of the addition operations should not affect the outcome of the belief space—the more credible beliefs should survive, and the less credible are disbelieved. However, this is *not* true. In the case of base theory change with consistency maintenance, the lack of adherence to the Recovery postulate results in the order of non-prioritized additions affecting the makeup of the resulting base. If one considers all the possible ways to re-order incoming information and the various bases that would result from the different input orders, then, typically, one or more of the resulting bases are preferred over the others. Therefore, the less preferred bases are sub-optimal.

I refer to this kind of sub-optimality as a *negative effect of operation order*. Improving adherence to Recovery goes hand in hand with eliminating the negative effects of operation order.

1.4 Research Contributions

Contributions of this research include:

- The introduction, definition and formalization of the operation of *reconsideration* —an operation of hindsight, belief base optimization
- An analysis of how reconsideration offers an aspect of Recovery for belief bases
- A discussion of how reconsideration eliminates the negative effects of the order of non-prioritized belief change operations

- An efficient, anytime algorithm called *dependency-directed reconsideration* (DDR) that can be used to implement reconsideration in a truth maintenance system
- An expression of DDR as a constraint satisfaction problem
- A formalization of a non-ideal belief space called a *deductively open belief space* (DOBS) along with the terminology that applies to a DOBS
- An discussion of how DDR can be applied to a DOBS
- A data structure and algorithm that allows a computationally efficient way to maintain a collection of minimal sets—useful for maintaining a collection of origin sets for an individual belief as well as maintaining the collection of minimal nogoods in a set of beliefs.

A Deductively Open Belief Space—A Brief Introduction to Dispel Misconceptions

Although Chapter 5 provides a complete formalization of a deductively open belief space (DOBS), I briefly discuss it here, to insure that the reader does not assume that a DOBS is merely a belief base. It is distinctly different from a base.

First presented in [Johnson & Shapiro2000a] and later refined in [Johnson & Shapiro2001], a DOBS is a belief space that is by definition *not guaranteed to be deductively closed*. It contains some core set of beliefs (a belief base) and the beliefs that *have been* derived from them *so far*—this recognizes the fact that reasoning takes time. For example, a DOBS might include the beliefs p and $p \rightarrow q$ without the derivable proposition q.

Although a DOBS is defined in terms of the foundations approach to belief change with a set of core base beliefs, there is a marked difference between the concept of a DOBS and that of a belief base. Any belief base has a belief space that is its *deductive closure*. By contrast, a DOBS is a finite belief space that is made up of only *explicit* beliefs (both base beliefs and explicitly derived beliefs); and this DOBS can increase in size when additional deductions add to its set of explicitly derived beliefs. This increase in the number of beliefs in a DOBS can happen even as the base remains unchanged, because the set of explicitly derived beliefs is increasing in size as the system reasons from its base.

However, the DOBS for some belief base B *does* form a finite set of beliefs whose deductive closure is the same as the deductive closure of B, itself—thus, technically, both the DOBS and B are belief bases for the same deductively closed belief space. As stated before, I favor Hansson's description of a belief base, however, as referring to some core set of beliefs which are asserted with *independent standing*—as opposed to *any* finite set of beliefs whose closure is a predetermined belief space [Nebel1989]. Therefore, I do *not* refer to the DOBS, itself, as a belief base. See [Hansson1999] for a more complete discussion of various belief base approaches.

1.5 Dissertation Outline

The remainder of this dissertation gradually builds a formalization for a knowledge state that can perform reconsideration using the algorithm for dependency-directed reconsideration (DDR). After introducing related work in Chapter 2, I define reconsideration and DDR in Chapters 3 and 4. Chapter 5 includes a formalization of the DOBS and discusses how DDR can be applied to a DOBS. An extended case study is presented in Chapter 6 that illustrates the anytime features of DDR when performed on a DOBS. Chapters 7 and 8 contain conclusions and future research topics, respectively.

The proofs for the theorems in Chapter 4 are extensive, and most of them are located in the Appendix. There is also an index provided to assist with recalling terminology that is used across several chapters.

Chapter 2

Background and Related Work

2.1 Introduction

2.1.1 Purpose of This Chapter

This chapter introduces the background and related research that this dissertation will build upon or be compared to. This includes a discussion of the extensive terminology used by belief change researchers and how I will be using that terminology in this dissertation. It also includes a brief overview of belief change systems which will be mentioned in later chapters.

2.1.2 Belief Change vs. Belief Revision

There are many terms used to describe the research field that encompasses the theories of how collections of beliefs should be maintained. I referred to two of these terms in Chapter 1: theory belief change and base belief change, the distinguishing parts being a deductively closed set of beliefs, called a "theory," and a finite set of beliefs, called a "base".

This field has been commonly referred to as the field of "belief revision" (often abbreviated as BR).

However, there is also a *specific* belief change operation called “revision” (which I will discuss later in this chapter), and this leads to potential confusion when discussing a belief revision operator that is *not* “revision”.

Other work in this field refers to BR as the field of ”belief change” or research about “knowledge in flux”. I will use the term “belief change” to describe the overall concept of any operation that changes the state of the beliefs. Unfortunately, the abbreviation “BC” is never used in the literature. Therefore, I will use “BR” as the abbreviation for “belief change”—knowing that it is not ambiguous with the specific operation of “revision”.

2.1.3 Merging Factions of Belief Change

When belief change research was in its early stages, there were two distinct research groups working on belief change theories: philosophers (of which [Alchourrón, Gärdenfors, & Makinson1985] is the most referenced work) and computer scientists—most notably the truth maintenance system (or TMS) researchers [McAllester1978, Doyle1979, Martins & Shapiro1983, de Kleer1986]. Differences between these groups were easy to see in their early years, and might at first glance be distinguished by the terms theoretical vs. implementation. This would be dangerous, however, because it incorrectly implies that the philosophers' theories could not be implemented and that the computer science implementations had no theory behind them.

Hansson makes a better distinction in [Hansson1991a] when he refers to belief change research as using “sentential representations” for beliefs, and then divides these representations into the two categories of “pure” vs. “composite”. Beliefs in pure representations are solitary entities, whereas beliefs in the composite representations were each paired with additional information. The AGM field of research (named for the 1985 paper of Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson) falls into the pure category.

I present the basics of their work in this chapter along with other relevant work with pure sentential belief base representations [Hansson1991a, Hansson1999, Williams & Sims2000, Wassermann1999], and I refer to this collection of researchers as the "AGM/Base" research group.

Regarding the composite group, Hansson divided it into two further categories: "probabilistic" and "track-keeping". The early research on TMSs falls into the latter category, because each belief is paired with the *justifications for* [McAllester1978, Doyle1979] or *origins of* [Martins & Shapiro1983, de Kleer1986] that belief—the reasons for having that belief. This was a way of keeping track of the derivation history of a belief.

The probabilistic group pairs each belief (including both base beliefs and derived beliefs) with a quantitative measure of the likelihood of whether the belief is true or accurate. Probabilistic belief change research will not be discussed in this dissertation.

This dissertation achieves a partial merging of the AGM/Base and TMS factions by

1. presenting an operation for Base belief change that allows an improved adherence to AGM theory constraints;

2. providing an anytime[1] algorithm to implement into a TMS the operation mentioned in (1) and that improves the belief change performance of the TMS to more closely adhere to the AGM/Base constraints;

3. providing a formalization for an *implemented* KR&R system along with additional terminology that can be used to:

    - alter the guidelines and constraints that were developed for *ideal* agent belief change (from the AGM/Base group) in order to formalize implementable guidelines and constraints for belief

[1]The many aspects and benefits of an *anytime* algorithm will be discussed later (in Section 4.5.1), but it provides an ever-improving result as it progresses—even if it does not run to completion.

change by *non-ideal* reasoning agents;

- provide additional ways to describe and compare implemented systems that differ in ways that cannot be described using the ideal agent terminology of the traditional AGM/Base research.

2.1.4 Agents: Ideal vs. Resource-Bounded

Belief change research has produced many theories on how to perform belief change operations, but they always have to consider the reasoning agent—i.e., the theories need to take into account the reasoning abilities and limitations of the reasoning system. Some key qualities of a reasoning system are completeness and soundness.

- Sound reasoning insures that any belief that is derived from a set of beliefs is truly entailed by those beliefs. This is fairly straightforward in a monotonic system.
- Complete reasoning insures that any belief that is entailed by a set of beliefs (A) is derivable from that set. This becomes computationally expensive or impossible as the set A increases in size or as the logic becomes more complex (e.g., FOPL with functions).

.

When working with pure theory, researchers can assume an ideal agent—one that can reason soundly. completely, and instantly, and has infinite space for retaining beliefs. This type of agent can consider the deductive closure of a set of beliefs whenever it has to determine the consistency of that set or remove beliefs to maintain consistency.

In a real-world implementation, resources are limited—agents take time to reason, and their memory space is finite. The theoretical deductive closure of a set is usually infinite, so a resource-bounded agent [Wassermann1999] cannot consider the deductive closure. Reasoning in an implemented system also takes time, so the system is likely to have implicit beliefs it is not yet aware of.

For this dissertation, I am not examining an agent's limits of logical ability [Wassermann1999] as another factor contributing to some implicit but unknown inconsistency. However, regarding implemented KR&R systems, I do recognize that the theoretical logic (as described by a system designer) and the logical abilities of the system (as they have been implemented) might differ—with the latter falling short of the ideal design. This is a prime example of why even the most simple KR&R system cannot guarantee correct information at all times.

2.2 AGM/Base Research

2.2.1 Foundations and Coherence Approaches

Implemented systems typically follow a foundations approach [Doyle1979, Gärdenfors1992, Hansson1993] where the belief space is made up of a base set of beliefs, which are self-supporting (i.e., they are asserted without having to be derived), and beliefs derived from (i.e., supported by) those base beliefs. Belief change operations alter the presence of beliefs in the base set (e.g., beliefs are added to and/or removed from the base), and the derived beliefs that lose their support are no longer believed.

By contrast, a coherence approach [Alchourrón, Gärdenfors, & Makinson1985] allows previously derived beliefs to remain as long as they are not responsible for an inconsistency, even if their support is lost. Simplistically, the belief "responsible" for an inconsistency is the belief that should be chosen for retraction to resolve that inconsistency. Although belief maintenance is much more complicated than that, the focus of this dissertation is to improve belief change results in a non-ideal system, based on the information it already has, rather than focusing on which beliefs should be removed to maintain consistency. Although our DOBS formalization follows the foundations approach, I offer a coherence version of the DOBS in Section 5.2.2 that would allow a coherence approach to DOBS belief change.

In the following subsections, I discuss some belief change integrity constraints and the rationality postulates for various belief change operations. The belief change operations discussed are expansion, contraction, revision, kernel-contraction, kernel consolidation, and kernel semi-revision. As an overview:

- expansion is the addition of a belief to an existing set of beliefs with no attempt to maintain consistency;
- contraction is the removal of a belief from a set of beliefs (including its closure);
- revision is a form of addition combined with consistency maintenance;
- consolidation is an operation on a finite set of beliefs that eliminates inconsistency.

Note that any form of inconsistency elimination or avoidance may require the contraction of one or more beliefs that are involved in deriving the inconsistency; so expansion is the only operation that is *guaranteed not* to include the removal (or contraction) of some belief.

2.2.2 Integrity Constraints for Belief Change Operations on a Knowledge Space

Gärdenfors and Rott [1995] list four integrity constraints (paraphrased below) for a deductively closed knowledge space (a.k.a. theory) undergoing belief change:

1. a knowledge space should be kept consistent whenever possible;

2. if a proposition can be derived from the beliefs in the knowledge space, then it should be included in that knowledge space;

3. there should be a minimal loss of information during belief change;

4. if some beliefs are considered more important or entrenched than others, then any belief change operation that removes beliefs should remove the least important ones.

These constraints are the basic guidelines for belief change in the AGM/Base research. Constraint 1 clearly suggests that inconsistencies should be avoided. Constraint 2 favors deductive closure and complete reasoning—everything entailed by a set should be in that set. Constraint 3 reminds us that any beliefs removed to insure consistency should be directly related to some inconsistency—if unconnected to an inconsistency, a belief should remain in the base. Constraint 4 suggests removing weaker or less credible beliefs rather than those that are more credible (if given the choice).

How to satisfy both Constraints 3 and 4 is an ongoing debate. A typical consideration is what to do when many weak beliefs overrule a single, more credible (up till now, that is) belief.

These constraints are discussed in more detail in Chapter 5, which introduces the formalization of a DOBS, and are altered for use as guidelines for a DOBS system and for implementing automated belief revision in an implemented KR&R system.

2.2.3 AGM Operations: Expansion, Contraction, and Revision

The AGM belief change operations (expansion, consolidation, and revision) are performed on a deductively closed knowledge space (also called a theory) K. Therefore, $K = Cn(K)$.

Expansion

The belief space K expanded by the proposition p is written as $K+p$ and defined as $K+p =_{def} Cn(K \cup \{p\})$.

Note that the resulting space can be inconsistent, and (for classical logic) all inconsistent belief spaces for some given language are identical—consisting of the set of *all* possible formulas in that language, due to deductive closure. Inconsistent belief spaces are indistinguishable due to their deductive closure—they are the same belief space: $K_\perp = \mathcal{L}$. Therefore, all *theory* belief change operations are focused on preventing inconsistency whenever possible.

Contraction

The belief change operation of contraction is not directly defined but, instead, is constrained by rationality postulates. For contraction of a belief space K, by the proposition p (written as K–p), the six basic AGM postulates are [Alchourrón, Gärdenfors, & Makinson1985]:

(K–1) K–p is a belief space *Closure*

(K–2) $K\text{–}p \subseteq K$ *Inclusion*

(K–3) If $p \notin K$, then $K\text{–}p = K$ *Vacuity*

(K–4) If not $\vdash p$, then $p \notin K\text{–}p$ *Success*

(K–5) $K \subseteq (K\text{–}p) + p$ *Recovery*

(K–6) If $\vdash p \leftrightarrow q$, then $K\text{–}p = K\text{–}q$ *Extensionality*

K–1 states that the result of contraction is a deductively closed belief space.

K–2 states that no new beliefs are added as a result of contraction of a belief space—the belief space resulting from contraction is included in (i.e., is a subset of) the original belief space.

K–3 states that contracting a belief space by some belief that is *not* in the belief space should result in no change to that belief space.

K–4 states that contracting a belief space by some belief p should result in a belief space that does not entail p, provided that p is not a tautology.

K–5 states that any beliefs lost (i.e., that become disbelieved) due to contraction by a belief can be recovered (i.e., returned to the belief space) through a subsequent expansion by that same belief.

K–6 states that if, for two beliefs, p and q, neither can exist in a belief space without the other, then the belief space resulting from contraction by one of the beliefs (p) is equal to the belief space resulting from contraction by the other belief (q).

Recovery (K–5) is the most controversial of the contraction postulates. Although adherence to Recovery is not necessarily unwanted, the argument concerns whether it should be a *primary* constraint of a contraction operation. A detailed discussion of the Recovery postulate is presented in Section 2.3.

Revision

The belief change operation of revision is not directly defined. Like contraction, it is constrained by rationality postulates. The six basic AGM postulates for revision of a belief space K, by the proposition p (written as $K*p$) are [Alchourrón, Gärdenfors, & Makinson1985]:

(K∗1) $K*p$ is a belief space *Closure*

(K∗2) $p \in K*p$ *Success*

(K∗3) $K*p \subseteq K+p$ *Expansion 1*

(K∗4) If $\neg p \notin K$, then $K+p \subseteq K*p$ *Expansion 2*

(K∗5) $K*p = K_\perp$ only if $\vdash \neg p$. *Consistency preservation*

(K∗6) If $\vdash p \leftrightarrow q$, then $K*p = K*q$ *Extensionality*

K∗1 states that the result of revision is a deductively closed belief space.

K∗2 states that revision by a belief results in a belief space that contains that belief.

K∗3 states that expansion of K by some belief p results in a superset of the belief space resulting from revision of K by p.

K$*$4 states that revision of K by a belief (p) that is not inconsistent with K results in a superset of the belief space resulting from expanding K by p.

K$*$5 states that revision of K by some belief p results in a consistent belief space unless p, itself, is inconsistent.

K$*$6 states that if, for two beliefs, p and q, neither can exist in a belief space without the other, then the belief space resulting from revising K by one belief (p) is equal to the belief space resulting from revising K by the other belief (q).

Note that the operation of AGM revision requires that the belief being added *must* be contained in the resulting belief space. This has prompted the belief change community to refer to this form of addition as *prioritized revision*—where the new belief has highest priority (i.e., it must survive any operation to retain consistency, even if it is *itself* inconsistent), and the remaining beliefs may be ordered (or not) as long as that ordering does not place any of them above the incoming belief. By contrast, *non-prioritized* belief change [Hansson1999] does not mean that beliefs are *not* ordered in any way, but merely that highest priority is not always assigned to an incoming belief. I present some of Hansson's non-prioritized belief change operations later—cf. page 31.

The inter-relation between revision and contraction is shown by the following two well-known identities:

- $K*p = (K-\neg p) + p$ *Levi Identity*
- $(K-p) = K*\neg p \cap K$ *Harper Identity*

The Levi Identity shows that revision by p is equivalent to contraction by $\neg p$ followed by expansion by p. This is called *internal revision* and will be contrasted with *external revision* later in this chapter (cf. page 28). The Harper Identity shows that contraction by p can be achieved by revising a theory by $\neg p$ and then removing any beliefs that were not in the original theory. These identities are named after Isaac Levi and William Harper, respectively, but no citations accompany them in the literature.

2.2.4 Hansson's Postulates for Expansion, Contraction, and Revision, of a Belief Base

Hansson [1991a,1993,1999] proposes AGM-style postulates that can apply to belief base revision, where a belief space BS is revised by performing a belief change operation on some finite belief base H, where $Cn(H) = BS$. Per the foundations approach recommended by Hansson and used by the TMS researchers, I assume that this base is not just some arbitrary base whose closure is BS. I assume it is the basis for the construction of the belief space BS.

Hansson's postulates were originally written for contraction and revision of a belief base by a *set* of propositions. Below are rewritten versions, altered for contraction (and revision) of a belief base, H, by a *proposition* p (as done in [Gärdenfors & Rott1995]), where Z is a set of propositions in $\mathcal{L}$ and q is a proposition in $\mathcal{L}$.

Though expansion can be directly defined, contraction and revision are constrained by postulates. I present a brief, though incomplete, description here to aid in understanding the postulates below. Contraction of H by p (H–p) includes removing elements of H to form H' such that the belief *space* for H'—($Cn(H')$)—no longer contains p. Revision of H by p ($H*p$) means adding p to H to form H' such that $p \in H'$ and H' is consistent (if possible)—this may require removing beliefs so that $H \setminus H' \neq \emptyset$.

Expansion

Expansion for a belief base is similar to that for a theory without the deductive closure. The belief base H expanded by the proposition p would be written as $H+p =_{def} H \cup \{p\}$.

Note that the resulting base can be inconsistent. However, unlike a deductively closed belief space, it is possible to distinguish between finite belief bases that are inconsistent. The closure of an expanded base is equivalent to the closure of its belief space: If $Cn(H) = K$, then $Cn(H + p) = K + p$.

Contraction

Contraction of a belief base H by a proposition p (H–p):

(H–1) $H–p \subseteq H$ *Inclusion*

(H–2) If not $\vdash p$, then $p \notin Cn(H–p)$ *Success*

(H–3) If $q \in H \backslash H–p$, then there is some H' such that $H–p \subseteq H' \subseteq H$ and $p \notin Cn(H')$, but $p \in Cn(H' \cup \{q\})$ *Relevance*

(H–4) If it holds for all subsets H' of H that $p \notin Cn(H')$ if and only if $q \notin Cn(H')$, then $H–p = H–q$ *Uniformity*

(H–5) If not $\vdash p$ and each element of Z implies p, then $H–p = (H \cup Z)–p$ *Redundancy*

H–1 states that the base resulting from a contraction is a subset of the original base.

H–2 states that contracting a belief base by some belief p should result in a belief base that does not entail p, provided that p is not a tautology.

H–3 states that any belief removed during a contraction operation must contribute in some way to the derivation of that belief.

H–4 states that if, for two beliefs, p and q, neither can exist in the closure of a subset of the base without the other, then the belief base resulting from contraction by one of the beliefs (p) is equal to the belief base resulting from contraction by the other belief (q).

H–5 states that, as long as p is not a tautology, any set Z made up of beliefs that entail p can be added to the base before contracting by p, and the result will be the same as contraction performed without adding Z (all the beliefs in Z will be removed from the base).

Base contraction cannot adhere to Recovery. As mentioned before, a detailed discussion of the Recovery postulate is presented in Section 2.3. The research presented in this dissertation includes an alternate Recovery formulation that belief change operations for bases adhere to in more preferred situations. This formulation and the discussion of base belief change adherence is presented in Section 3.3.

Revision

Revision of a belief base H by the proposition p ($H*p$ whose belief space is $Cn(H*p)$):

(H∗0) If not $\vdash \neg p$, then $H*p$ is consistent. *Consistency*

(H∗1) $H*p \subseteq H \cup \{p\}$ *Inclusion*

(H∗2) If $q \in H \setminus H*p$, then there is some H' such that $H*p \subseteq H' \subseteq H \cup \{p\}$, H' is consistent, and $H' \cup \{q\}$ is inconsistent *Relevance*

(H∗3) $p \in Cn(H*p)$ *Success*

(H∗4) If for all $H' \subseteq H$, $H' + p$ is inconsistent if and only if $H' + q$ is inconsistent, then $H \cap H*p = H \cap H*q$ *Uniformity*

(H∗5) If not $\vdash \neg p$ and each formula in Z is logically inconsistent with p, then $H*p = (H \cup Z)*p$ *Redundancy*

H∗0 states that, as long as $\neg p$ is not a tautology, revision by p will produce a consistent base.

H∗1 states that expansion of the base H by some belief p results in a superset of the belief base resulting from revision of H by p.

H∗2 states that any belief removed during revision of H by a belief p must contribute in some way to an inconsistency derivable from H expanded by p.

H$*$3 states that revision by a belief results in a belief base that contains that belief.

H$*$4 states that if, for two beliefs, p and q, any subset of the current base is inconsistent when expanded by p if and only if it is also inconsistent when expanded by q, then the belief base resulting from revision by p and the belief base resulting from revision by q share the same elements of the original base (i.e., the beliefs removed to maintain consistency when revising by p are the same beliefs as those removed when revising by q).

H$*$5 states that, as long as p is not a tautology, any set Z made up of beliefs that are logically inconsistent with p can be added to the base before revising by p, and the result will be the same as revision performed without adding Z (all the beliefs in Z will be removed from the base).

Base revision can also conform to the Levi Identity: $H*p = (H{-}\neg p) + p$. As with theory revision, this is referred to as internal revision (cf. page 24). As can be seen from H$*$3, this is also a form of prioritized revision.

Additionally, base revision can be constructed so that to conform with a reversed version of the Levi Identity: $H*p = (H+p){-}\neg p$ [Hansson1993]. This is called *external revision*. Reversing the Levi Identity does not work for deductively closed belief spaces, because, if the expansion process makes the space inconsistent, the result is $K_\perp$.

2.2.5 Hansson's Belief Base Postulates for Belief Change Operations Using Kernels

This section briefly reviews the belief change operations of kernel contraction, kernel consolidation, and kernel semi-revision of a finite belief base B [Hansson1994, Hansson1999].

Kernel Contraction

The kernel contraction [Hansson1994] of a base B by a belief p is written as $B \sim p$.

For this dissertation, $B \sim p$ is the kernel contraction of the belief base B by p (retraction of p from B) and, although constrained by several postulates, is basically the base resulting from the removal of at least one element from each p-kernel (def 1.2.1 on page 4) in B—unless $p \in Cn(\emptyset)$, in which case $B \sim p = B$. Selecting which elements of the p-kernels to remove is done by a decision function that is called an *incision function* because it determines what p-kernel elements will be "cut out" of the existing base to achieve the retraction of p.

The postulates constraining kernel contraction are formalized in Hansson's theorem on kernel contraction:

Theorem 2.2.1 *[Hansson1994] The operator $\sim$ for B is a kernel contraction if and only if it satisfies:*

1. *If $\nvdash p$, then $B \sim p \nvdash p$;* *(Success)*
2. *$B \sim p \subseteq B$;* *(Inclusion)*
3. *If $q \in B$ and $q \notin B \sim p$, then $(\exists B') : B' \subseteq B$, $B' \nvdash p$ and $B' \cup \{q\} \vdash p$;* *(Core-retainment)*
4. *$(\forall B' \subseteq B)$: If $B' \vdash p$ if and only if $B' \vdash q$, then $B \sim p = B \sim q$.* *(Uniformity)*

This theorem says that the kernel contraction of B by p results in a subset of B (Inclusion) that cannot derive p, unless p is a tautology (Success). Additionally, any beliefs removed must be elements of a p-kernel (Core-retainment), thus the core beliefs in B that do *not* contribute to the derivation of p must remain in the base after contraction is performed. The Core-retainment postulate is similar to the Relevance postulate for base contraction discussed in Section 2.2.4. Inclusion and Success are identical to those for base contraction.

Uniformity states that an incision function should base its selection of beliefs to be removed on the choices available and should be consistent whenever presented with those same choices.

Example 2.2.2 Given $B = \{r, r \rightarrow q \wedge p\}$, there are three possible results of $B \sim p$ and the same three possible results for $B \sim q$: $\{r\}$ or $\{r \rightarrow q \wedge p\}$ or $\{\}$. Uniformity says that given these choices, the incision function should not make a different choice for $B \sim p$ than it does for $B \sim q$. The case where $B \sim p = \{r\}$ and $B \sim q = \{r \rightarrow q \wedge p\}$ violates Uniformity.

Uniformity for kernel contraction is identical to the Uniformity postulate for base contraction discussed in Section 2.2.4 and is similar in concept (though not identical) to the AGM Extensionality postulate discussed in Section 2.2.3.

Kernel Consolidation

Consolidation [Hansson1994] is the elimination of any inconsistency and is defined for belief *bases* only. Recall that, for deductive closure in classical logic, any inconsistent belief *theory* is $K_\perp$, making inconsistency removal a non-issue—theory belief change operations focus on preventing inconsistencies from occurring.

$B!$ —the kernel consolidation of B [Hansson1994]—is the removal of at least one element from each $\perp$-kernel[2] in B s.t. $B! \subseteq B$ and $B! \nvdash \perp$. This means that kernel consolidation is the kernel contraction of an inconsistency ($B! = B \sim \perp$, where $\sim$ is the operation of kernel contraction), also referred to as *contraction by falsum*, so it adheres to the same postulates as kernel contraction (except that Uniformity is not applicable). However, specific postulates are mentioned in Hansson's kernel consolidation theorem:

Theorem 2.2.3 *[Hansson1997] An operation "!" is an operation of kernel consolidation if and only if for all sets B of sentences:*

1. *$B!$ is consistent;* *(Consistency)*
2. *$B! \subseteq B$;* *(Inclusion)*
3. *If $q \in B$ and $q \notin B!$, then $(\exists B') : B' \subseteq B$, $B' \nvdash \perp$ and $B' \cup \{q\} \vdash \perp$.* *(Core-retainment)*

[2]Recall from Section 1.2, page 1.2 that a $\perp$-kernel is a minimally inconsistent set.

The Consistency postulate for consolidation is equivalent to the kernel contraction postulate of Success: $Cn(B!) \nvdash \bot$. Inclusion and Core-Retainment are the same as for kernel contraction.

Kernel Semi-Revision

The belief change operation of *kernel semi-revision* [Hansson1997] is a form of non-prioritized belief change. This is in contrast to *prioritized* belief revision of B by a ($B * a$), where the belief currently being added is given highest priority, until the *next* prioritized addition, and which must adhere to the Success postulate: $a \in (B * a)$—cf. page 24. With semi-revision, if the incoming belief raises an inconsistency, the KR&R system can *choose* whether to (1) reject the incoming belief or (2) accept it and remove some previously held base belief(s) to obtain consistency.

The kernel semi-revision of the base B by the belief a is written as $B +_! a$, and it is defined as $B +_! a =_{def} (B+a)!$, which is equivalent to $(B \cup \{a\})!$. The operation $+_!$ is an operator of kernel semi-revision IFF [Hansson1997]

1. $Cn(B +_! a) \nvdash \bot$ (*Strong Consistency*)
2. $B +_! a \subseteq B+a$ (*Inclusion*)
3. $p \in (B \setminus B +_! a), \Rightarrow (\exists X) : X \subseteq B+a,\ Cn(X) \nvdash \bot,$ and $Cn(X+p) \vdash \bot$ (*Core-retainment*)
4. $(B+a) +_! a = B +_! a$ (*Pre-expansion*)
5. If $p, q \in B$, then $B +_! p = B +_! q$ (*Internal Exchange*)

The key concept is that $B +_! a$ is equivalent to $(B+a)!$—expansion by a followed by consolidation—where the consolidation operation might actually *remove a*. The first three postulates follow directly from those for consolidation. Note: Strong Consistency also insures that any inconsistency previously existing in B, whether related to the incoming belief or not, will also be resolved by semi-revision. Regarding Pre-expansion, since $(B+a) +_! a = ((B+a)+a)!$ (by definition), and $((B+a)+a) = ((B \cup \{a\}) \cup \{a\}) =$

$(B \cup \{a\}) = B + a$, then $(B+a) +_! a \; = \; (B+a)! \; = \; B +_! a$. Lastly, Internal Exchange is obvious when considering that $B +_! p \; = \; B +_! q$ is equivalent to $(B+p)! = (B+q)!$ which is equivalent to $B! = B!$, because $p, q \in B$.

Kernel semi-revision is not possible for deductively closed belief spaces, because the initial expansion might result in the inconsistent set $K_\perp$. Note that $N_\perp = \mathcal{L}$.

2.2.6 Wassermann's Resource-Bounded Theories

Wassermann's formalization for "resource-bounded belief revision" is described *theoretically* in her Ph.D. Thesis [Wassermann1999], and specifically deals with the issue of resource-boundedness. Wassermann defines the concept of "embedded local change"— the belief set being altered during some belief change operation using some proposition p should be some subset of the full base of beliefs that is *relevant* to p (i.e., the elements in the set have some relation to p). This relation could be syntactic, logical, etc., but unrelated propositions should not be considered during the belief change operation. As an example, p would be related to the belief $p \rightarrow q$, and this would result in q also being related to p, though more distantly.

Wassermann also gives an anytime algorithm for determining this subset of beliefs—selecting beliefs that are related by determining those closely related and gradually finding those that are more distantly related. The longer the algorithm runs, the more distantly related the beliefs that are added to the set. After this set is determined, however, the belief change operation must still be performed.

If the set is small enough (and assuming a simple logic like classical logic), traditional belief change operations (ideal operations) can be used successfully. The anytime algorithm can be stopped whenever the set reaches some predetermined maximum limit (to deal with resource limitations regarding memory or computation time). This means that "related" elements outside the selected set are unaffected by the belief change operation.

The research focus is on issues that must be addressed in order to perform AGM/Base belief change on a non-ideal KR&R implementation. Although the goals for this research share similarities with ours, the approach is different and will be compared in Chapter 4 (cf. page 170).

2.2.7 Belief Liberation

Basic Belief Liberation Notation

In this section, I summarize σ-liberation [Booth *et al.*2005].

Belief Liberation operations assume a linear sequence (chain) of beliefs which is called $\sigma = p_1, \ldots, p_n$.[3] The sequence is ordered by recency, where p_1 is the most recent information[4] the agent has received (and has highest preference), and the notation $[[\sigma]]$ is the set of all the sentences appearing in σ.

A Belief Sequence Relative to *K*

In [Booth *et al.*2005], the ordering of σ is used to form the preferred maximal consistent subset of $[[\sigma]]$ iteratively by defining the following: (1) $\mathcal{B}_0(\sigma) = \emptyset$. (2) for each $i = 0, 1, \ldots, n-1$: if $\mathcal{B}_i(\sigma) + p_{(i+1)} \nvdash \perp$, then $\mathcal{B}_{(i+1)}(\sigma) = \mathcal{B}_i(\sigma) + p_{(i+1)}$, otherwise $\mathcal{B}_{(i+1)}(\sigma) = \mathcal{B}_i(\sigma)$. That is, each belief—from most recent to least—is added to the base only if it does *not* raise an inconsistency.

Definition 2.2.4 [Booth *et al.*2005] Let K be a theory and $\sigma = p_1, \ldots, p_n$ a belief sequence. We say σ is a belief sequence relative to K iff $K = Cn(\mathcal{B}_n(\sigma))$.

[3] This sequence is also used in [Chopra, Georgatos, & Parikh2001].

[4] I have reversed the ordering from that presented in [Booth *et al.*2005] to avoid superficial differences when comparing their preference ordering with ours in future chapters. I have adjusted the definitions accordingly.

Removing a Belief q from K

In [Booth *et al.*2005] the operation of removing the belief q is defined using the following: (1) $\mathcal{B}_0(\sigma,q)=\emptyset$. (2) for each $i=0,1,\ldots,n-1$: if $\mathcal{B}_i(\sigma,q)+p_{i+1}\nvdash q$, then $\mathcal{B}_{(i+1)}(\sigma,q)=\mathcal{B}_i(\sigma,q)+p_{(i+1)}$, otherwise $\mathcal{B}_{(i+1)}(\sigma,q)=\mathcal{B}_i(\sigma,q)$. Note that "$\mathcal{B}_n(\sigma)=\mathcal{B}_n(\sigma,\perp)$ and $\mathcal{B}_n(\sigma,q)$ is the set-inclusion maximal amongst the subsets of $[[\sigma]]$ that do *not* imply q." [Booth *et al.*2005]

Given a belief sequence σ relative to K, σ is used to define an operation $\sim_\sigma$ for K such that $K\sim_\sigma q$ represents the result of removing q from K [Booth *et al.*2005]: $K\sim_\sigma q=Cn(\mathcal{B}_n(\sigma,q))$ if $q\notin Cn(\emptyset)$, otherwise $K\sim_\sigma q=K$.

Definition 2.2.5 [Booth *et al.*2005] Let K be a belief theory and $\sim$ be an operator for K. Then $\sim$ is a σ-liberation operator (for K) iff $\sim=\sim_\sigma$ for some belief sequence σ relative to K.

Example [Booth *et al.*2005]. Suppose $K=Cn(p\wedge q)$ and let $\sigma=p\rightarrow q,p,\neg p\wedge\neg q$ be the belief sequence relative to K — where $\neg p\wedge\neg q$ was originally blocked from inclusion in $\mathcal{B}_3(\sigma)$ by the inclusion of the more recent (and more preferred) belief p. Suppose we wish to remove p. We must first compute $\mathcal{B}_3(\sigma,p)$. We have $\mathcal{B}_0(\sigma,p)=\emptyset$, $\mathcal{B}_1(\sigma,p)=\{p\rightarrow q\}=\mathcal{B}_2(\sigma,p)$, and $\mathcal{B}_3(\sigma,p)=\{p\rightarrow q,\neg p\wedge\neg q\}$. Hence $K\sim_\sigma p=Cn(\mathcal{B}_3(\sigma,p))=Cn(\neg p\wedge\neg q)$ Note how, when determining $\mathcal{B}_2(\sigma,p)$, p is nullified, which leads to the reinstatement, or liberation, of $\neg p\wedge\neg q$.

Additional Belief Liberation Terminology

To assist in later discussions, I refer to the theory associated with σ as K_σ.

Additionally, I define σ-addition (adding a belief to σ) as follows: $\sigma+p$ is adding the belief p to the sequence $\sigma=p_1,\ldots,p_n$ to produce the new sequence $\sigma_1=p,p_1,\ldots,p_n$.[5]

[5]This is also the technique described in [Chopra, Georgatos, & Parikh2001]—though, again, I have reversed the order.

This research is even closer to the work in this dissertation than the research done by Wassermann. The key differences will be discussed in detail in Chapter 3.

2.3 Recovery

Theory Contraction Adheres to Recovery

The Recovery postulate for theory[6] contraction [Alchourrón, Gärdenfors, & Makinson1985] states that for a theory K and a belief p, $K \subseteq (K–p) + p$, where $K–p$ is theory contraction, and $K + p =_{def} Cn(K \cup \{p\})$, theory expansion[7] (to review, see Section 2.2.3).

Example 2.3.1 Let $K = Cn(\{s,d\})$. Avoiding logically equivalent beliefs, $K = \{s,d,s \wedge d,s \vee d,s \rightarrow d,d \rightarrow s,s \leftrightarrow d,s \vee \neg s\}$.[8] Assuming minimal change: $K–(s \vee d)$ *would be* $\{s \rightarrow d,d \rightarrow s,s \leftrightarrow d,s \vee \neg s\}$. $(K–(s \vee d)) + (s \vee d) = Cn(\{s \rightarrow d,d \rightarrow s,s \leftrightarrow d,s \vee \neg s\} \cup \{s \vee d\}) = Cn(\{s \vee d,s \rightarrow d,d \rightarrow s,s \leftrightarrow d,s \vee \neg s\})$—and this closure is equivalent to K.[9] Therefore, $K \subseteq Cn((K–(s \vee d)) + (s \vee d))$.

Recovery holds for belief theories, because classical logic closure requires that $(\forall p,q \in K) : p \rightarrow q$ is also in K. This enables Recovery to hold for belief theories, and it is the missing ingredient for belief base contraction adhering to Recovery.

Base Contraction Does Not Adhere to Recovery

Recovery for a belief base B and the belief p is written: $B \subseteq Cn((B \sim p) + p)$ and is called *Base-Recovery*. An even more rigid version of recovery for bases is $B \subseteq ((B \sim p) + p)$, and I call that *Strict Base Recovery*.

---

[6]Recall that a theory is a deductively *closed* set of beliefs: $K = Cn(K)$

[7]Theory expansion is specifically *not* defined as $K + p =_{def} K \cup \{p\}$. Theory expansion results in a new *theory*.

[8]Note that this assumes that s and d are the only atomic sentences in $\mathcal{L}$. If, for example, $e \in \mathcal{L}$, then other beliefs would also be in $Cn(K)$—such as $s \vee e, s \vee \neg e$, etc. My thanks to Sven Hansson for pointing this out.

[9]This is because $\{s,d,s \wedge d\} \subseteq Cn\{s \vee d,s \leftrightarrow d\}$.

For kernel contraction of bases that are *not* logically closed (but are assumed to be consistent), I divide the possible belief bases into four cases for examining whether Recovery holds:

1. If $p \in Cn(B) \wedge p \notin Cn(B \setminus \{p\})$, then $B \sim p = B \setminus \{p\}$ and $(B \sim p) + p = B$. So, Recovery holds.

2. If $p \in Cn(B \setminus \{p\}$, then Recovery is *not* guaranteed to hold (see example below)

3. If $p \notin Cn(B)$ and $B + p \nvdash \perp$, then $B \sim p = B$. $(B \sim p) + p = B \cup \{p\}$. So, Recovery holds.

4. If $p \notin Cn(B)$ and $B + p \vdash \perp$, then Recovery holds vacuously, because $(B \sim p) + p$ is inconsistent.

Note that cases 1, 3, and 4, actually adhere to the stricter and more impressive Strict Base Recovery. The only case that does not adhere to Recovery is case 2:

Example 2.3.2 Given the belief base $B = \{s, d\}$, $B \sim (s \vee d) = \{\}$. $(B \sim (s \vee d)) + (s \vee d) = \{(s \vee d)\}$. $\{s, d\} \nsubseteq \{(s \vee d)\}$ Therefore, $B \nsubseteq Cn((B \sim (s \vee d)) + (s \vee d))$.[10]

However, for any finite base B that is not logically closed, $(\forall p \notin Cn(B \setminus \{p\}))$: $B \sim p = B \setminus \{p\}$. Therefore, $(\forall p \notin Cn(B \setminus \{p\}))$: $B \subseteq ((B \sim p) + p)$. This statement covers all three of the recovery-adhering cases (1,3, and 4) mentioned above. In other words, for contraction by p, Strict Base Recovery does hold for a belief base whose set of p-kernels is either $\{\}$ or $\{\{p\}\}$, regardless of whether p is consistent with the base or not.

I assume the operation of selecting the beliefs to be removed from a base B during the operation of kernel contraction is performed by a *global incision function* [Hansson1999] guided by minimal change. An *incision function* is a function that determines which beliefs should be removed during the operations of kernel contraction and kernel consolidation. A *local* incision function is associated with a single base; once it is used to change the base, it can no longer be used, because it is not defined for the new base. A *global* incision function is applicable to all bases and can be used in a series of belief change operations (referred

[10] This example comes from the Cleopatra's children example introduced in [Hansson1991b] and discussed later.

to in the literature as *iterated belief revision* or *iterated belief change*).[11]

As seen in case 2, above, Recovery does not hold for kernel contraction when elements of a p-kernel in B are retracted during the retraction of p, but are not returned as a result of the expansion by p followed by deductive closure. Not only do these base beliefs remain removed from the base, but derived beliefs that depend on them are also not recovered into the belief space.

Example Given the base $B = \{s, d, s \rightarrow q\}$, $B \sim s \vee d = \{s \rightarrow q\}$, and $(B \sim s \vee d) + s \vee d = \{s \vee d, s \rightarrow q\}$. Neither s nor d is recovered as individual base beliefs, and the derivable belief q is also not recovered in the belief space (i.e., its implicit presence in the belief space is not recovered). I feel the assertion of $s \vee d$ means that its earlier retraction was, in hindsight, not valid *for this current state*, so all effects of that retraction should be undone.

General Support of Recovery

There are various criticisms of Recovery in the literature (see [Hansson1999] and [Williams1994a] for discussions and further references). I address these criticisms below, but state my general argument here. My defense of Recovery is predicated on the fact that the recovered beliefs were at one time in the base as base beliefs. The recovery of those previously retracted base beliefs should occur whenever the reason that caused them to be removed is, itself, removed (or invalidated). In this case, the previously retracted beliefs should be returned to the base, specifically because they were base beliefs and the reason for disbelieving them no longer exists. However, this return hinges on the verification that no alternative and stronger reason for the continued removal of these beliefs has been introduced into the system. It seems reasonable that taking something away and then returning it should recover the original state—even for belief bases. And,

[11] The research regarding iterated belief change is expansive. In this case, the key concept it that any decision function for performing belief change on some knowledge state KS_1 must be independent of that *specific* knowledge state (i.e., *global*). Otherwise, once the knowledge state is changed to become KS_2, the decision function can no longer be used to perform belief change on the new knowledge state. This topic is discussed in more detail in Section 4.5.4.

although expansion is not a true "undoing" of contraction, I choose to address the general discontentment with Recovery as a whole.

Criticisms of Recovery

The general argument is that Recovery is not as important a contraction postulate as the other postulates presented by the AGM research, and I understand that argument. But, adhering to Recovery is nice, if possible.

Arguments both for and against Recovery follow a typical pattern:

1. $\mathcal{A}$ believes some set of beliefs B; and S is a subset of B.
2. Some new highly credible evidence forces contraction by a belief p that is derivable from each element in S, therefore $S \cap Cn(B \sim p) = \emptyset$.
3. New information (more credible than the evidence in (2)) returns p to $\mathcal{A}$'s beliefs.
4. Pro-Recovery arguments then discuss why S should return to the belief space. Anti-Recovery arguments state how Recovery indicates that all elements in S should return, but they then point out how some element in S seems strange to believe *merely* as a result of the information coming in at step (3).

The agent had some reason to have faith in those beliefs in step (1). Their retraction was due to the (in hindsight) *erroneous* evidence in step (2); so it is not unreasonable to have faith in those beliefs at (4) once the "error" in (2) is discovered.

Table 2.3 shows the beliefs of the agent $\mathcal{A}$ in two anti-Recovery arguments from [Hansson1991b].

The anti-Recovery argument regarding Cleopatra's children has several troubling issues.

1. There is an assumption that information from a historical novel is not suitable for a belief base. If the novel had seemed totally fictional, $\mathcal{A}$ would not have believed s and d in the first place, so the information in the novel must have some credibility. As recommended in [Rapaport & Shapiro1995],

	Cleopatra's Children	George the Criminal
1.	"Cleopatra had a son." (s) "Cleopatra had a daughter." (d) $B_{1C} = \{s, d\}$	"George is a criminal." (c) "George is a mass-murderer." (m) $B_{1G} = \{c, m\}$
2.	The source of s and d was a historical novel, so $\mathcal{A}$ decides she had no children, at all. $B_{2C} = B_{1C} \sim (s \vee d) = \{\}$	"George is *not* a criminal." $\neg c$ $m \rightarrow c$, so $m \notin B_{2G}$ $B_{2G} = (B_{1G} \sim c) + \neg c = \{\neg c\}$
3.	"Cleopatra had a child." ($s \vee d$) $B_{3C} = B_{2C} + (s \vee d) = \{s \vee d\}$ Recovery would return s and d	"George is a shoplifter." (x) $x \rightarrow c$, so c replaces $\neg c$; Recovery would return m $B_{3G} = (B_{3G} + x) = \{x, c\}$
4.	Why believe Cleopatra has a son *and* a daughter just because "she had a child"?	Why believe George is a mass-murderer just because he is a shoplifter?

Table 2.1: The belief changes and reasonings made during two classic anti-Recovery arguments. Row 4 gives the "reason" that Recovery seems counter-intuitive. These reasons are addressed in the text.

a reasonable way to deal with texts that contain some fiction and some truth is for an agent to believe information from novels when necessary (when the agent finds that information to be useful for some real world reasoning), unless or until there exists some more credible reason to doubt it.[12]

2. When discounting the information from the novel, it should have been removed in the same syntax in which it was entered: by retracting s and d separately.

3. The semantics behind "Cleopatra had a child." are ambiguous: if it means that she had *exactly* one child, then $\mathcal{A}$ should add $((s \vee d) \wedge \neg(s \wedge d))$ to the base and Recovery is not an issue; if it means that she is merely not childless, then the historical novel was, at the very least, "half-right" regarding discussion of her children, so why not give it the benefit of the doubt (barring any contradictory evidence) and return both s and d to the belief space?

[12] This approach was adopted, as discussed in [Rapaport & Shapiro1995], to implement a cognitive agent that could "read" fiction. They state, this "allows us. . . to learn from narratives—indeed, to understand the real world in terms of narratives."

Regarding the criminal example, the argument is that it seems strange to force $\mathcal{A}$ to believe George is a mass-murdered just because he is a shoplifter. However, the original reason to believe George is a mass-murder still exists (as far as *we* know), and there is no longer the presence of more credible information refuting it. The uneasy feeling we get when looking at this example comes from "shoplifter" being so far from "mass-murder" along the spectrum of "criminal". Change "shoplifter" to "kidnapper" and "mass-murderer" to "killer". Now Recovery seems less strange. Sometimes intuition does not follow logical reasoning, and I hope that one use for implemented computational reasoners will be to alert humans to contradictions between logic and intuition. It is possible that much of human intuition is merely reasoning from a set of subconscious rules; if true, exposing the contradictions between logic and intuition will expose those rules, and the humans can either add them to the system's rule set, or make a revision within their human minds.

Pro-Recovery Examples

For every anti-Recovery argument, there are examples that support the postulate of Recovery.

Battle Scenario: Pro-Recovery Example Consider a war scenario where two sources give conflicting information. Source-1 says that Country-A is planning to attack Country-B and Country-C: ($p = HasPlanAttack(A,B) \land HasPlanAttack(A,C)$). Source-2 (considered more reliable that Source-1) says that Country-A and Country-B are allies: ($q = Ally(A,B)$), where Ally is symmetric (i.e., $Ally(A,B) \Leftrightarrow Ally(B,A)$).

There are two rules that are considered more credible than either p or q:

- [Rule-1] A country does not plan to attack its allies: $\forall(x,y) : Ally(x,y) \rightarrow \neg(HasPlanAttack(x,y))$.
- [Rule-2] No country attacks another without a plan: $\forall(x,y) : Attack(x,y) \rightarrow HasPlanAttack(x,y)$.

Therefore, there is a contradiction (arising from Rule-1 and q conflicting with p), and consistency maintenance forces the removal of the weaker base belief p.[13]

When Country-A actually *does* attack Country-B ($Attack(A,B)$), then that information combines with Rule-2 to generate the belief $HasPlanAttack(A,B)$ which conflicts with Rule-1 and belief q. Since the attack and the two rules are considered more credible than belief q, the contradiction forces the retraction of q. When q is removed, it seems intuitive to make the following changes:

1. The information from Source-1 (p) should be recovered (re-inserted) back into the base. (Recovery)
2. Country-C should prepare for a possible attack from Country-A.

Altered Cleopatra: Pro-Recovery Example Now, consider a pro-Recovery version of Hansson's Cleopatra example.

1. I believe for some reason (currently not recalled) that Cleopatra had a son (s) and also that Cleopatra had a daughter (d). My current belief base regarding this topic is $B_1 = \{s,d\}$.
2. Then a dinner companion states, with an air of expertise, that Cleopatra was childless. Because of this, I add $\neg(s \vee d)$ to my base (contracting it by $s \vee d$ for consistency maintenance): $B_2 = (B_1 \sim (s \vee d)) + \neg(s \vee d) = \{\neg(s \vee d)\}$.
3. Curious, upon returning home, I query the internet for more information on Cleopatra and discover numerous websites referring to her as a mother. So, I return the removed belief $(s \vee d)$ to my base at a stronger position than the information added in (2), and consolidate: $B_{3i} = (B_2 + (s \vee d))! = \{s \vee d\}$.
4. If Recovery held for bases, s and d would also return to the base (or at least the belief space). Having these two beliefs return to the *actual base*, not just the belief space, seems intuitively correct to me, because the reason for their removal was the (seemingly) erroneous belief (introduced in step 2) that Cleopatra had no children. With that removed, whatever reason prompted me to believe s and d still

[13] In this example, I assume that p is considered weaker than q, because its source is the weaker source.

exists (as far as I know) to justify that belief. And, now, there is no longer any contradiction to block those beliefs. So, s and d should both return to my set of base beliefs.

This is a good place to mention the issue of the granularity of base beliefs. One could argue that the belief $\neg(s \vee d)$ inserted in step (2) was not really a base belief, but a belief derived from other base beliefs — "experts are credible", "someone sounding like an expert is an expert", "we should believe what credible people say", etc. But these "base beliefs" are also derived from even more basic beliefs, and we have wandered onto a slippery slope. For the purpose of this dissertation, I will assume on faith that a base belief is, simply, just that: a base belief.

2.4 TMS Research

2.4.1 Introduction

A truth maintenance system (TMS) stores the reasons for belief derivations for later recall — to avoid repeating derivations and to retrieve derivation information. It is this storing the reasons that underlie the derivation of a belief that prompted Hansson to refer to TMS research as a "track-keeping" subset of the "composite" sentential representations research group [Hansson1991a]. This section discusses and compares the differences between (1) a justification-based truth maintenance system (JTMS) as presented in [Doyle1979] and (2) an assumption-based truth maintenance system (ATMS) as described in [de Kleer1986, Martins1983, Martins & Shapiro1988]. Both systems follow a foundations approach to belief revision [Hansson1993]. This means that they distinguish derived beliefs [Hansson1999] from base beliefs—base beliefs being the core beliefs, which have independent standing. In this approach, a derived belief that loses its support (the reasoning by which it was derived) is no longer believed.

There is also a brief explanation of a logic-based TMS (LTMS) and an incremental TMS (ITMS)—two progressively altered versions of a JTMS (cf. Sections 2.4.5 and 2.4.6, respectively). These systems differ in many aspects, but the general method of storage, maintenance and retrieval of derivation information will be the primary focus of discussion. Since the LTMS and ITMS are derivations of the JTMS, our discussion will eventually be reduced to comparing and contrasting the benefits of JTMS vs. ATMS systems.

The central issue being considered is *how* to store and maintain a record of the supports (or reasons) for each derived belief. Accessing a belief's supports is necessary to determine (1) whether it is currently believed—to avoid repeating derivations, (2) if it might be believed under specified circumstances, (3) *why* it is currently believed, and (4) how the base beliefs might be altered to remove that belief. Essentially, the appropriate TMS system to choose depends on the application to which it will be applied.

2.4.2 Terminology

For the purpose of this dissertation, I am assuming a monotonic KR&R system that needs to perform truth maintenance. Most TMSs separate the TMS from the problem solver to simplify the algorithms for each. One exception is the SNePS system, where the inference engine and ATMS-style belief maintenance system are integrated [Shapiro & The SNePS Implementation Group2004, Martins & Shapiro1988].

The KR&R system can reason about the beliefs in the knowledge base—which are also referred to as propositions. These do not reflect the difference between facts, guesses, suppositions, etc. Beliefs that are asserted (by the user) to be true with independent standing (i.e., they do not depend on any other belief) are called base beliefs (also called hypotheses or assumptions). This core set of beliefs is called the belief base. These base beliefs can be removed (or retracted) by the TMS.[14]

The base beliefs unioned with the beliefs known to be derived from them make up the belief space

[14] Although the TMS literature differentiates between a premise (which is always believed and cannot be removed) and an assumption (which is retractable), this simple alteration to either system is not worth discussing here.

[Martins & Shapiro1983] for the KR&R system.[15] A derived belief p cannot be retracted explicitly, because it is dependent on the set A of base beliefs that minimally underlie its derivation. In this case, *minimal* means that no proper subset of A is *known* to derive p. This minimal set is called an *origin set* for p [Martins & Shapiro1983]. A belief can have more than one origin set.

To remove a derived belief from the belief set, at least one of the base beliefs in each of its origin sets must be retracted to eliminate its re-derivation.

If we think of a derivation tree for some belief q, I refer to q as the child node derived (using a single rule of inference in the system) from its parent nodes. In this sense, the origin set is comprised of the leaf nodes of the derivation tree representing the outermost ancestors along each path.[16] The parent nodes are the propositions that "directly produced" q [Martins & Shapiro1988].

A derivation tree is shown in Figure 2.1, where $\neg q$ is a derived node whose parent nodes contain the beliefs t and $t \rightarrow \neg q$. The origin set for $\neg q$, however, is the set of base beliefs $\{s,\ s \rightarrow t,\ t \rightarrow \neg q\}$. The belief z has *two* origin sets: $\{r \rightarrow z,\ r\}$ and $\{r \rightarrow z,\ p \rightarrow r,\ p\}$.

A *justification* for a derived belief p contains a set of parent nodes, used to derive p in one inference step, as well as the inference rule that was used in the derivation.[17] Since a justification is provided by the problem solver to the TMS, technically, it contains three elements [de Kleer1986, Forbus & de Kleer1993]: the node that was derived (the consequent), the set of parent nodes (the antecedents), and the inference rule used in the derivation (the informant). Two different (one-step) derivations of the same belief result in two different justifications. In Figure 2.2, q has two justification sets: $\{p,\ p \rightarrow q\}$ and $\{r,\ r \rightarrow q\}$.

In all TMS systems, a node also has a *label* that indicates the belief status of the node based on the

[15] This belief space was called the belief set in [Doyle1979] and the context in [de Kleer1986].

[16] In the case where a base node is also derived, it may not "look" like a leaf node, but the path *may* stop with the base node that it hits or it may continue on to follow the *alternate derivation* of that node. There is a danger of cycling, however, which is discussed later.

[17] This corresponds to Doyle's SL-justification. His more complicated CP-justification can be reduced to a series of SL-justifications, and will not be discussed here [Doyle1979].

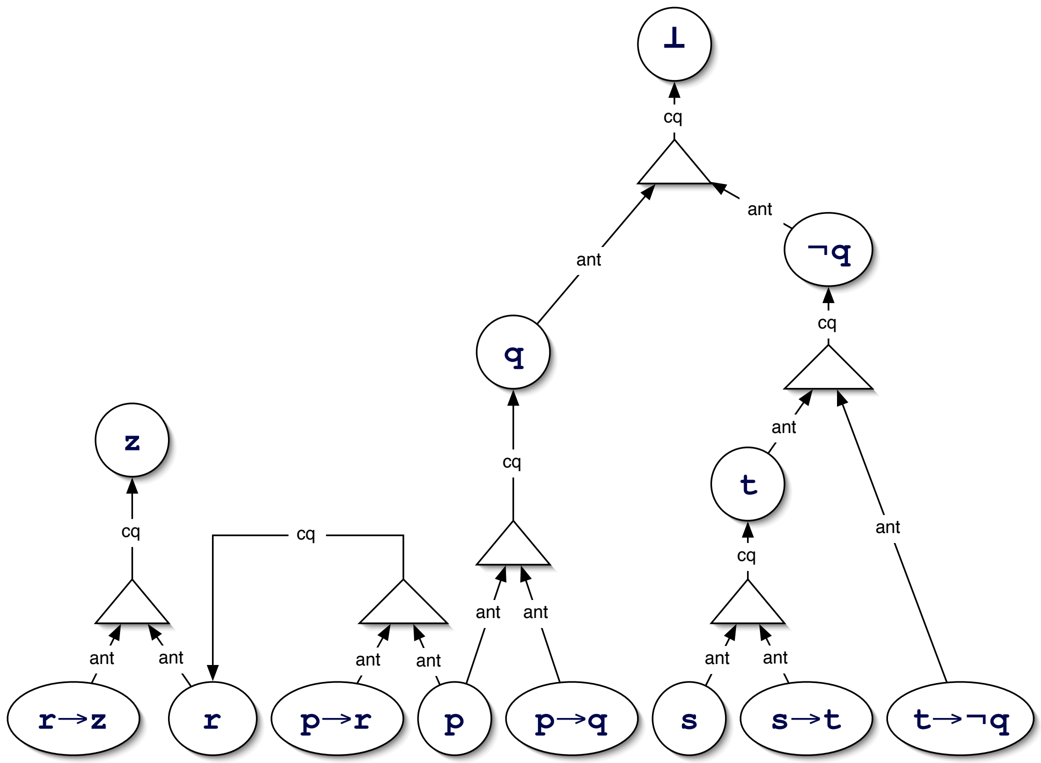

Figure 2.1: A derivation tree. Circles and ovals are belief nodes that each contain a belief (or an inconsistency, as in the topmost node of the tree). Triangles connect derived beliefs (by the *cq* arc) with the beliefs they were *immediately* derived from—in one derivation step (connected with the *ant* arc). The bottom row of beliefs are base beliefs. The belief nodes within the tree are derived beliefs. Note that *r* is both a base belief *and* a derived belief.

current set of base beliefs.

The early TMS systems dealt with different kinds of data (the representation of the proposition from the problem solver). Doyle's JTMS could only represent definite clauses [Doyle1979, Forbus & de Kleer1993]; McAllester reasoned with disjunctive clauses [McAllester1978]; de Kleer's ATMS dealt with Horn clauses [de Kleer1986, Forbus & de Kleer1993]; and Martins and Shapiro presented an ATMS-style system in a higher-order relevance logic [Martins & Shapiro1988]. This dissertation, ignores the limitations and specificities of the *original* systems and their implementation details. Instead, it will focus on the general techniques for storing derivation information and their corresponding benefits and detriments.

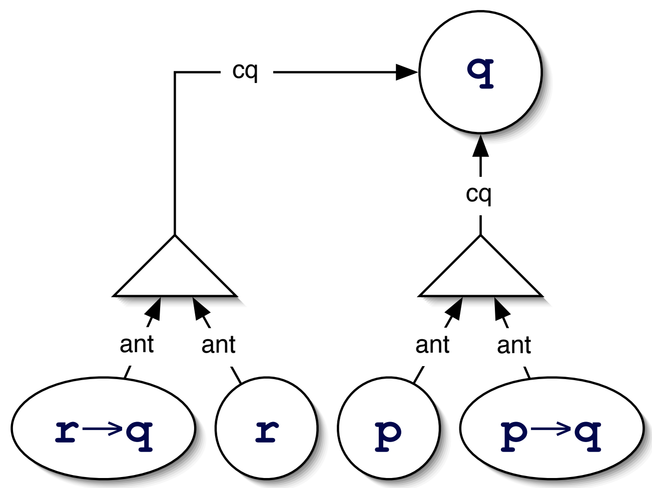

Figure 2.2: A small derivation tree with two justifications for q.

2.4.3 JTMS

Doyle's JTMS [Doyle1979] launched the beginning of the study of truth maintenance systems by applying *dependency-directed backtracking* [Stallman & Sussman1977] to truth-maintenance. A newly detected contradiction in the system could be traced, through the use of the justifications, to the underlying set of base beliefs from which it was derived. If one of those base beliefs was removed, then the contradiction was no longer known to be derivable using that derivation path.

Previously, elimination of a contradiction in a KR&R system used *chronological backtracking*, which removed base beliefs in reverse order of their assertion until the contradiction was no longer known to be derivable. Although removed nodes that did not underlie the contradiction could be reinstated, this simple algorithm was inefficient.

The JTMS resulted in a less damaging resolution of the contradiction by not removing beliefs unrelated to the derivation of the contradiction just because they were added or derived after the base beliefs underlying the contradiction. It also allowed more direct access to the possible culprit hypotheses (those underlying the contradiction).

How JTMS derivation information is stored

When a belief is derived, its justification is sent to the JTMS for storage and updating of the system. The actual implementation of this storage is immaterial as long as the information is retrievable.

- It might be stored as a justification node between the parent nodes and the derived child node, using "ant" arcs (for antecedent) to the parent nodes and a "cq" arc (for consequent) to the child node, as shown in Figure 2.1.
- Alternatively, the parent nodes could merely list the child node in a consequent list, and the derived node could store the rest of the justification as a tuple containing the set of parent nodes (an antecedent list) and the derivation rule used.

The justification for a base belief has an empty set for the parent nodes (antecedents) and *nil* for the derivation rule. In Figure 2.1, the justifications stored with the belief r could be stored as:
$(\langle\{\}, \mathit{nil}\rangle, \langle\{p,\ p \rightarrow r\}, \mathrm{MP}\rangle)$, where MP stands for *modus ponens*.

Given a set of base beliefs, a JTMS marks all its beliefs as either *in* (believed) or *out* (disbelieved or unknown). The base beliefs are *in* by definition. If the parent nodes in one of a derived belief's justifications are *in*, then the belief is marked as *in*. If a belief has multiple justifications, only the elements of one justification parent set need be marked as all *in* for the belief to be marked *in*. A node can be a base belief and also have justifications — if it is retracted as a base belief, it might still be derivable through one of the justifications. This would be the case if r was removed from the set of base beliefs in Figure 2.1—it would still be derivable from $\{p,\ p \rightarrow r\}$.

By tracing down a node's justifications recursively, the JTMS can determine the series of derivations that produced the derivation—called the "well-founded supporting argument" in [Doyle1979]. The base nodes at the ends of the derivation paths (for one well-founded supporting argument) make up a set of base beliefs that support the derived node.

Any removal of a base hypothesis requires updating the *in*/*out* markers. Likewise, adding a justification to a belief that results in that belief's label converting from *out* to *in* also requires recursively updating the *in*/*out* labels in its consequent list and any successive consequent lists of nodes changed from *out* to *in*.

JTMS Cycles

A well-founded supporting argument for a derived belief requires that the chain of justifications *not* have any cycles in it. This way a node cannot support itself. The testing for this, however, is non-trivial and one of the drawbacks of using a JTMS—see [Doyle1979] for the algorithms to check for cycles.

Resetting node labels required setting the labels of all beliefs to *nil*, resetting the base beliefs labels to *in*, then rechecking each consequent node (recursively) for a well-founded supporting argument.[18]

2.4.4 ATMS

When the problem solver sends an ATMS a justification, the ATMS stores it just like the JTMS-style systems [de Kleer1986, Forbus & de Kleer1993] (except for SNePS [Martins & Shapiro1988], where it is used to compute the label, then discarded).

The labeling is the key difference between the ATMSs and the JTMSs. The label for a derived node in the ATMS is the node's origin set (the set of base beliefs that *minimally* underlie the derivation of the belief).

The origin set for a hypothesis is the singleton set containing just that hypothesis. When a belief is derived from other beliefs, there is a series of rules governing the combining of the "parent" origin sets to form

[18] A later version of a JTMS, called an ITMS, shortens this process (cf. Section 2.4.6).

the origin set for the "child". For example, Modus Ponens uses set union: assuming that $os(p)$ is the origin set for the belief p, then p derived from q and $q \rightarrow p$ has the origin set $os(p) = os(q) \cup os(q \rightarrow p)$. Implication introduction, however, involves set subtraction [Martins1983, Martins & Shapiro1988, Shapiro1992].

Multiple derivations can result in multiple origin sets, and these duplications can multiply depending on the rules for origin set combining. In the Modus Ponens example above, if q and $q \rightarrow p$ each had two origin sets, then their derivation of p could result in p having four origin sets!

Given a set of base beliefs, B any belief is known to be derivable from that set if at least one of its origin sets is a subset of B.

An origin set is not saved if either (1) it is a superset of another origin set for the same node or (2) it is known to be an inconsistent set. Case (1) is used to prevent circular supports—so that no supports with cycles are used—and to insure that retraction of a single element of any origin set eliminates that specific derivation.

These origin-set labels indicate which belief bases (also called contexts in [Martins & Shapiro1988]) a node is derivable from. Any context that is a superset of a node's origin set (any one of its origin sets) will have that node in its belief set (the set of explicit beliefs).

No updates are needed when a base node is removed from the context. If a node gets a new justification, however, any alteration to its origin sets is propagated recursively through its consequences and their consequences, and so on. An exception to that is SNePS [Martins & Shapiro1988], where propagation does not happen automatically (unless forward inferencing is used).

As stated earlier, the origin set for $\neg q$ (in Figure 2.1) is the set of base beliefs $\{s,\ s \rightarrow t,\ t \rightarrow \neg q\}$. The belief z has *two* origin sets: $\{r \rightarrow z,\ r\}$ and $\{r \rightarrow z,\ p \rightarrow r,\ p\}$. Therefore, z would be believed even if r were removed from the base, because it would still be derivable from the set $\{r \rightarrow z,\ p \rightarrow r,\ p\}$.

ATMS labels also allow the system to reason from a set of base beliefs that is not the current base.

For example, using Figure 2.1, the system can answer the question, “Is z in the known belief space of $\{r,\ p,\ s\}$?” without having to re-compute in/out labels or repeat previous derivations. The answer is “no,” because no origin set for z is a subset of $\{r,\ p,\ s\}$. Unlike the in/out labels of a JTMS, origin set information is *not* lost as the current context changes. Because the ATMS labels may contain origin sets that are not subsets of the current context, ATMS reasoning is referred to as reasoning in multiple belief spaces [Martins & Shapiro1983].

2.4.5 LTMS

McAllester [1978] developed an alternate version of the JTMS called a logic-based TMS (LTMS). Nodes were propositional formulas whose labels were “true”, “false”, and “unknown”. The *informant* part of the justification is represented as a clause: e.g., to represent that q was derived from p and $p \rightarrow q$, the clause $((p.\text{false}) \vee (p \rightarrow q.\text{false}) \vee (q.\text{true}))$ would be the informant, q would be the consequent, and $\{p, p \rightarrow q\}$ would be the antecedent set. If the node is an assumption (hypothesis; base belief), the informant is merely “Assumption”.

Antecedent nodes are labeled “true” or “false” before the derived node label is set; therefore, it cannot be supporting itself. Additionally, each node has *only one* stored support. If the label for a node is changed, resetting the labels on the affected nodes involved setting the label of each possibly affected consequent node to “unknown”, then re-determining its support.

2.4.6 ITMS

The incremental TMS (ITMS [Nayak & Williams1998]) is an altered version of the LTMS. It was developed to reduce the number of unnecessary node label changes (changes from “true” to “unknown” and then back to “true” again, once it was re-evaluated).

When a consequent node loses its support, the ITMS tests for alternate supports before changing all consequent nodes recursively. The cycle prevention heuristics (using increasing numerical tags) are an improvement over the algorithms used by the JTMS and LTMS in all but a select few worst case examples. The average improvement was a seven-fold reduction in unnecessary label changes.

2.4.7 General Pros and Cons

There are differences between the ATMS and JTMS implementations other than those discussed above, but our focus is on how they maintain the belief status of a node in its label. Note that *both* the JTMS and ATMS recommend saving the justification information for updating label information [Doyle1979, McAllester1978, de Kleer1986, Forbus & de Kleer1993].

Computational load

The JTMS label update algorithm is linear in space and time. The ATMS update or query answer can possibly be exponential in both. But depending on the application, an ATMS can save time (such as when label sets remain small, where a task requires finding most or all solutions, when the number of context changes is large compared to the number of queries, and when multiple context reasoning at one time is needed). The JTMS requires work to insure a well-founded supporting argument has no cycles, and ATMS algorithms eliminate that work.

Multiple contexts

For reasoning in multiple belief spaces, [Martins1983], the ATMS style of recording supports is ideal. Whether shifting contexts or querying in a context other than the current one, it is easy to determine if a belief is known to be derivable from a set of base beliefs using a subset test on the origin sets and context. Basically, reasoning done in one context is available as origin set information in all contexts.

A context change would require the JTMS to reset *in*/*out* markers for the entire belief space. An alternate context query would require a traversal down the justification paths with backtracking until a successful derivation from the context beliefs was found.

Belief revision

Accessing the base beliefs for rapid culprit selection is speedy in the ATMS as compared to the JTMS. Combining origin sets of contradictory beliefs provides the user with sets that are known to be inconsistent. Removal of at least one belief from each inconsistent set (also called a nogood; cf. page 4) eliminates known derivations of the contradiction. Additionally, no updating is necessary after hypothesis removal.

If a single context's base beliefs will be asserted and retracted often, ATMS is, again, the better option. Better, that is, assuming that (1) there is no pressing need to update *all* the flags that label whether each belief is currently asserted (believed) by the system and (2) determining the value of a previously derived belief by accessing its origin set is preferred to tracing the derivation tree to the base nodes underlying its derivation.

Multiple supports for a belief

It is easy to record that a belief is both a hypothesis and a derived belief in either system—JTMS or ATMS. Likewise for multiple derivations for a belief. Multiple derivations can be determined in an LTMS, but the labels for any given node only contain one such derivation; if that derivation is lost, there is a search for an alternate derivation to take its place.

2.4.8 What do current TMS systems do?

Exploration of recently implemented TMS systems indicates that both ATMS and JTMS styles are still popular.

Cyc [Cycorp2001a, Cycorp2001b], the largest KR&R system in use, is a JTMS. It has also put speed as a primary concern, because (1) its size could be a major cause for slow processing and (2) its clients often need results in real time.

An Incremental TMS [Nayak & Williams1998] (an extension of McAllester's LTMS [McAllester1978], which was expanded from the JTMS) is currently in use on Nasa's Deep Space One (DS-1) [Pell *et al.*1997].

Systems focusing on multi-agent reasoning need the multiple context application of an ATMS. Such systems include DiBeRT [Malheiro & Oliveira1996] and DARMS [Beckstein, Fuhge, & Kraetzschmar1993].

Castro and Zurita proposed a Generic ATMS [Castro & Zurita1996] to remove the requirement that beliefs have binary truth values (elements of $\{0,1\}$). The generic ATMS was then (a) particularized to deal with the multivalued (in the range of [0,1]) logic and (b) particularized to deal with a fuzzy logic.

2.4.9 TMS Conclusions

An application that involves multiple contexts, frequent removal of base beliefs (contraction), and a monotonic logic would benefit from an ATMS-style implementation. Conversely, reasoning in a single context with little need for base contraction favors a JTMS-style implementation.

2.5 Different Ways to Order the Base Beliefs

Throughout this dissertation I discuss sets of beliefs that are ordered in various ways. The concept of prioritizing beliefs during belief change operations is a reflection of some type of ordering. One common way to order beliefs is by recency—the more recent a belief is the more preferred it is. There are alternatives to recency, however, and these require an understanding of the various orderings that are possible.

To facilitate the discussion and comparisons of these *orderings* (also called *orders*), I present an overview of the various orders that will be used. These orders are binary relations represented as $\succeq$ and $\succ$, where $p \succeq q$

means that p is preferred over q and $p \succ q$ means that p is strictly preferred over q.[19] These binary relations have one or more of the following properties:

- Reflexivity: $p \succeq p$;
- Antisymmetry: If $p \succeq q$ and $q \succeq p$, then $q = p$.
- Transitivity: If $p \succeq q$ and $q \succeq r$, then $p \succeq r$,
- Comparability: For all p, q, $p \succeq q$ or $q \succeq p$.

Lastly, $p \succ q$ IFF $p \succeq q$ and $q \not\succeq p$.

A *total order* has all four properties: reflexivity, antisymmetry, transitivity, and comparability. A total ordering is also referred to as a *linear ordering* or a *chain*. A common example of a total order is the relation of $\leq$ on the set of integers: e.g., $4 \leq 5$; given any two integers x and y, if $x \leq y$ and $y \leq x$, then $x = y$.[20]

A *partial order* is reflexive, antisymmetric, and transitive, but is not *required* to be comparable. A partially ordered set is also called a *poset* or just an ordered set. A well known example of a partial order is the subset relation ($\subseteq$): $\{1,2\} \subseteq \{1,2,3\}$, but the two sets $\{1,2\}$ and $\{2,3\}$ are not comparable using the $\subseteq$ relation ($\{1,2\} \not\subseteq \{2,3\}$ and $\{2,3\} \not\subseteq \{1,2\}$). A total order is also a partial order.

A *pre-order* (also referred to as a *quasiorder*) is an ordering that is both reflexive and transitive. A partial order is also a pre-order. Below is an example of a pre-order over the following foods: ice cream (i), steak (s), artichokes (a), raspberries (r), beets (b), and liver (l). I like ice cream the best and liver the least; I like steak and artichokes equally well; I like both steak and artichokes better than either beets or raspberries; and,

[19] Most literature represents orderings from weakest to strongest using the symbol $\preceq$. In this dissertation, I reverse that order, because the orderings will be later used to order priority queues, where the first element in the queue is the most preferred element in the queue. It would be confusing to have the first element in the queue be the last element of the ordering sequence, and switching the direction of the ordering at that time would also be confusing. So, the base beliefs are ordered from strongest to weakest to maintain a consistent representation throughout this dissertation.

[20] Note that antisymmetry is *not* the same as "not being symmetric": the relation $<$ on integers is not symmetric, but it is only vacuously antisymmetric: for any two integers x and y, it is impossible to have both $x < y$ and $y < x$.

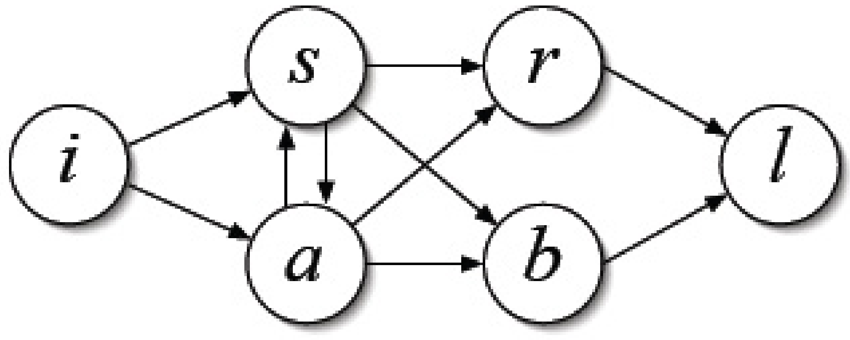

Figure 2.3: A graph showing a **pre-order** with arcs from more preferred items to less preferred ones. The lack of antisymmetry is shown by the arcs going both ways between a and s. The lack of comparability is shown by the absence of any path between r and b. Note: if an arc existed between r and b, the ordering would be comparable and would be called a *total pre-order*.

regarding beets and raspberries, I am unable to decide which I prefer.[21] These preferences form a pre-order that can be represented with the following expressions: $i \succeq s$, $i \succeq a$, $s \succeq a$, $a \succeq s$, $s \succeq r$, $a \succeq r$, $s \succeq b$, $a \succeq b$, $b \succeq l$, $r \succeq l$. It can also be shown by the graph in Figure 2.3, where each arc goes from a more preferred item to a less preferred item. The lack of antisymmetry is shown by the arcs going both ways between a and s. The lack of comparability is shown by the absence of any path between r and b.

A *total pre-order* is a pre-order that is also comparable. Figure 2.3 shows a total pre-order for the *subset* of items $\{i,s,a,r,l\}$—a set that is missing the item b and thus eliminating the incomparability between r and b that was previously discussed.

This next example illustrates a total pre-order over the set of beliefs $\{a,b,c,d\}$ given that:

- each belief is assigned a rank: $\text{Rank}(a) = 1$; $\text{Rank}(b) = 2$; $\text{Rank}(c) = 2$; $\text{Rank}(d) = 3$;
- $p \succeq q$ IFF $\text{Rank}(p) \leq \text{Rank}(q)$.

Note the lack of antisymmetry: $b \succeq c$ and $c \succeq b$, but $b \neq c$ (even though $\text{Rank}(b) = \text{Rank}(c)$).

[21] Although I am using the transitive relation ”like-better-than”, the relation ”like” is typically *not* transitive: If I like Mary and Mary likes John, it does not necessarily follow that I like John.

Lastly, I review the concept of the *minimal* and *maximal elements* of these orderings. A *maximal element* of some set S with an ordering relation $\succeq$ is any element m s.t. $(\nexists p \in S): \ p \succ m$. A *minimal element* of some set S with an ordering relation $\succeq$ is any element m s.t. $(\nexists p \in S) : m \succ p$. A set can have more than one maximal element; likewise, a set can have more than one minimal element. If a set has a *unique* maximal element, that element is called the *greatest* element of the set. If a set has a unique minimal element, that element is called the *least* element of the set. An element in a set that is not comparable to any other element in the set is both a maximal and a minimal element.

Looking at the elements in Figure 2.3, we can see that i is the greatest element, and l is the least element—they are the unique maximal and the unique minimal elements respectively. However, for the subset $\{s,a,r,b\}$, the set of maximal elements is $\{a,s\}$, the set of minimal elements is $\{r,b\}$, and there is no least or greatest element of that subset.

I re-iterate: a set of maximal (or minimal) elements need not be a singleton set; and any maximal (or minimal) element of a set is not, necessarily, *strictly* stronger (weaker) than the other elements in the set—instead, there is no element in the set that is *strictly* stronger (weaker) that it (review the definition above if this is still confusing).

As a last example, if all elements of a set are equivalent (i.e., no element in the set is strictly stronger than any other), then all the elements are both minimal and maximal elements of the set.

2.6 Some KR&R Implemented Systems

In this section, I briefly describe some belief revision systems that I will be discussing and comparing in this dissertation.

2.6.1 SATEN

SATEN, developed by Mary-Anne Williams, is an "object-oriented, web-based extraction and belief revision engine" [Williams & Sims2000]. It implements the AGM [Alchourrón, Gärdenfors, & Makinson1985] approach for belief revision and uses user specified rankings for the beliefs. The extraction process recovers a consistent theory base from an inconsistent ranking. SATEN can reason in either propositional or first-order logic. To query whether a belief is in the belief space of the base, you ask what its ranking would be. A degree of 0 means it is not in the belief space, else the appropriate ranking for the belief is given. No reply means it does not know the ranking. This system can be run over the Internet from the URL:
`http://magic.it.uts.edu.au/systems/stdvsmaxi.html` .

2.6.2 Cyc

Doug Lenat and Cycorp have developed Cyc [Cycorp2001a], a large knowledge base and inferencing system that is "built upon a core of over a million hand-entered assertions or rules about the world and how it works." This system attempts to perform commonsense reasoning with the help of this large corpus of beliefs (mostly default, with some that are monotonic). It divides its knowledge base into smaller contexts called microtheories, which contain specialized information regarding specific areas (such as troop movement, physics, movies, etc.). Belief revision is performed within microtheories or within a small group of microtheories that are working together, and the system is only concerned with maintaining consistency within that small group (as opposed to across the entire belief space). For example, in an everyday context, a table is solid, but within a physics context, it is mostly emptiness.

A belief can have only one truth value, so no microtheory can contain both p and $\neg p$. The technique for maintaining consistency is to check for contradictory beliefs whenever a proposition is asserted into a microtheory (as a base assertion or through derivation). When contradictions are found, their arguments (derivation histories) are analyzed, and a decision is made regarding the truth value of the propositions involved. Ranking of beliefs, however, is not a part of the system—it uses specificity to determine the truth value of a default belief. For example: Opus the penguin does not fly, even though he is a bird, because penguins don't fly. If there can be no decision based on specificity, the truth value of the default belief is unknown. A default belief loses out to a monotonic one.

Lastly, according to Cyc trainers and other contacts, contradictions that are purely monotonic bring the system to a halt until they are "fixed" by the knowledge engineers—e.g., the knowledge base is altered to eliminate the contradiction—a change made by the implementers, not the end users. During Cyc training, however, I attempted to prove this last statement and failed—revision was performed. The instructors were surprised, but thought the "user-friendly" training interface might be the cause. I hope to explore this further, but it is an example of how a system description might not match its actual performance. Typically, however, descriptions predict ideal results and performance falls short. Interestingly, my Cyc sources were expecting failure, but the system produced a viable result.

The system can be run over the Internet. The main URL for Cycorp is:
`http://www.cyc.com/` .

2.6.3 SNePS

SNePS [Shapiro & The SNePS Implementation Group2004, Shapiro & Rapaport1992] is a KR&R system whose belief space represents the belief space of a cognitive agent called Cassie. SNePS reasons using a paraconsistent, relevance-based logic, and is able to reason in multiple contexts, similar to Cyc.

SNePS attempts to derive propositions as they are asked for—either by the user or by the system as it performs backward or forward inference.A contradiction is detected when a specific proposition and its negation are both explicitly present in the belief space. At this point, belief revision is called to resolve the inconsistency, and the system considers only the propositions in the current belief space—asserted hypotheses and the propositions currently known to be derived from them—without considering any implicit beliefs. Because justifications for the contradictory beliefs are stored as sets of base beliefs that are known to underlie the contradiction (ATMS-style), SNePS automatically reduces the set of beliefs under consideration to only those involved with the derivation of the contradiction.

The SNePS system is available online for downloading by following links from the research group website. The main URL for the SNePS Research Group is:

`http://www.cse.buffalo.edu/sneps/` .

2.7 The Need for a DOBS Formalization

The research of this dissertation focuses on the fact that most of the approaches described above use the concept of retaining (or returning) consistency to define their belief revision operations. Even a belief base is considered inconsistent if an inconsistency exists in its deductive closure. Whether the inconsistency is found by deductive closure or some procedure such as resolution refutation or a SAT test, it still requires looking past what is known explicitly into the implicit beliefs to find an inconsistency. This requires time and space.

2.7.1 Some Limitations of the Above Systems

Most of the systems above check consistency either at the time a belief is entered or when consistency is an issue (e.g., during contraction, revision, extraction, query, etc.). Attempting to insure consistency takes time and, depending on the logic, might not be able to be guaranteed.

SATEN is limited to logics with SAT-determined consistency. Additionally, derived beliefs are determined at the time of query and are not saved. For a system that is constantly building on its beliefs, re-derivation could be costly.

As mentioned above, Cyc did not perform as described, and there must be some question as to other possible differences from design theory. Most specifically, Cyc literature [Cycorp2001b] refers to the microtheories as *consistent*, for lack of a better word. Inconsistencies can exist between two microtheories; but, within a microtheory, beliefs are consistent.

When asked, contacts agreed that it was possible that unknown contradictions might exist that had not, yet, been derived. In this sense, Cyc can only guarantee that its microtheories are *not known to be inconsistent*. Ideal terminology, such as *consistent* and *derivable*, is often not appropriate for use with a large or complex implemented system.

SNePS typically recognizes an inconsistency when a belief p and its complement $\neg p$ are explicitly believed by the system (explicit beliefs consist of base beliefs and previously derived beliefs). Although there are some other contradiction detection algorithms used by SNePS, it is possible that the system can be inconsistent without being aware of it (i.e., implicit contradictions can exist).

2.7.2 Needs of a DOBS

How do you implement ideal techniques in a non-ideal system? We address the need to formalize theories that take into account the fact that deductive closure cannot be guaranteed in a real-world, need-based, resource-bounded, implemented system. These theories need to define a belief maintenance system that:

1. is not dependent on deductive closure (thus, a DOBS);
2. takes time and computational limitations into account,
    - recognizing that adhering to these limitations might result in revision choices that are poor *in hindsight*; and

3. catching and correcting these poor choices as efficiently as possible.

Because of these needs, a new formalization is needed to describe the status of a DOBS—such as a concept similar to consistency, but which is not based on an ideal agent's deductive closure. The formalization for a DOBS is presented in Chapter 5.

Before dealing with the formalization of a DOBS, I first formalize the belief base optimizing operation of *reconsideration*; this is done in Chapter 3 and uses the base expansion and kernel consolidation. I then develop the efficient, anytime algorithm for dependency-directed reconsideration (DDR) in Chapter 4, and prove that DDR optimizes the belief base for a TMS (assuming it has an ideal reasoner). It is in Chapter 5, then, after defining a DOBS, that I prove that the DDR algorithm from Chapter 4 will also optimize the base of a DOBS system—optimizing the base with respect to the information that the system has, given that its reasoner is likely non-ideal (not complete).

Chapter 3

Reconsideration

This chapter contains material that was published in [Johnson & Shapiro2005a]. It has been altered and expanded for clarity.

3.1 Introduction

3.1.1 Motivations and Assumptions

Motivation: Eliminating Negative Operation-Order Side Effects

As mentioned in Section 1.3.2, there are several motivations driving the research in this dissertation. One of the main motivations is the reduction (ideally, the elimination) of the negative side effects of operation order.

Assuming a Foundations Approach

The main assumption is that the reasoning agent or system is reasoning from and with a belief *base*—a finite set of core or base beliefs (also called hypotheses in [Martins & Shapiro1988]) that have independent standing and are treated differently from derived beliefs. This follows the foundations approach [Doyle1979, Gärdenfors1992, Hansson1993] discussed in Sections 1.2 (page 4) and 2.2.1 (page 19). A base belief is information entered into the belief base as input from some external source (external to the reasoning system; e.g., sensor readings, intelligence report, rules of a game, etc.). Beliefs that are derived through reasoning from such base beliefs are believed only if the base beliefs underlying their derivation are in the base. The currently believed base beliefs are the *foundation* for all the beliefs in the current belief space.

Since the belief base is finite, it is *not* deductively closed. Even though one could create a finite belief base that was "closed", in a sense, by selecting one belief from each set of logically equivalent beliefs in the logical closure (e.g., retain just a, as opposed to a, $a \vee a$, $a \wedge a$, etc.), this is impractical for a belief space of any size. It also violates one of the main purposes of the foundations approach, which is to set apart the base beliefs as the "foundation" upon which the reasoning agent's belief space stands. These base beliefs are the input to the agent's beliefs that come from some source of information other than reasoning and deduction, and they should be preserved that way.

Assuming a Linear Preference Ordering

In defining reconsideration, I make the assumption that there is a recency-independent, linear preference ordering ($\succeq$) over all base beliefs. Thus, any base can be represented as a unique sequence of beliefs in order of descending preference: $B = p_1, p_2, \ldots, p_n$, where $p_i \succ p_{i+1}, 1 \leq i < n$. Note: $p_i \succ p_j$ means that p_i is strictly preferred over p_j (is stronger than p_j) and is true IFF $p_i \succeq p_j$ and $p_j \not\succeq p_i$.

Limiting the ordering of base beliefs to a linear ordering is unrealistic for real world applications. I discuss non-linear orderings of base beliefs at the end of this chapter, but assume the linear ordering at this time to aid in theorems and proofs for this chapter and the next. Future work will address orderings that are not anti-symmetric and/or cannot guarantee comparability.

Because any finite set of beliefs whose elements are ordered linearly can also be represented as a sequence of beliefs in descending order of preference (or credibility), I will do so interchangeably in this dissertation—e.g. $B = p_1, p_2, \ldots, p_n$ means that $B = \{p_1, p_2, \ldots, p_n\}$ in a linear ordering where $p_i \succ p_{i+1}, 1 \leq i < n$. Unless otherwise stated, the elements in an ordered set will be ordered in descending order of preference (or credibility), as shown in the previous sentence.

Another assumption is that the agent can somehow order the incoming beliefs by credibility. This preference ordering is assumed to be attached to an incoming belief and reflects its importance (or credibility or reliability) *as a base belief*. Since base beliefs can be retracted, there is the question of how retractions affects the ordering. Therefore, relative ordering of base beliefs should be unaffected by the presence of other beliefs in the currently believed base *or* by whether these beliefs are derivable from other beliefs in the base *or* even by whether the beliefs are *in* the base.

Example 3.1.1 An example of a set of base beliefs that ordered from strongest to weakest is: $p \succ q \succ d$. Whether the set in the current base is $B1 = \{p,q\}$ or $B2 = \{q,d\}$, the ordering is unaltered, so q is the weaker element in $B1$ and the stronger element in $B2$. If the belief r is added to the set of base beliefs, the relative ordering of the other beliefs should be unaffected: it is still the case that $p \succ q \succ d$, regardless of where in this chain r gets placed.

My preference ordering is not the same as the epistemic entrenchment [Gärdenfors & Makinson1988, Gärdenfors1988] or ensconcement [Williams1994a] approaches. These orderings reflect the way that beliefs will survive belief change operations. One postulate for epistemic entrenchment, called Dominance, states that if $a \vdash b$, then $a \leq b$. This merely means that contraction by a may not require retracting b, but contraction by b forces the retraction of a, therefore b is at least as entrenched as a (possibly even *more* entrenched, in which case $a < b$).

Since the preference ordering is unaffected by the composition of the current base, any part that recency plays in the importance of an incoming belief should be reflected in its incoming ordering information. This means that even if recency is a factor, we can use the non-prioritized belief change operation of *kernel semi-revision* (expansion followed by kernel consolidation; as mentioned in Section 2.2.5) when adding beliefs while maintaining consistency. The result is that a belief being added to the base might not survive the consolidation step if it conflicts with stronger base beliefs.

Improving Recovery Adherence to Reduce Negative Operation-Order Side Effects

In the case of base theory change with consistency maintenance, the lack of adherence to the Recovery postulate [Alchourrón, Gärdenfors, & Makinson1985] results in the order of non-prioritized additions affecting the makeup of the resulting base. Improving the Recovery aspect of belief change operations on a belief base can reduce the negative operation order side effects resulting in a more reliable base.

In the remainder of this chapter, I discuss the side effects of operation order and how I implement hindsight reasoning much the way a jury uses a jury room to review the testimony in a case before rendering a decision. This hindsight reasoning is called reconsideration. I then present a lengthy analysis of the improved adherence to a reconsideration-altered formulation of the Recovery postulate.

The final sections include a discussion of reconsideration performed with base beliefs in a non-linear ordering followed by a detailed analysis of reconsideration performed on a total pre-order of base beliefs using SATEN. This analysis extends an example from [Williams & Sims2000] that compares six different adjustment strategies, which correspond to six different incision functions.

3.1.2 A Brief Example Showing Operation Order Side-Effects

The examples below illustrate the effect that operation order can have on a belief base.[1] Given the belief base $B0 = \{s, d\}$, whose beliefs are linearly ordered as $s > d$, the results of adding two more beliefs using kernel semi-revision are shown below—recall that $B0$ is an ordered set that can also be represented as a sequence: $B0 = s, d$.

Example1: The first belief added to $B0$ is $\neg(s \vee d)$, where $\neg(s \vee d) \succ s \succ d$. This forces the retraction of *both* s and d to maintain consistency. The second belief added is $s \vee d$, with $s \vee d \succ \neg(s \vee d)$. Now, $\neg(s \vee d)$ must be retracted for consistency maintenance. The resultant base is $B1 = \{s \vee d\}$.

Example2: If we reverse the order, however, of the last two additions—i.e., add $s \vee d$, then add $\neg(s \vee d)$, keeping the same overall preference ordering as before ($s \vee d \succ \neg(s \vee d) \succ s \succ d$)—the addition of $s \vee d$ would not force any retractions for consistency maintenance, and the following addition of $\neg(s \vee d)$ would "fail" in the sense that $\neg(s \vee d)$ would not be added to the base (because it directly contradicts the more preferred $s \vee d$). The resultant base would be $B2 = \{s \vee d, s, d\} = s \vee d, s, d$.

To aid in understanding these examples, consider the semantics of the atomic propositions to be the following (from [Hansson1991b]): s denotes "Cleopatra had a son" and d denotes "Cleopatra had a daughter." The two examples show the addition of two new beliefs in varying order but with the same preference ordering between them (and in relation to the existing beliefs). The new beliefs are "Cleopatra had no children" and the most preferred belief "Cleopatra had a child."[2]

The two examples differ only in the order that the additions take place, yet the resulting base in Example 2 ($B2$) contains s and d, whereas that of Example 1 ($B1$) does not. $B1$ is not considered optimal, because the return of both s and d would expand the base without introducing any inconsistency (since $\neg(s \vee d)$ has

[1]These examples are based on the Cleopatra's children example mentioned in [Hansson1991b].

[2]In this version, as in [Hansson1991b], "Cleopatra had *a* child" is assumed to mean that "Cleopatra was not childless" (i.e., she had *at least* one child), as opposed to "Cleopatra had a *single* child."

already been retracted and is not in the base). Note that $B2$ is also the base that would result if both beliefs had been added *before* consistency maintenance was triggered.

When reading about reconsideration later in this chapter, keep in mind that performing reconsideration at the end of Example 1 would return both s and d to the belief base. Reconsideration would not alter the base in Example 2.

3.1.3 Jury-room Reasoning

A person sitting on a jury during a trial is constantly performing belief change operations in series. Therefore, they are also susceptible to experiencing the negative side effects of operation order.

As a juror hears testimony, his beliefs about the facts in the trial might change. New evidence might cause him to retract a belief from earlier testimony if it conflicts, or perhaps the new testimony will not be accepted as true if the earlier conflicting testimony appears to be more credible (an example of kernel semi-revision).

In either case, the order of belief change operations—when done in series—affects the future belief base (and, therefore, belief space).

Jurists, however, are not limited to performing belief change operations in series. Before rendering a decision about a case, they retire to a jury room to review *all* the testimony and evidence. This is tantamount to a batch consolidation of *all* their beliefs, including those that were retracted earlier. During this deliberation, the jury might reconsider a previous retraction, and a previously removed base belief might be returned to the base.

In this case, returning a removed belief to the belief *base* is specifically *not* the same as just returning it to the belief *space*—a belief can return to the belief *space* if it is *derivable* from beliefs in the base (it need not, actually, be in the base, itself). Returning a removed belief to the base is a more impressive return.

If a KR&R system gains new information that, in hindsight, might have altered the outcome of an earlier belief change decision, the earlier decision should be re-examined. By consolidating over all past and current information, an optimal belief base is produced. I call this operation of "jury room reasoning" *reconsideration*.

3.1.4 Anticipating Reconsideration—Other Belief Change Operations that Appear Similar

In the next section, I define the operation of hindsight belief change that I call reconsideration—an operation that returns beliefs that were previously removed from the belief base when a more recent retraction eliminates any compelling reasons for them to remain disbelieved. This return is actually more of an exchange, because the return of some beliefs may require the removal of weaker ones to maintain consistency. I also explore how this exchange of beliefs optimizes the base w.r.t. all previous base beliefs and affects the Recovery aspect of base belief change.

The research on *belief liberation* [Booth *et al.*2005] also supports the concept that removing a belief from a base might allow some *previously* removed beliefs to return. Because of this common view, I cannot present the Recovery aspect of reconsideration without also examining the similarities and differences between belief liberation and reconsideration (see Section 3.2.4). I then use belief liberation terminology and reconsideration in separate formulations of the Recovery postulate; and I discuss and compare the specific cases that do (or do not) adhere to these formulations as well as to the traditional Recovery postulate for belief bases.

In addition to belief liberation, there are several other algorithms for belief change that mention returning base beliefs that were removed. However, for all these algorithms, this base belief return process is a second step in a two-step belief-change algorithm. For three of the algorithms, the first step selects a set of beliefs to remove from the base, and the second step then returns the removed base beliefs that are derivable from the retained ones. The algorithms that fall into this category include saturated kernel contraction [Hansson1994], safe contraction [Alchourrón & Makinson1985], and standard adjustment [Williams1994a]. Hybrid adjustment [Williams & Sims2000] also performs a two-step process to select what base beliefs to remove, but it favors saving beliefs that are not expressly *causing* an inconsistency, rather than just saving those *derivable* from the first cut for base belief removal. In all four cases mentioned, the two-step process, which includes returning some base beliefs, is all part of *one* single belief change operation: the removed beliefs that are considered for possible return were only the beliefs *just removed in step one*; their "return" is not evident as a result of the operation, because they were in the base when the operation began. The base beliefs that *are* removed by these algorithms are *not* available for return at some later date. Therefore, these algorithms are not likely to optimize the base w.r.t. *all* previously held base beliefs, and they cannot be reasonably compared to reconsideration.

3.2 Reconsideration

3.2.1 The Knowledge State for Reconsideration

The Knowledge State Triple

The knowledge state $\mathbf{B}$ used to formalize reconsideration is a tuple with three elements, $\langle B, B^{\cup}, \succeq \rangle$, where $B \subseteq B^{\cup} \subseteq L$, and $\succeq$ is a linear ordering over $B^{\cup}$. All triples are assumed to be in this form.

Starting with a new knowledge state $\mathbf{B}_0$ that has an empty base $B_0 = \emptyset$, B_n is the belief base that results from a series of n belief change operations on $\mathbf{B}_0$ (and the subsequent resulting knowledge states: $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$,...), and $B_n^{\cup} = \bigcup_{0 \leq i \leq n} B_i$. X_n is the set of base beliefs removed—and currently disbelieved: $(\forall n)\ B_n \cap X_n = \emptyset$ —from these bases during the course of the series of operations: $X_n =_{def} B_n^{\cup} \setminus B_n$.

The ordering changes with each new addition to the belief state, and how it changes is explained below.

For now, it is sufficient to say that $\succeq_n$ is the ordering for $B^{\cup}_n$.

The belief state after these n operations is $\langle B_n, B^{\cup}_n, \succeq_n \rangle$.

Credibility Values for Bases and Their Beliefs

The linear ordering ($\succeq$) can be numerically represented by assigning a numerical value to each belief in the ordering such that a more credible belief has a higher value. By making each belief twice as valuable as the belief immediately less credible than it in the ordering, every set of base beliefs can be associated with the sum of its beliefs' values. This sum is unique to *that* set—it cannot be matched by any other set of base beliefs. These values and sums are helpful when explaining how the preference ordering is used to optimize the knowledge state.

A numerical value for credibility of a belief $p_i \in B^{\cup} = p_1, p_2, \ldots, p_n$:

Definition 3.2.1 The credibility value for any belief $p_i \in B^{\cup} = p_1, p_2, \ldots, p_n$ is: $\mathrm{Cred}(p_i, B^{\cup}, \succeq) =_{def} 2^{n-i}$.

Note that $p_i \succ p_j$ IFF $\mathrm{Cred}(p_i, B^{\cup}, \succeq) > \mathrm{Cred}(p_j, B^{\cup}, \succeq)$.

A numerical value for credibility of a consistent base is calculated from the preference ordering of $B^{\cup} = p_1, p_2, \ldots, p_n$. An inconsistent base has a credibility value of -1.

Definition 3.2.2 $\mathrm{Cred}(B, B^{\cup}, \succeq) =_{def} \sum_{p_i \in B} 2^{n-i}$ (the bit vector, or bitmap, indicating the elements in B) when $B \nvdash \bot$. Otherwise, when $B \vdash \bot$, $\mathrm{Cred}(B, B^{\cup}, \succeq)$ = -1.

This means that the credibility value of a belief is equivalent to the credibility value of its singleton set (assuming that the belief is not, itself, an inconsistency): $\mathrm{Cred}(p_i, B^{\cup}, \succeq) = \mathrm{Cred}(\{p_i\}, B^{\cup}, \succeq)$. And the credibility value of a *consistent* set is equivalent to the sum of the credibility values of its beliefs (also equivalent to the sum of the credibility values of its singleton subsets): $\mathrm{Cred}(B, B^{\cup}, \succeq) = \sum_{p_i \in B} \mathrm{Cred}(p_i, B^{\cup}, \succeq)$.

Base	Bitmaps for beliefs indicated.						Consistent?	Value
	p	$\neg p$	q	$\neg q \vee p$	r	$\neg r \vee \neg p$		
$B^{\cup}$	1	1	1	1	1	1	NO	-1
$B1$	1	0	1	1	1	0	YES	46
$B2$	1	0	1	1	0	1	YES	45
$B3$	0	1	0	1	1	1	YES	23
$\{p\}$	1	0	0	0	0	0	YES	32
$\{\neg p\}$	0	1	0	0	0	0	YES	16
$\{q\}$	0	0	1	0	0	0	YES	8
$\{\neg q \vee p\}$	0	0	0	1	0	0	YES	4
$\{r\}$	0	0	0	0	1	0	YES	2
$\{\neg r \vee \neg p\}$	0	0	0	0	0	1	YES	1

Table 3.1: The credibility value of a base is its bitmap when the base is consistent, -1 otherwise. This value is also equivalent to the sum of the bitmaps for its singleton subsets (provided the base is consistent).

Example 3.2.3 Given $B^{\cup} = p, \neg p, q, \neg q \vee p, r, \neg r \vee \neg p$, Table 3.1 shows the bitmaps and credibility values for $B^{\cup}$ as well as for the following sets: $B1 = p, q, \neg q \vee p, r$, $B2 = p, q, \neg q \vee p, \neg r \vee \neg p$, $B3 = \neg p, \neg q \vee p, r, \neg r \vee \neg p$, and the singleton subsets of $B3$.

An ordering over bases ($\succeq_{\perp}$) that is dependent on credibility values is also defined:

Definition 3.2.4 $B \succeq_{\perp} B'$ IFF $\text{Cred}(B, B^{\cup}, \succeq) \geq \text{Cred}(B', B^{\cup}, \succeq)$.

Likewise, $B \succ_{\perp} B'$ IFF $\text{Cred}(B, B^{\cup}, \succeq) > \text{Cred}(B', B^{\cup}, \succeq)$. Note that $B \succ_{\perp} B'$ implies $B \succeq_{\perp} B'$.

Note that $\succeq_{\perp}$ provides a linear ordering over all *consistent* subsets of $B^{\cup}$, because the credibility value for any *consistent* base is unique—equivalent to the bitmap of its beliefs. Inconsistent subsets of $B^{\cup}$ cannot be included in this linear ordering, because they all have a credibility value of -1. Therefore, if there is at least one consistent subset of $B^{\cup}$, there is a unique maximal (greatest) consistent subset of $B^{\cup}$.

The following observation is a formalization of the concept that, when comparing two bases (assuming both are consistent), you only need to compare their sequences, belief by belief, from strongest to weakest, until you either come to the end of one sequence (belonging to the weaker base) or you find two mis-matched beliefs (the stronger belief is in the stronger base). This is identical to comparing their bitmaps, bit by bit, from left to right, and noting when the first mis-match (0 vs. 1) occurs—the base with the 1 is the stronger base.

Observation 3.2.5 *Given $B^{\cup}$, with elements ordered linearly by $\succeq$, and the two bases $B = b_1, b_2, \ldots, b_n$ and $B' = b'_1, b'_2, \ldots, b'_m$, where $B \nvdash \perp$, $B' \nvdash \perp$, $B \subseteq B^{\cup}$ and $B' \subseteq B^{\cup}$:*

if $Cred(B, B^{\cup}, \succeq) > Cred(B', B^{\cup}, \succeq)$, then one of the following must be true:

- $B' \subsetneq B$
- $(\exists i) : b_i \in B,\ b'_i \in B', b_i \succ b'_i$ *and* $b_j = b'_j$ *for* $1 \leq j < i$.

Subsequent chapters will include definitions for orderings over bases and knowledge state whose calculations are *not* dependent on consistency. These orderings will use (overload) the symbols $\succeq$ and $\succ$. That is why this ordering that is dependent on consistency includes $\perp$ as a subscript. For the remainder of this chapter, however, $\succeq$ will be used interchangeably with $\succeq_{\perp}$ and $\succ$ will also be used interchangeably with $\succ_{\perp}$. Only in later chapters will there need to be a distinction between consistency-oriented base (and knowledge state) orderings and other base orderings.

3.2.2 Defining an Optimal Base and an Optimal Knowledge State

An optimal base is a base that is preferred over any other base for some given $B^{\cup}$ and $\succeq$:

Definition 3.2.6 Given a possibly inconsistent set of base beliefs, $B^{\cup} = p_1, p_2, ..., p_n$, ordered by $\succeq$, the base B is considered *optimal* w.r.t. $B^{\cup}$ and $\succeq$ IFF $B \subseteq B^{\cup}$ and $(\forall B' \subseteq B^{\cup}) : B \succeq_{\perp} B'$. This is written as: Optimal-per-$\perp(B, B^{\cup}, \succeq) \equiv B \subseteq B^{\cup} \wedge (\forall B' \subseteq B^{\cup}) : B \succeq_{\perp} B'$.

An *optimal knowledge state* is a knowledge state whose base is optimal w.r.t. its $B^{\cup}$ and $\succeq$.

Definition 3.2.7 The $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$, is *optimal* IFF Optimal-per-$\perp(B, B^{\cup}, \succeq)$. This is written as Optimal-per-$\perp(\mathbf{B})$ IFF Optimal-per-$\perp(B, B^{\cup}, \succeq)$.

Essentially, if a base is optimal, no base belief p that is disbelieved ($p \in X$) can be returned without raising an inconsistency that can only be resolved by either the removal of p or the removal of some set of beliefs A containing a belief that is strictly stronger than p ($(\exists q \in A) : q \succ p$). This favors retaining a single strong belief over multiple weaker beliefs—an approach supported in the literature (cf. [Williams & Sims2000, Chopra, Georgatos, & Parikh2001, Booth *et al.*2005]). For this chapter, any reference to an optimal base is using Definition 3.2.6, and any reference to an optimal state is referring to Definition 3.2.7.

As in [Johnson & Shapiro2005a], the assumption is made that an operation of contraction or consolidation produces the new base B' by using a global decision function that maximizes Cred$(B', B^{\cup}, \succeq)$ *w.r.t. the operation being performed*. This means that contracting B by some belief p will result in a base B' that is a *subset* of B (equal to B if $p \notin B$) whose credibility value is the largest of all subsets of B *that do not derive* p—i.e. $(\forall B'' \subseteq B) : B'' \nvdash p \Rightarrow (B \sim p) \succeq B''$. Likewise, consolidating B results in a base B' that is a subset of B (equal to B if B is consistent) whose credibility value is the largest *of all subsets of* B. There may be some $B_2 \subseteq B^{\cup}$, however, that has a larger credibility value than either the result of $B \sim p$ or $B!$, because it contains some belief that is not in B (i.e. $B_2 \setminus B \neq \emptyset$), but this more credible base cannot be the result of either contraction or consolidation.

Maximizing $\mathrm{Cred}(B', B^{\cup}, \succeq)$ *without* concern for any specific operation (e.g., without the restriction that $B' \subseteq B$ or $B' \nvdash \perp$) would result in $B' = B^{\cup}!$.

For the purpose of this dissertation and unless otherwise stated, any maximizing decision function or operation returns the maximal base B which is assumed to be maximizing $\mathrm{Cred}(B, B^{\cup}, \succeq)$ for some known $B^{\cup}$ and $\succeq$—after first satisfying the conditions for the operation being performed. For a linear ordering of beliefs, maximizing a base is equivalent to finding *the unique* maximal base, which is also the greatest base (due to the linear ordering resulting in unique bitmaps for all bases). Likewise, there is only one, unique, minimally consistent base: the empty set ($\emptyset$).

Observation 3.2.8 *If consolidation uses a global decision function that finds the maximal consistent subset of a finite set of linearly ordered beliefs, then the consolidation of a base B is the greatest (unique maximal)* subset *of* that particular base[3] *(w.r.t. B and* $\succeq$*):* $B! \subseteq B$ *and* $(\forall B' \subseteq B) : B! \succeq B'$.

Observation 3.2.9 *If consolidation uses a global decision function that finds the maximal consistent subset of a finite set of linearly ordered beliefs, then the consolidation of* $B^{\cup}$ *produces the optimal base w.r.t.* $B^{\cup}$ *and its ordering:*

Optimal-per-$\perp$*(**B**) IFF* $B = B^{\cup}!$.

If a knowledge state is optimal, and its beliefs are ordered linearly with the optimal base being the maximal consistent subset of all the base beliefs as determined by the linear ordering, then the beliefs that are not in the optimal base (all $x \in X$) are each *individually inconsistent* with that base. Additionally, there is no set of beliefs that can be removed from $B + x$ that will restore consistency without removing some belief that is *stronger* than x (which is a bad trade, because it reduces the credibility value of the base). This is expressed in the theorem below.

[3]Since consolidation produces a subset of the existing base, the ordering of the beliefs in the base is all that is needed to measure the value of its subsets. The beliefs in $B^{\cup}$ that are not in B and their ordering information need not be known for this operation.

Theorem 3.2.10 *Given the knowledge state, $\boldsymbol{B} = \langle B, B^{\cup}, \succeq \rangle$ where $\succeq$ is a linear ordering of the beliefs in $B^{\cup}$ and base optimality is defined as the maximal (greatest) consistent subset of $B^{\cup}$, then:*

$\boldsymbol{B}$ is optimal $\Leftrightarrow (\forall x \in X) : B \not\vdash \perp,\ B + x \vdash \perp$, and $(\nexists B' \subseteq B)$ s.t. both $(B \setminus B') + x \not\vdash \perp$ and $(\forall b \in B') : x \succeq b$.

Proof.

$(\Rightarrow)$ **B** is optimal, so we know that $B \not\vdash \perp$ and $(\forall B' \subseteq B^{\cup}) : B \succeq B'$. Since $X \subseteq B^{\cup}$, $(\forall x \in X) : B + x \subseteq B^{\cup}$.

Prove $B + x \vdash \perp$: if $B + x \not\vdash \perp$, then $B \subsetneq B + x$ and Cred$(B + x, B^{\cup}, \succeq) >$Cred$(B, B^{\cup}, \succeq)$, thus $B + x \succ B$ making B non-optimal. Contradiction. $\square$

Prove $(\nexists B' \subseteq B)$ s.t. both $(B \setminus B') + x \not\vdash \perp$ and $(\forall b \in B')x \succeq b$: if $\nexists B' \subseteq B$ s.t. both $(B \setminus B') + x \not\vdash \perp$ and $(\forall b \in B')x \succeq b$, then Cred$((B \setminus B') + x, B^{\cup}, \succeq) >$Cred$(B, B^{\cup}, \succeq)$, thus $((B \setminus B') + x) \succ B$ making B non-optimal. Contradiction. $\square$

$(\Leftarrow)$ $(\forall x \in X) : B \not\vdash \perp,\ B + x \vdash \perp$, and $(\nexists B' \subseteq B)$ s.t. both $(B \setminus B') + x \not\vdash \perp$ and $(\forall b \in B') : x \succeq b$.

Therefore, $(\forall B'' \subseteq B^{\cup}) : B \succeq B''$. $\square$

3.2.3 Operations on a Knowledge State

The following are operations on the knowledge state $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$.

Expansion of B

Expansion of **B** by p and its preference information, $\succeq_p$, is: $\mathbf{B} + \langle p, \succeq_p \rangle =_{def} \langle B + p, B^{\cup} + p, \succeq_1 \rangle$, where $\succeq_1$ is $\succeq$ adjusted to incorporate the preference information $\succeq_p$—which positions p relative to other beliefs in $B^{\cup}$, while leaving the relative order of other beliefs in $B^{\cup}$ unchanged. The resulting ordering is the transitive closure of these relative orderings. Additionally, if $p \in B^{\cup}$, then the location of p in the sequence might change—i.e., its old ordering information is removed before adding $\succeq_p$ and performing closure—but all other beliefs remain in their same relative order.

The following example shows how a new belief is inserted into the linear ordering of $B^{\cup}$.

Example 3.2.11 Start with a knowledge state $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$, where $B = B^{\cup}!$ and $B^{\cup} = p, p \rightarrow q, \neg q$. Since $B = B^{\cup}!$, we know that $B = p, p \rightarrow q$. $\mathbf{B} + \langle q, \succeq_q \rangle = \mathbf{B}_1 = \langle B_1, B_1^{\cup}, \succeq_1 \rangle$, where $B_1 = B \cup \{q\}$, and $B_1^{\cup} = B^{\cup} \cup q$. If $\succeq_q$ states that $p \succ q \succ p \rightarrow q$, then the sequence $B^{\cup} = p, q, p \rightarrow q, \neg q$.

The next example shows how a belief p that is already in $B^{\cup}$ is added through expansion and relocated to a different position in the chain. Notice how the remaining elements in $B^{\cup}$ (specifically, the beliefs in $B^{\cup} \setminus \{p\}$) remain in the same order relative to each other.

Example 3.2.12 Start with a knowledge state $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$, where $B = B^{\cup}!$ and $B^{\cup} = s, t, p, p \rightarrow q, \neg q, m$. Since $B = B^{\cup}!$, we know that $B = s, t, p, p \rightarrow q, m$. $\mathbf{B} + \langle p, \succeq_p \rangle = \mathbf{B}_1 = \langle B_1, B_1^{\cup}, \succeq_1 \rangle$, where $B_1 = B \cup \{p\}$ and $B_1^{\cup} = B^{\cup} \cup p$.

If $\succeq_p$ states that $\neg q \succ p \succ m$, then $B_1^{\cup} = s, t, p \rightarrow q, \neg q, p, m$. The base remains unchanged ($B_1 = B$), because this is an operation of simple *expansion* and p was already in B. Only the ordering of the beliefs is changed.

Note that the set of all base beliefs has not changed—the *sets* $B_1^{\cup} = B^{\cup}$; but, since $\succeq_1 \neq \succeq$, the *sequences* for $B_1^{\cup}$ and $B^{\cup}$ are different. This distinction will be rare in the discussions in this dissertation.

The process of merging the existing ordering with incoming ordering information is an ongoing research topic. Whether the belief being added is a new belief or merely an existing base belief being relocated in the ordering, there needs to be a formalized process by which this merging can take place. For now, I will assume that the new ordering takes priority over the existing ordering and that its information is complete enough to precisely locate the new belief within the chain and no other beliefs are relocated with respect to each other. There is a discussion in Chapter 4 (cf. page 170) regarding re-ordering multiple beliefs in $B^{\cup}$, and for further discussion of merging orderings, cf. [Hansson1999, Meyer2000].

Contraction of B

Contraction of **B** by p is: $\mathbf{B} \sim p \ =_{def} \ \langle B \sim p, B^{\cup}, \succeq \rangle$, where "$\sim$" is some operation of base contraction.

In this dissertation, I focus on the base contraction operation of *kernel contraction*, which produces, essentially, a base resulting from the removal of at least one element from each p-kernel

Consolidation of B

Consolidation of **B** is: $\mathbf{B}! =_{def} \langle B!, B^{\cup}, \succeq \rangle$, where "!" is base consolidation as discussed in Section 2.2.5.

Again, I prefer to focus on *kernel* consolidation. Recalling the discussion in Observation 3.2.8: due to the beliefs in the base B being ordered linearly, a maximizing global decision function used for consolidation results in $B!$ being optimal w.r.t. B and $\succeq$. Note, again, that the optimality is w.r.t. B, not $B^{\cup}$; because there may be some belief in the base that *is* optimal w.r.t. $B^{\cup}$ that is *not* in B, and consolidation produces a *subset* of the base being consolidated.

Although SATEN's theory extraction is a form of consolidation, linear adjustment and standard adjustment definitely violate Core-Retainment, as discussed in Section 2.2.5. This is because beliefs that do not underlie the contradiction might get removed just because they are located at (or below, for standard adjustment) a contradictory ranking. Regarding optimality and assuming a linear ordering: SATEN would extract an optimal base using maxi-adjustment, hybrid adjustment, linear adjustment and quick adjustment—but *not* when using standard or global adjustments (cf. Section 3.5). [4] As stated above, the resulting base would be optimal w.r.t. the base that was being consolidated and its linear ordering. [5]

[4]Global adjustment could produce an optimal base if any intersecting $\perp$-kernels share the same weakest element. Standard adjustment would produce an optimal base only in the case where *all* its $\perp$-kernels share the same weakest element *and* that element is the weakest belief in the ordering.

[5]This assumes and a base that is not too large (nor a logic to complex) to prevent SAT testing from being used to perform consolidation.

Semi-revision of B

The semi-revision of **B** by the belief p and its preference information, $\succeq_p$, is:

$\mathbf{B} +_! \langle p, \succeq_p \rangle =_{def} \langle (B+p)!, B^{\cup} + p, \succeq_1 \rangle$,   where "+" is base expansion, "!" is base consolidation, and $\succeq_1$ is $\succeq$ adjusted to incorporate the preference information $\succeq_p$ as for expansion of a knowledge state (explained above).

Because of the consolidation of $B+p$, p might not actually "survive" the semi-revision addition into the base. It will, however, always be in the union set of bases (now $B^{\cup} \cup p$). This is an operation of non-prioritized belief revision, where the belief being added must survive the addition and consistency maintenance on its own merits, not just because it is the current belief being added (as in prioritized belief revision).

Again, I favor kernel semi-revision (as discussed in Section 2.2.5). The only difference between kernel semi-revision on a knowledge state and kernel semi-revision on a base is the additional update to $B^{\cup}$ and the ordering $\succeq$. Semi-revision on a knowledge state is susceptible to the same operation order side-effects mentioned for base semi-revision (cf. Example 3.1.2).

Reconsideration of B

Reconsideration of **B** [Johnson & Shapiro2005a] is: $\mathbf{B}!_{\cup} =_{def} \langle B^{\cup}!, B^{\cup}, \succeq \rangle = \langle B^{\cup}, B^{\cup}, \succeq \rangle!$.

Reconsideration replaces the current base B with a new base B' that is the consolidation of the current $B^{\cup}$. As a result of reconsideration, beliefs in $B^{\cup}$ that are not in B may be present in B', and beliefs that are in B may be absent from B'. In the case of these changes, B' is an improvement over B and the exchange of beliefs in $B \setminus B'$ for those in $B' \setminus B$ is considered *a good trade*. If consolidation guarantees to produce an optimal base, then so does reconsideration:

Theorem 3.2.13 *[Johnson & Shapiro2005a] Assuming a linear ordering on the beliefs in $B^{\cup}$ and a decision* *function for consolidation that results in an optimal subset of the base being consolidated, the base resulting from reconsideration of* $\boldsymbol{B}$ *is optimal w.r.t.* $B^{\cup}$ *and* $\succeq$.

Proof. $\langle B, B^{\cup}, \succeq \rangle \overset{!}{\cup} \;=_{def}\; \langle B^{\cup}!, B^{\cup}, \succeq \rangle$. $(\forall B' \subseteq B^{\cup}) : B^{\cup}! \succeq B'$. (from Obs. 3.2.8) □

Example 3.2.14 Recalling Example 3.2.12, above: start with a knowledge state $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$, where $B = B^{\cup}!$ and $B^{\cup} = s, t, p, p \rightarrow q, \neg q, m$. Since $B = B^{\cup}!$, we know that $B = s, t, p, p \rightarrow q, m$. $\mathbf{B} + \langle p, \succeq_p \rangle = \langle B \cup \{p\}, B^{\cup} \cup p, \succeq' \rangle$. If $\succeq_p$ states that $\neg q \succ p \succ m$, then $B^{\cup} = s, t, p \rightarrow q, \neg q, p, m$. B remains unchanged, because this is an operation of simple expansion and p was already in B. Only the ordering of the beliefs is changed.

However, the base is now sub-optimal, due to the relocation of the belief p in the ordering. Reordering beliefs creates a need to reconsider past belief change operations (such as the removal of $\neg q$ for consistency maintenance) in light of the new ordering. In this case, reconsideration will result in a base where $\neg q$ is believed and the now weaker, conflicting p is removed:

$\langle B, B^{\cup}, \succeq' \rangle \overset{!}{\cup} = \langle B', B^{\cup}, \succeq' \rangle$, where $B' = B^{\cup}! = \{s, t, p \rightarrow q, \neg q, m\}$.

The next observation states that reconsideration results in the optimal base (and, thus, the optimal knowledge state), regardless of the makeup of the belief base at the time that reconsideration is performed. In other words, the result of reconsideration on a knowledge state is independent of the base of that knowledge state.

Observation 3.2.15 Reworded from [Johnson & Shapiro2005a] *Given* any *knowledge state for* $B^{\cup}$ *with a linear ordering* $\succeq$ *over* $B^{\cup}$ *and a maximizing global decision function for consolidation (that finds the maximal consistent subset of the base being consolidated), reconsideration on that state produces* the optimal knowledge state*:* $(\forall B \subseteq B^{\cup}) : \langle B, B^{\cup}, \succeq \rangle \overset{!}{\cup} = \langle B_{opt}, B^{\cup}, \succeq \rangle$*, where* B_{opt} *is the optimal base w.r.t.* $B^{\cup}$ *and* $\succeq$.

Although the concept of contraction is formalized by many researchers, the most practical use for contraction is as a part of the revision process: adjusting a belief space to maintain consistency during the process of adding new information. The various operations of revision that were discussed in Chapter 2 (theory revision, base revision, and semi-revision) can all be expressed as expansion combined with contraction[6] or consolidation (recall that consolidation is a specific form of contraction). Contraction is rarely used by itself. Although there are scenarios one can imagine to justify contraction as a sole operation, typically, a belief is retracted when some other belief provides a reason for that belief's removal. As explained in [Chopra, Georgatos, & Parikh2001] "Agents do not lose beliefs without a reason: to drop [i.e., retract] the belief . . . p is to *revise* by some information that changes our reasoning."

In the same way that contraction is used with addition to maintain consistency, a system can use reconsideration following each addition to maintain optimality (assuming that the decision function can achieve optimality). This would be helpful when new beliefs are added to the base as well as when existing beliefs are relocated in the order. It also would be beneficial for additions and/or relocations of sets of beliefs—such as when databases are merged.

This pairing of addition followed by reconsideration is called *optimized-addition*.

Optimized-addition to B

expansion Optimized-addition to **B** of the pair $\langle p, \succeq_p \rangle$ is: $\mathbf{B} +_{\overset{!}{\cup}} \langle p, \succeq_p \rangle =_{def} (\langle B, B^{\cup}, \succeq \rangle + \langle p, \succeq_p \rangle)\overset{!}{\cup}$.

Optimized-addition is the pairing of the operation of expansion followed by reconsideration on a knowledge state. If $B^{\cup}$ and $\succeq$ are known (or obvious), I adopt the shorthand writing of: (1) $B +_{\overset{!}{\cup}} \langle p, \succeq_p \rangle$ to stand for $\langle B, B^{\cup}, \succeq \rangle +_{\overset{!}{\cup}} \langle p, \succeq_p \rangle$; and (2) $B^{\cup} +_{\overset{!}{\cup}} \langle p, \succeq_p \rangle$ to stand for $\langle B', B^{\cup}, \succeq \rangle +_{\overset{!}{\cup}} \langle p, \succeq_p \rangle$ for *any* $B' \subseteq B^{\cup}$ (recall Observation 3.2.15). If the effect of adjusting the ordering by $\succeq_p$ is also known (such as with ordering

[6] Thanks to the Levi Identity.

by recency), then $\langle p, \succeq_p \rangle$ can be reduced to p. Which shorthand version is used (1 or 2) depends on the situation and what is more intuitive to the author/reader of the text where the shorthand is being used. An example of using this shorthand can be seen in the (OR) column in Table 3.2.

Observation 3.2.16 *Optimized-addition does not guarantee that the belief added will be in the optimized base—it might be removed during reconsideration.*

Example Let $B^{\cup} = p, p \rightarrow q, \neg q, p \rightarrow r, \neg r, m \rightarrow r, m$. And assume that $B = B^{\cup}! = p, p \rightarrow q, p \rightarrow r, m \rightarrow r, m$. $\langle B, B^{\cup}, \succeq \rangle +_{\cup!} \langle \neg p, \succeq_{\neg p} \rangle = \neg p, p \rightarrow q, \neg q, p \rightarrow r, \neg r, m \rightarrow r$, assuming that $\succeq_{\neg p}$ indicates $\neg p \succ p$. Notice the return of $\neg q$ and $\neg r$ to the base due to the removal of p, and the simultaneous removal of m to avoid a contradiction with $\neg r$ and $m \rightarrow r$. If, on the other hand, $\succeq_{\neg p}$ indicated $p \succ \neg p$, then the base would have remained unchanged.

Assuming a linear ordering of beliefs, the expansion of an optimal knowledge state by a belief that is consistent with the base but is not actually *in* the base is equivalent to the optimized addition of that belief—i.e., expansion by a new and consistent belief results in a new and *optimal* knowledge state. This is expressed in theorem 3.2.17.

Theorem 3.2.17 *Given $\boldsymbol{B} = \langle B, B^{\cup}, \succeq \rangle$, where $B = B^{\cup}!$, $X = B^{\cup} \setminus B$, $B^{\cup}$ is ordered linearly by $\succeq$, and $B^{\cup}!$ is the greatest (unique maximal) consistent subset of $B^{\cup}$ (as determined by $\succeq$), then $(\forall p$ s.t. $B + p \nvdash \perp$ and $p \notin B)$:*

$$\boldsymbol{B} + \langle p, \succeq_p \rangle = \langle B + p, B^{\cup} + p, \succeq' \rangle = \langle (B^{\cup} + p)!\ , B^{\cup} + p, \succeq' \rangle_{\cup!} = \langle B + p, B^{\cup} + p, \succeq' \rangle_{\cup!}\ .$$

Proof.

Since $B = B^{\cup}!$, we know that $(\forall x \in X) : B + x \vdash \perp$ and $(\nexists B' \subseteq B)$ s.t. both $(B \setminus B') + x \nvdash \perp$ and $(\forall b \in B')x \succeq b$ (Theorem 3.2.10). Therefore, since $B + p \nvdash \perp$, $p \notin X$, and $B + p \cup X = B^{\cup} + p$. Now, $(\forall x \in X) : (B + p) + x \vdash \perp$ and $(\nexists B' \subseteq (B + p))$ s.t. both $((B + p) \setminus B') + x \nvdash \perp$ and $(\forall b \in B')x \succeq b$ (from Theorem 3.2.10). Therefore, $(\forall B'' \subseteq (B^{\cup} + p)) : (B + p) \succeq_{B^{\cup} + p} B''$. Thus, $B + p = (B^{\cup} + p)!$. $\square$

In the following two examples, the belief being added p is not already an element of $B^{\cup}$. In the first example, p is also not in the belief space $Cn(B)$. In the second example, p is derivable from B. In the examples below, B is represented in set notation ($\{p_1, \ldots, p_n\}$) rather than in sequence notation ($p_1, \ldots, p_n$) expressly because the final ordering after expansion ($\succeq'$) is unknown and immaterial to this theorem.

Example 3.2.18 Given $B^{\cup} = q, q \rightarrow r, \neg r$, the optimal knowledge state is $\mathbf{B1} = \langle\{q, q \rightarrow r\}, B^{\cup}, \succeq\rangle$.

$\mathbf{B2} = \mathbf{B1} + \langle p, \succeq_p\rangle = \langle\{q, q \rightarrow r\} \cup \{p\}, B^{\cup} \cup \{p\}, \succeq'\rangle$. $\mathbf{B2}\overset{!}{\cup} = \langle\{q, q \rightarrow r, p\}, B^{\cup} \cup \{p\}, \succeq'\rangle$.

Example 3.2.19 Given $B^{\cup} = q, q \rightarrow p, \neg p$, the optimal knowledge state is $\mathbf{B3} = \langle\{q, q \rightarrow p\}, B^{\cup}, \succeq\rangle$.

$\mathbf{B4}$= $\mathbf{B3}$$+\langle p, \succeq_p\rangle = \langle\{q, q \rightarrow p\} \cup \{p\}, B^{\cup} \cup \{p\}, \succeq'\rangle$. $\mathbf{B4}\overset{!}{\cup} = \langle\{q, q \rightarrow p, p\}, B^{\cup} \cup \{p\}, \succeq'\rangle$.

In Example 3.2.19, even if p is added below $\neg p$ in the ordering, p would survive reconsideration, because all elements of a single p-kernel, $\{q, q \rightarrow p\}$, are higher than $\neg p$ (or any of its kernels) in the ordering. If $\neg p$ was higher than either q or $q \rightarrow p$, then p would not survive reconsideration unless it was inserted above $\neg p$ in the ordering; however, in this case, $\neg p$ would be in the original base and expanding by p would cause a contradiction and, thus, violate the premise of Theorem 3.2.17, so expansion by p is *not* guaranteed to produce an optimal base.

Assuming a linear ordering of beliefs ($\succeq$), expansion of an *optimal* knowledge state with base B by a belief $p \in B$ (provided p is *not* being *re*located to a *less credible* position in the ordering) results in a knowledge state whose base remains unchanged and is *optimal* w.r.t. the set of all base beliefs and the new ordering. This is expressed in the theorem below. Note: The belief p survives the optimizing step of reconsideration because of the location of p in the ordering $\succeq'$. If p were to be relocated to a lower preference in that ordering, then p might *not* survive reconsideration.

Theorem 3.2.20 *Given an optimal knowledge state* $\boldsymbol{B} = \langle B, B^{\cup}, \succeq\rangle$, *where* $X = B^{\cup} \setminus B$, *and* $\succeq$ *is a linear ordering, then* $(\forall p \in B)$: $\boldsymbol{B} + \langle p, \succeq_p\rangle = \boldsymbol{B}_1 = \langle B_1\ ,\ B_1^{\cup}\ ,\ \succeq_1\rangle$, *where* $B_1 = B + p$, *and the* sets $B_1^{\cup} = B^{\cup} + p$.

If $\succeq_1$ *is ordered such that* $p = p_i \in B^{\cup} = p_j \in B_1^{\cup}$ *and* $j \leq i$*, then* p *is not relocated to a weaker position in the ordering, and* $B_1 = B$*, the* sets $B_1^{\cup} = B^{\cup}$*, and* $\boldsymbol{B}_1 = \langle B_1,\ B_1^{\cup}\ ,\ \ \succeq_1\rangle_{\cup}^{!}$ *and is optimal w.r.t.* $B_1^{\cup}$ *and* $\succeq_1$.

Proof.

Because $\mathbf{B}$ is optimal, we know $B = B^{\cup}!$. So, we also know that $(\forall x \in X) : B + x \vdash \perp$ and $(\nexists B' \subseteq B)$ s.t. both (1) $(B \setminus B') + x \nvdash \perp$ and (2) $(\forall b \in B') x \succeq b$ (from Theorem 3.2.10). Therefore, since $p \in B$, we know that $B + p = B$, $B^{\cup} + p = B^{\cup}$, and $p \notin X$.

Since $p = p_i \in B^{\cup} = p_j \in B_1^{\cup}$ and $j \leq i$, we know

1. $(\forall q \in B^{\cup}) : p \succ q \Rightarrow p \succ_1 q$, because p did not move to a weaker position in the ordering, and
2. all other beliefs kept their same relative order— $(\forall q, r \in B^{\cup}) : q \succ r \Rightarrow q \succ_1 r$.

Therefore, $(\forall x \in X_1) : B_1 + x \vdash \perp$ and $(\nexists B' \subseteq B_1)$ s.t. both (1) $(B_1 \setminus B') + x \nvdash \perp$ and (2) $(\forall b \in B') : x \succeq_1 b$. And, from Theorem 3.2.10, B_1 is optimal w.r.t. $B_1^{\cup}$ and $\succeq_1$. Therefore, $(\forall B' \subseteq B_1^{\cup}) : B_1 \succeq_1 B'$. And, $\mathbf{B}_1 = \langle B_1, B_1^{\cup}, \succeq_1\rangle_{\cup}^{!}$. □

Example 3.2.21 Given $B^{\cup} = p, q, q \rightarrow r, \neg r$, the optimal knowledge state is $\mathbf{B5} = \langle \{q, q \rightarrow r\}, B^{\cup}, \succeq \rangle$. **B6=** **B5**$+\langle q, \succeq_q \rangle = \langle B, B^{\cup}, \succeq' \rangle$. But is B optimal, now?

If the ordering of $\succeq'$ is $p \succ' q \rightarrow r \succ' \neg r \succ' q$, then the optimal base w.r.t. $B^{\cup}$ and $\succeq'$ is $\{p, q \rightarrow r, \neg r\}$. B is no longer optimal, because q has been relocated below $\neg r$ in the ordering.

If the ordering of $\succeq'$ is $p \succ' q \succ' q \rightarrow r \succ' \neg r$ or $q \succ' p \succ' q \rightarrow r \succ' \neg r$, the base is unchanged by the expansion.

If the ordering of $\succeq'$ is $p \succ' q \rightarrow r \succ' q \succ' \neg r$, then the base will also be unchanged by the expansion. Notice that, in this case, q is lowered in the ordering, but the base remains unchanged. That is because q is still above $\neg r$. However, when $q \in B$ and B is optimal, only if q is *not* lowered in the ordering can expansion be *guaranteed* to result in an optimal base.

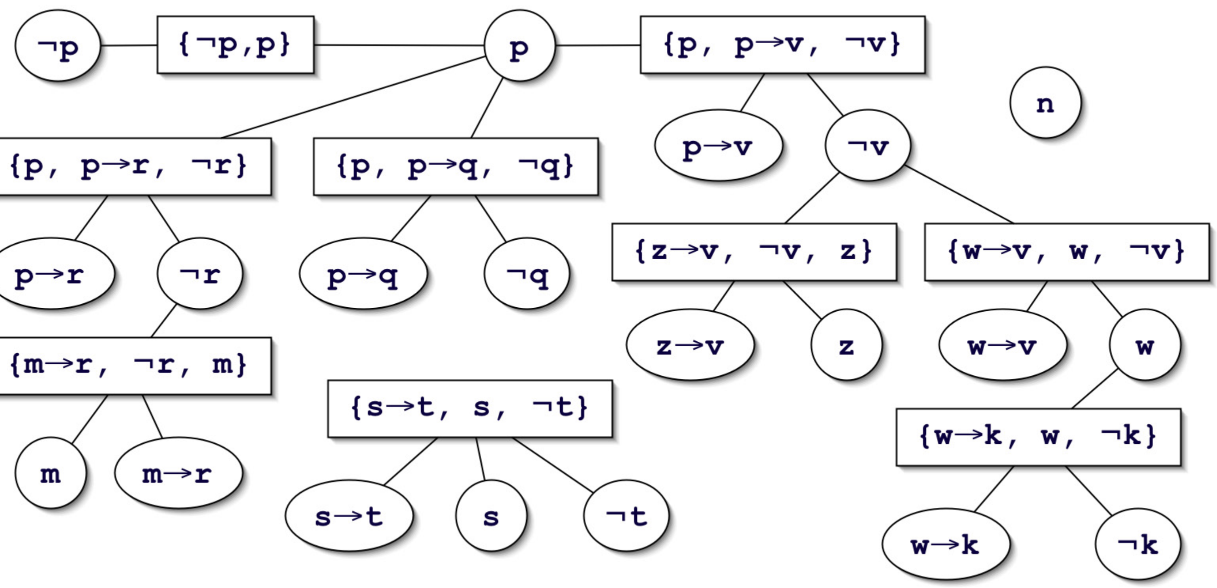

Figure 3.1: A graph showing the elements of $B^{\cup}$ (circles/ovals) of a knowledge state connected to their respective nogoods (rectangles). Consider the optimal knowledge state $\mathbf{B}_1$ where $\neg p$ was not yet in the set of all base beliefs: $\mathbf{B}_1 = \langle B_1, B_1^{\cup}, \succeq_1 \rangle$. In this case, $\succeq_1$ would order $B_1^{\cup}$ in the following way: $B_1^{\cup} = p, p \rightarrow q, p \rightarrow r, m \rightarrow r, s \rightarrow t, w \rightarrow v, w \rightarrow k, p \rightarrow v, z \rightarrow v, n, \neg r, w, s, \neg v, m, z, \neg q, \neg t, \neg k$. Therefore, the optimal base for $\mathbf{B}_1$ would be $B_1 = \{p, p \rightarrow q, p \rightarrow r, m \rightarrow r, s \rightarrow t, w \rightarrow v, w \rightarrow k, p \rightarrow v, z \rightarrow v, n, w, s, m, z\}$. The semi-revison addition of $\neg p$ (preferred over p) followed by reconsideration is described in Example 3.2.22 in the text.

A Reconsideration Example

Example 3.2.22 If we start with an optimal knowledge state $\mathbf{B}_1$, where $B_1^{\cup} = p, p \rightarrow q, p \rightarrow r, m \rightarrow r, s \rightarrow t, w \rightarrow v, w \rightarrow k, p \rightarrow v, z \rightarrow v, n, \neg r, w, s, \neg v, m, z, \neg q, \neg t, \neg k$. The optimal base would be $B_1 = p, p \rightarrow q, p \rightarrow r, m \rightarrow r, s \rightarrow t, w \rightarrow v, w \rightarrow k, p \rightarrow v, z \rightarrow v, n, w, s, m, z$, as described in Figure 3.1. The semi-revision addition of $\neg p$ (where $\neg p \succ p$) forces the retraction of p to maintain consistency. NOTE: Most systems would stop here.

However, the belief base optimizing operation of reconsideration produces the following changes: (1) $\neg q$ and $\neg r$ return to the base, and (2) m is removed (for consistency maintenance; to allow $\neg r$ to return; a "good trade"). The "recovery for bases" flavor of reconsideration is seen if we add p back into the base at a *stronger* preference than $\neg p$: $p \succ \neg p$. Reconsideration would return p, remove $\neg p$, remove $\neg q$, and remove $\neg r$ while returning m to the base. The resulting base would be B_1.

Implementing Reconsideration

Later in this chapter, there will be a discussion about testing a naive application of reconsideration using an online BR engine called SATEN [Williams & Sims2000]. I show and discuss the results of the different adjustment strategies available in SATEN performing reconsideration.

The next chapter fully presents and discusses a TMS-friendly, anytime algorithm that implements reconsideration in a dependency-directed technique that is more efficient than a naive implementation. This algorithm was first presented in [Johnson & Shapiro2005a].

3.2.4 Comparing Reconsideration To Belief Liberation

Key Difference Between Liberation and Reconsideration

Reconsideration was intended specifically to improve the optimality of a specific belief base and to impart an aspect of Recovery. The research in belief liberation [Booth *et al.*2005] focuses on defining liberation operators for some belief theory K relative to some *arbitrary* σ (essentially, some arbitrary linear ordering whose closure is the belief theory K). The focus is on K and how it changes when a contraction is performed—whether there is *any* σ that shows that a given contraction operation is an operation of σ-liberation. The authors do not advocate maintaining any one, specific σ. Although it is clearly stated that σ-liberation does not adhere to Recovery, the similarity between σ-liberation and reconsideration—the fact that both favor belief change operations possibly triggering the return of some subset of previously removed base beliefs—prompted us to compare them in detail.

Similarities to Reconsideration

Assume $B^{\cup} = [[\sigma]]$ and is ordered by recency; and the belief theory associated with σ as K_{σ} (my term).

$\mathcal{B}_n(\sigma)$ is the maximal consistent subset of $[[\sigma]]$—i.e., $\mathcal{B}_n(\sigma) = [[\sigma]]! = B^{\cup}!$. Similarly, $\mathcal{B}_n(\sigma, p)$ is the kernel contraction of $[[\sigma]]$ by p. In other words, $\mathcal{B}_n(\sigma, p) = B^{\cup} \sim p$.[7] Thus, $K \sim_{\sigma} p = Cn(B^{\cup} \sim p)$.

If we assume that $B = B^{\cup}! = \mathcal{B}_n(\sigma)$, I define σ_B to be the recency ordering of *just* the beliefs in $\mathcal{B}_n(\sigma)$, and $K_{\sigma} = K_{\sigma_B}$. Now I can define contraction of an *optimal* knowledge state in terms of contraction for σ-liberation: $B \sim p = (K_{\sigma} \sim_{\sigma_B} p) \cap B$ and $Cn(\langle B, B^{\cup}, \succeq\rangle \sim p) = K_{\sigma} \sim_{\sigma_B} p$.

This contraction operation produces different results from that discussed in [Booth *et al.*2005]: $K \sim p = K \sim_{\sigma} p \cap K$, where *any beliefs removed* by $K \sim_{\sigma} p$, whether to retract p or for adjustment to belief liberation, may be absent from the resulting belief space. In the contraction version that I use, which is equivalent to base kernel contraction, no beliefs can return (because the sequence is σ_B—there are no other beliefs) and, therefore, no removals occur except those to achieve retraction of p.

I define the operation of σ-*addition* (adding a belief to the belief sequence σ) in the same way that sequence expansion is defined in [Chopra, Georgatos, & Parikh2001]: $\sigma + p$ is adding the belief p to the sequence $\sigma = p_1, \ldots, p_n$ to produce the new sequence $\sigma_1 = p, p_1, \ldots, p_n$.

If σ is the sequence for $B^{\cup}$, then the optimized-addition of p to any knowledge state for $B^{\cup}$ results in a base equivalent to the base associated with the sequence resulting from the addition of p to σ: Given $B^{\cup} +_{\cup\!!} p = \langle B', B^{\cup} + p, \succeq'\rangle$, then $B' = \mathcal{B}_{n+1}(\sigma + p)$.[8]

Likewise, σ-addition followed by recalculation of the belief theory is equivalent to optimized-addition followed by closure: Given $B^{\cup} = \sigma$, $K_{\sigma+p} = Cn(B^{\cup} +_{\cup\!!} p)$.

[7]Note: specifically *not* $\mathcal{B}_n(\sigma, p) = B^{\cup}! \sim p$.

[8]My notation for the base associated with a σ-addition is not inconsistent with the notation of [Booth *et al.*2005] for the base associated with a σ-liberation operation. Addition changes the sequence, so I am determining the base for the *new* sequence $(\sigma + p)$: $\mathcal{B}(\sigma + p)$. The operation of σ-liberation changes the *base* that is used to determine the belief theory (from $\mathcal{B}(\sigma)$ to $\mathcal{B}(\sigma, p)$), but the sequence σ remains unchanged.

Cascading Belief Status Effects of Liberation

It is important to realize that there is a potential cascade of belief status changes (both liberations *and* retractions) as the belief theory resulting from a σ-liberation operation of retracting a belief p is determined; and these changes cannot be anticipated by looking at *only* the nogoods and kernels for p.

Example Let $\sigma = p \rightarrow q, p, \neg p \wedge \neg q, r \rightarrow p \vee q, r, \neg r$. Then, $\mathcal{B}_6(\sigma) = \{p \rightarrow q, p, r \rightarrow p \vee q, r\}$. Note that $r \in K_\sigma$ and $\neg r \notin K_\sigma$. $K \sim_\sigma p = Cn(\{p \rightarrow q, \neg p \wedge \neg q, r \rightarrow p \vee q, \neg r\})$. Even though r is not in a p-kernel in $[[\sigma]]$, $r \notin K \sim_\sigma p$. Likewise, $\neg r$ is liberated even though $\nexists N$ s.t. N is a nogood in $[[\sigma]]$ and $\{\neg r, p\} \subseteq N$.

Reconsideration has an identical effect. If $B^{\cup} = \sigma$, and $B = B^{\cup}! = \mathcal{B}_6(\sigma)$, then $\langle B, B^{\cup}, \succeq\rangle +_{\natural} \langle \neg p, \succeq_{\neg p}\rangle$, where $\succeq_{\neg p}$ indicates $\neg p \succ p$, would result in the base $B_1 = \{\neg p, p \rightarrow q, \neg p \wedge \neg q, r \rightarrow p \vee q, \neg r\}$.

3.3 Improving Recovery for Belief Bases

3.3.1 Reconsideration Provides an Improved Recovery *Aspect* to Base Belief Change

Recovery is a Postulate Constraining the Operation of Contraction

The Recovery postulate is a *contraction* postulate: it constrains the operation of theory contraction; it is an axiom to which its authors recommend any theory contraction operation should adhere. Recovery states that, for a theory K and a belief p, $K \subseteq (K - p) + p$, where $K - p$ is theory contraction and $K + p$ is theory expansion.

Base Recovery says that $B \subseteq Cn((B - p) + p)$, where $B - p$ is base contraction and $B + p$ is base expansion. Note here that the base is recovered in the belief space. It is not required to be in the revised base (i.e., Base Recovery does not require that $B \subseteq (B - p) + p$).

Comparing whether different belief change operations adhere to Recovery is limited to comparing different operations of *contraction*—by checking whether an expansion by some belief p that immediately follows contraction by p results in a superset of the original theory. Any contraction of a belief *base* (when defined to result in some subset of that base) cannot guarantee to adhere to Recovery. Since contraction of a knowledge state is defined by contraction of its base, knowledge state contraction cannot adhere to Recovery.

This section discusses how performing reconsideration *following* the contraction-expansion sequence of operations in the Recovery axiom increases the number of cases showing the benefits of Recovery. Reconsideration does not perform any deductive closure, it merely shows an alternate approach to gain some of the benefits that Recovery provides.

Benefits of Recovery

As mentioned above, Recovery constrains theory contraction. The goal is to remove as little as possible from a theory to accomplish the retraction of some belief p. So little should be removed, in fact, that if p were returned to the theory (and deductive closure performed), all beliefs removed during the contraction would be returned.

So there are two main benefits resulting from adherence to Recovery:

1. One benefit of a contraction adhering to Recovery, obviously, is that the set of beliefs removed during the contraction is minimized.
2. A second benefit of adherence to Recovery, however, is this sense that a contraction by a belief can be *undone* if immediately followed by expansion by that same belief (undone in the sense that it is returned to the belief space—which, for a theory, is indistinguishable from its base).

Although base contraction does not adhere to Recovery, assuming a linear ordering guarantees that any foundations-oriented contraction operation that maximizes the resulting base (provided it adheres to the Success postulate for base contraction, see Section 2.2.4) will minimize the set of beliefs it removes from the base. So the first benefit of adherence to Recovery is satisfied.

The second benefit of Recovery is realized (in most cases) by performing reconsideration after each expansion (expansion and reconsideration performed in series is called optimized-addition; cf. Section 3.2.3). Given a base B and some belief p, contraction by p followed by expansion by p followed by reconsideration of the knowledge state will return a knowledge state whose base B' is a superset of B (i.e., $B \subseteq B'$)—*except* in the case where $\neg p$ is in the original belief space (i.e.$\neg p \in Cn(B)$).

The exception just noted is trivial, because it occurs in the case where expansion produces an inconsistent set whose closure (for classical logic) is the set of all beliefs—i.e., *all* theories would be a subset of that inconsistent set (due to closure and assuming we are not using a paraconsistent logic). Reconsideration removes the contradiction by (typically, though not always; discussion in a later section) retracting $\neg p$, which means that the original base will not be a subset of the post-reconsideration base or its closure.

Reconsideration improves the aspect of Recovery for belief bases, but not to the extent of allowing an inconsistent base.[9] Additionally, in an implementation, contraction would rarely be used unless the belief being contracted was actually *in* the current belief space (and, therefore, its negation would *not* be in the belief space—assuming B is consistent), so this exceptional case is of lesser importance when attempting to realize the benefits of Recovery for practical, real-world applications.

The following section includes an extensive discussion regarding the realization of this second benefit of Recovery for various cases and various Recovery-like formulations.

[9]The aspects of Recovery that I refer to are, in actuality, criteria that are associated with the Recovery postulate—the *goals* of the Recovery postulate. I refer to the shift from satisfying less important goals to more important goals as an improvement in the aspect of Recovery that can be associated with belief base operations. This shift can be seen in the upcoming Table 3.2 on page 93.

3.3.2 Comparing Recovery-like Formulations

This section explores and discusses Recovery-like formulations, and whether they are adhered to during base belief change.

Let $\mathbf{B} = \langle B, B^{\cup}, \succeq \rangle$, s.t. $B^{\cup} = [[\sigma]]$, $B = B^{\cup}! = \mathcal{B}_n(\sigma)$, $K = K_{\sigma} = Cn(B)$, $\mathbf{P} =$ the set of p-kernels in B, $\mathbf{p} = \langle p, \succeq_p \rangle$, $\mathbf{B}_1 = \langle B_1, B_1^{\cup}, \succeq_1 \rangle = (\mathbf{B} \sim p) +_{\Cup} \mathbf{p}$, and $X_1 = B_1^{\cup} \setminus B_1$. The first element in any knowledge state triple is recognized as the currently believed base of that triple (e.g., B in $\mathbf{B}$), and is the default set for any shorthand set notation formula using that triple (e.g., $A \subseteq \mathbf{B}$ means $A \subseteq B$). Note that the assumptions above include the assumption that B is optimal w.r.t. $B^{\cup}$ and $\succeq$.

Table 3.2 shows the cases where different Recovery formulations hold—and where they do *not* hold. There is a column for each formulation and a row for each case. The traditional Recovery postulate for bases $(Cn(B) \subseteq Cn((B \sim p) + p))$ is shown in column (TR). In column (LR), the recovery postulate for σ-liberation retraction followed by expansion (Liberation-recovery, LR, our term) is: $K \subseteq ((K \sim_{\sigma} p) + p)$.

In column (OR), the recovery-like formulation for kernel contraction followed by optimized-addition is: $K \subseteq Cn((B \sim p) +_{\Cup} \mathbf{p})$. I call this Optimized-recovery, OR, and it is formulated this way to make comparisons to TR and LR more intuitive. As it turns out, adherence to OR occurs because of adherence to the more demanding axiom of Strict Base Optimized Recovery: $B \subseteq ((B \sim p) +_{\Cup} \mathbf{p})$. This is more demanding than OR, because *base beliefs* are recovered in the *base*, itself, not just its *closure*.

There are *two* columns devoted to OR. For column (OR-i), the ordering for $B^{\cup}$ and $B_1^{\cup}$ is recency. For column (OR-ii) the assumptions include:

1. the ordering is *not* recency-based,
2. $p \in B^{\cup}$ (nullifying Case 3), and
3. optimized-addition returns p to the sequence in its *original* place (i.e., $\succeq = \succeq_1$).

Note that (OR) is not a *true* Recovery axiom for some contraction operation—i.e., it does not describe the result of contraction followed by expansion, but rather describes the desired result of contraction followed by expansion followed by reconsideration; it adds an additional operation (reconsideration) after the contraction-expansion pair. (OR) can also be rewritten as $K \subseteq Cn(\ ((B \sim p) + p)!_{\cup}\ )$, where reconsideration is performed *after* the expansion but *before* the closure to form the new belief space.

An entry of YES in the table means the formulation *always* holds for that given case; NO means it does not *always* hold; NA means the given case is not possible for that column's conditions. The second entry in each cell indicates whether the resulting belief space is guaranteed to be optimal (i.e., is the base from which the belief space is calculated optimal w.r.t. $B^{\cup}_{1}$ $(= B^{\cup} + p = \sigma + p)$ and its linear order). If not optimal, then a designation for consistency is indicated. Recall that optimality requires consistency.

Case 1 In this simple case, $\{p\}$ is the sole p-kernel in B. There are no other ways to derive p, therefore, retraction of p removes no other beliefs except p. For all formulations, p is removed then returned to the base, therefore all cases hold.

For (TR): $B \sim p = B \setminus \{p\}$, and $(B \sim p) + p) = B$.

For (LR): Since $\mathcal{B}_n(\sigma) \setminus \mathcal{B}_n(\sigma, p) = \{p\}$, $\mathcal{B}_n(\sigma) \subseteq \mathcal{B}_n(\sigma, p) \cup \{p\}$ and $K \subseteq (K \sim_{\sigma} p) + p$. If no beliefs are liberated by $K \sim_{\sigma} p$, then $\mathcal{B}_n(\sigma, p) = \mathcal{B}_n(\sigma) \setminus p$, and $(K \sim_{\sigma} p) + p = K$. If beliefs are liberated by $K \sim_{\sigma} p$, then $(\exists q \in \mathcal{B}_n(\sigma, p) \setminus \mathcal{B}_n(\sigma)) : \mathcal{B}_n(\sigma) + q \vdash \perp$. Therefore, $\mathcal{B}_n(\sigma, p) + p \vdash \perp$, so $(K \sim_{\sigma} p) + p \vdash \perp$ and Liberation-recovery holds.

For (OR), since $p \in B$, $(B \sim p) + p = B$, and OR holds. Additionally, for (OR-ii), $\succeq = \succeq_1$ and $B^{\cup} = B^{\cup}_{1}$, so $\mathbf{B} = \mathbf{B}_1$.

Case 2 Case 2 says that $p \in Cn(B)$ and there are p-kernels other than $\{p\}$ in B. Since there are p-kernels in B that consist of beliefs other than p, beliefs other than p must be removed from the base during contraction by p.

For (TR), if those p-kernel beliefs are not derivable from $(B \sim p) + p)$, then TR does not hold. For

		(TR)	(LR)	(OR)	
Case		$K \subseteq Cn((B \sim p) + p)$	$K \subseteq Cn((K \sim_\sigma p) + p)$	$K \subseteq Cn((B \sim p) +_{\cup!} \mathbf{p})$ but also $B \subseteq (B \sim p) +_{\cup!} \mathbf{p}$	
		recency	recency	(i) recency	(ii) $p \in B^\cup$ and $\succeq_1 = \succeq$
1.	$p \in Cn(B)$; $\mathbf{P} = \{\{p\}\}$	YES **optimal**	YES possibly inconsistent	YES **optimal**	YES **optimal**
2.	$p \in Cn(B)$; $\mathbf{P} \setminus \{\{p\}\} \neq \emptyset$	NO consistent	NO possibly inconsistent	YES **optimal**	YES **optimal**
3.	$p \notin Cn(B)$; $B + p \nvdash \perp$	YES **optimal**	YES **optimal**	YES **optimal**	NA
4.	$p \notin Cn(B)$; $B + p \vdash \perp$	YES inconsistent	YES inconsistent	NO **optimal**	YES **optimal**

Table 3.2: This table indicates whether each of three different Recovery formulations (TR, LR and OR) holds in each of four different cases (which comprise all possible states of belief). $K = Cn(B)$, $\mathbf{p} = \langle p, \succeq_p \rangle$, and $\mathbf{P}$ is the set of p-kernels. YES means the formulation *always* holds for that given case; NO means it does not *always* hold; NA means the given case is not possible for that column's conditions. See the text for a detailed description. Note: If requiring contraction for consistency maintenance only, a column for adherence to either $B \subseteq (B +_{\cup!} \neg p) +_{\cup!} p$ (ordered by recency) or $K_\sigma \subseteq K_{(\sigma + \neg p) + p}$ would match (OR-i).

example: $B = \{p \wedge q\}$, then $B \sim p = \emptyset$ and $(B \sim p) + p) = \{p\}$. Therefore, $K \nsubseteq Cn((B \sim p) + p)$, and (TR) does not hold.

For (LR), a similar example shows that (LR) does not hold. If $\sigma = p \wedge q$, then $K \sim_\sigma p = \emptyset$ and $(K \sim_\sigma p) + p = Cn(\{p\})$. So, $p \wedge q \notin (K \sim_\sigma p) + p$. (LR) does not hold.

For (OR), the key questions are (1) what is the optimal base: $(B^\cup + p)!$, and (2) is B a subset of that optimal base? Recall that reconsideration is on $B^\cup$ and its results are independent of any contractions performed earlier (see Observation 3.2.15). Since $p \in Cn(B)$, then $B + p \nvdash \perp$. Therefore, $(B^\cup! + p) \nvdash \perp$. If $p \in B$, it will be relocated to a spot that is not any weaker in the ordering (OR-i will have it as the most preferred belief; OR-ii does not change the ordering). Therefore, $(B^{\cup}+p)! = B+p = B$ (from Theorem 3.2.20). If $p \notin B$, then it is still the case that $(B^{\cup}+p)! = B+p$ (from Theorem 3.2.17). Either way, $B \subseteq (B^{\cup}+p)!$, and (OR) holds.

Case 3 In case 3, $p \notin Cn(B)$ and $B+p \not\vdash \perp$. Because of these assumptions, we know that $p \notin B^{\cup}$—otherwise, if $p \in B^{\cup}$ and $B+p \not\vdash \perp$, then $B+p \subseteq B^{\cup}$ and $(B+p) \succ B$ and $B \neq B^{\cup}!$ (which is not possible, because the premise is that B is equivalent to $B^{\cup}!$). So, $p \notin B^{\cup}$, and $p \notin \sigma$. Column (OR-ii) has NA (for "Not Applicable") as its entry, because (OR-ii) assumes that $p \in B^{\cup}$.

For the other columns, since $p \notin Cn(B)$, contraction results in no change to the base (or set) involved: $B \sim p = B$, $K \sim_{\sigma} p = K = Cn(B)$, and $\mathbf{B} \sim p = \mathbf{B}$. Clearly, (TR) holds and (LR) holds. (OR-i) also holds, since we know that $\mathbf{B}+p \not\vdash \perp$ and $p \notin Cn(B)$ (Theorem 3.2.17).

Case 4 In case 4, $p \notin Cn(B)$ and $B+p \vdash \perp$. Because $p \notin Cn(B)$, $B \sim p = B$ and $K \sim_{\sigma} p = K = Cn(B)$. Since $B+p \vdash \perp$, both (TR) and (LR) produce inconsistent spaces; therefore, they both hold.

For (OR), $\mathbf{B} \sim p = \mathbf{B}$. For (OR-i), the optimized-addition puts p at the most preferred end of the new sequence (most recent), so $p \in B_1$ forcing weaker elements of B to be retracted for consistency maintenance during reconsideration (recall $B+p \vdash \perp$). Therefore (OR-i) does not hold.[10] For (OR-ii), optimized-addition returns p to the same place in the sequence that it held in $B^{\cup}$ (recall $B_1^{\cup} = B^{\cup}$ and $\succeq = \succeq_1$). In this case, p will be retracted during reconsideration as it was before (to make the optimal base B). Therefore, $\mathbf{B}_1 = \mathbf{B}$ and (OR-ii) holds.

3.3.3 Discussion

When comparing the traditional base recovery adherence (in column TR) to optimized recovery adherence (shown in the OR columns), the latter produces improved adherence and/or improved results, because:

[10] Producing an optimal base is preferred to adhering to a recovery-like formulation by having an inconsistent base.

1. in all cases where retraction of p actually alters the base (both OR columns for cases 1 and 2), B is recovered following optimized addition;

2. using a recency-based ordering (OR-i), B is recovered in all cases where $\neg p \notin Cn(B)$ (cases 1-3);

3. if expansion by p traditionally makes the final base inconsistent (TR,4), the final base for (OR) *is* consistent *and* optimal, even if (as in the case of (OR-i,4) the base is not recovered. Recall that we favor optimality over adherence to recovery.

4. in the case where the contraction is most closely undone—where p was in $B^{\cup}$, and the ordering is not altered by expansion by p (OR-ii)—the original base is *always* recovered.

Reconsideration eliminates the results of any preceding contraction, because $B^{\cup}$ is unaffected by contraction: $(\boldsymbol{B} \sim p)!_{\cup} = \boldsymbol{B}!_{\cup}$ (see Observation 3.2.15). Likewise, optimized-addition also eliminates the results of any preceding contraction: $\forall q : (\boldsymbol{B} \sim q) +_{!\cup} \boldsymbol{p} = \boldsymbol{B} +_{!\cup} \boldsymbol{p}$. In a sense it is strange to say that reconsideration undoes the operation of contraction when it really just ignores any contractions that were done. However, contraction is typically done as a part of the belief change operations of revision or consolidation. Though Recovery is an interesting postulate, it is more interesting to see how it applies to the more practical instance of contraction for consistency maintenance.

When considering contraction *for consistency-maintenance only* (i.e., contraction as a result of consistency maintenance necessitated by the addition of a belief, or set of beliefs, to the base) and assuming ordering by recency, the recovery-like formulation $\mathbf{B} \subseteq (\mathbf{B} +_{!\cup} \neg p) +_{!\cup} \mathbf{p}$ would have column entries identical to those in the column under (OR-i). Likewise, the entries in a column for $K_{\sigma} \subseteq K_{(\sigma+\neg p)+p}$ would also be identical to the entries for column (OR-i). These (OR-i) results for $\mathbf{B} \subseteq (\mathbf{B} +_{!\cup} \neg p) +_{!\cup} \mathbf{p}$ and $K_{\sigma} \subseteq K_{(\sigma+\neg p)+p}$ show adherence to (R3) in [Chopra, Ghose, & Meyer2002]. (R3) states that if $\neg p \notin K(ES)$, then $K(ES) \subseteq K((ES * \neg p) * p)$, where ES is an epistemic state (a knowledge state), $ES * p$ is the operation of revising that state by the belief p, and $K(ES)$ is the belief space associated with that ES.

A key element of the improved Recovery aspect that reconsideration provides for bases is that it does not involve the addition of *extra* beliefs to the belief base. A belief base can "adhere" to Recovery if the contraction operation to remove p also inserts $p \rightarrow q$ into the base, for every belief q that is removed during that retraction of p. However, adding extra beliefs just to adhere to Recovery deviates from the assumption of a *foundations* approach, where the base beliefs represent the base input information from which the system or agent should reason. Not only would this technique insert unfounded *base beliefs*, but the recovery of previously removed beliefs would only show up in the belief *space*; whereas reconsideration actually returns the removed beliefs to the belief *base* (Strict Base Recovery).

There are contraction operations, other than liberation, that propose adding beliefs into the belief base as a part of the contraction process. In [Nayak1994], contraction of B by p would insert $p \rightarrow q$ for all beliefs q in B. Similarly, Nebel suggests adding $p \rightarrow \&B$ when contracting B by p, where $\&B$ is the conjunction of all beliefs in B [Nebel1989]. Both these techniques would result in Recovery adherence, but they also insert new beliefs into the base merely to satisfy this adherence. A different approach, called Disjunctive Maxi-Adjustment (DMA), is proposed in [Benferhat *et al.*2004], where beliefs that are removed from a base during consistency maintenance are replaced by a weakened version (selected various disjunctions of the beliefs), which maintains some of the information without raising an inconsistency. An example of this would be revising the set of equally believed beliefs $\{a,b,c\}$ with the more credible belief $\neg(a \wedge b)$: in addition to removing both a and b (skeptical reasoning), DMA inserts the belief $a \vee b$ into the base. This is different from the other methods, because the beliefs being inserted are a weaker version of pre-existing base beliefs and still support the foundations approach (much like replacing some bricks in a foundation with some temporary shoring that will not be in the way if the bricks get returned). Reconsideration could be used in conjunction with DMA as long as $B^{\cup}$ and $\succeq$ are updated with the newly inserted beliefs.

Returning to our discussion of the Recovery aspect of reconsideration and optimized addition: Reconsideration adds an aspect of Strict Base Recovery, and it does so equally well with paraconsistent logics (those where a contradiction does not automatically imply anything). In fact, relevance logics [Anderson & Belnap1975, Anderson, Belnap, & Dunn1992] are not only paraconsistent, but they also do not support the axiom that anything implies a true proposition. For reasoning and belief revision using non-classical logics such as these, even theory contraction would not adhere to the Recovery postulate, because deductive closure would fail to produce the beliefs needed by that recovery process. Given the set $B = \{p, q\}$, depending on the relationship of the beliefs (their relevance to each other), the belief $p \vee q$ might be in $Cn(B)$, but the belief $p \leftrightarrow q$ might not be in the closure, and $(K \sim (p \vee q)) + (p \vee q)$ might not be able to recover p or q. OR, however, would perform as it did in the Table 3.2, because it does not depend on closure to achieve its Recovery-like aspect.

If the linear ordering is *not* recency-dependent and $\succeq_1 \neq \succeq$, then there are cases where Optimized-recovery does *not* hold even though the resulting base will still be optimal. For Case 1, if p is re-inserted into the ordering at a weaker spot, then it might be retracted during reconsideration in the case where it is re-asserted in a position that is weaker than the conflicting elements of some pre-existing nogood *and* the incision function favors retracting p. This could also happen in case 2, unless the elements of some p-kernel are *all* high enough in the order to force the retraction of the beliefs conflicting with p. In Case 3 all Recovery formulations *always* hold. In Case 4, if p is inserted into the final ordering at a strong enough position (specifically, high enough that no kernel for $\neg p$ has all its elements higher), then p would survive the reconsideration step of optimized-addition—in which case, (OR) would *not* hold, even though the resulting base would still be optimal.

3.4 Belief Change Issues for Non-linear Orders

3.4.1 Introduction

When the ordering of the beliefs in $B^{\cup}$ is not linear, it becomes less clear what determines an optimal base. Even with a linear ordering, there is a point when deciding to remove many weaker beliefs rather than one strong belief begins to look like a questionable strategy. With a non-linear ordering, however, there are different issues to deal with.

Since I favor kernel operations, I will focus on the operation of kernel consolidation: specifically, which belief(s) should be selected for removal from active nogoods to eliminate all inconsistency. The issues discussed can easily be extrapolated to the operation of contraction: e.g., which belief(s) to remove from each p-kernel to retract the belief p from a belief space. All other operations on a knowledge state are determined by these two operations, except for expansion which is essentially unaffected by a change from a linear ordering to a pre-order.

3.4.2 A Single Nogood With a Non-Singleton Set of Minimal Beliefs

The main problem comes from a nogood having a minimal set of beliefs that is not a singleton set. In a linear ordering, every nogood has one unique minimal element, the belief that is less credible than any other belief in that nogood: given the nogood N, $(\exists p \in N) : \forall q \in N \setminus \{p\}, q \succ p$. Having a least element for every nogood makes culprit selection foolproof and simplifies the determination of how best to remove culprits from multiple nogoods (assuming, again, that optimization favors one strong belief over many weaker beliefs).

In a total pre-order, antisymmetry is not guaranteed; therefore, two or more beliefs can have the same level of credibility (called a ranking by [Williams & Sims2000]). In this case, is possible for a nogood to have more than one belief at the lowest ranking.

Rank	Beliefs in a Single Nogood	Beliefs Including 3 Nogoods
1	$p \quad p \rightarrow q$	$p \quad p \rightarrow q$
2	$q \rightarrow r$	$q \rightarrow r \quad s \vee w$
3	$r \rightarrow s \quad s \rightarrow t \quad \neg t$	$r \rightarrow s \quad s \rightarrow t \quad \neg t \quad \neg w \quad r$

Table 3.3: An example of nogoods in pre-orders. The left column of beliefs contains a single nogood. The right column of beliefs contains three nogoods: $NG1 = \{p,\ p \rightarrow q,\ q \rightarrow r,\ r \rightarrow s,\ s \rightarrow t,\ \neg t\}$, $NG2 = \{r \rightarrow s,\ s \rightarrow t,\ \neg t,\ r\}$, and $NG3 = \{s \vee w,\ s \rightarrow t,\ \neg t,\ \neg w\}$.

If a specific nogood has a non-singleton set of weakest elements (the culprit set), the choices for eliminating the inconsistency caused by *that* nogood include:

1. removing one of the elements in the culprit set (credulous reasoning);

2. removing all the elements from the culprit set (skeptical reasoning);

3. (least reasonable) removing more than one, but not all elements from the culprit set.

Note that, if there is only one nogood to consider,choice number 3 seems strange with only the ranking to assist in culprit selection, because there is no accepted reason for choosing a number between one and all culprits *non-inclusively*.

Example: Let N be a nogood represented as a set of beliefs paired with their respective ranks such that the lower the rank, the more credible the belief: $N = \{\langle p,1\rangle, \langle p \rightarrow q,1\rangle, \langle q \rightarrow r,2\rangle, \langle r \rightarrow s,3\rangle, \langle s \rightarrow t,3\rangle, \langle \neg t,3\rangle\}$

The first column of beliefs in Table 3.4.2 shows these ranking more clearly. Removing *all* beliefs at the third rank would be skeptical belief change. Removing *only one* of the beliefs at the third rank would be an example of credulous reasoning. Since there are three beliefs at the third rank, there are three possible outcomes of using credulous reasoning. Maintaining all three is a way of looking at all possible worlds.

3.4.3 Multiple Nogoods Having Non-Singleton Sets of Minimal Beliefs

A more complicated issue occurs when *multiple* nogoods have non-singleton culprit sets—especially when those sets intersect. To see the effect, expand on the example in the previous section by adding three more beliefs: $\langle s \vee w, 2\rangle, \langle \neg w, 3\rangle, \langle r, 3\rangle$. We now have three nogoods. They are the original nogood, $NG1 = \{p,\ p \rightarrow q,\ q \rightarrow r,\ r \rightarrow s,\ s \rightarrow t,\ \neg t\}$, and two additional nogoods: $NG2 = \{r \rightarrow s,\ s \rightarrow t,\ \neg t,\ r\}$ and $NG3 = \{s \vee w,\ s \rightarrow t,\ \neg t,\ \neg w\}$. These nogoods are represented in the rightmost column of Table 3.4.2. The minimal set of beliefs for each nogood lies at the third level, and these minimal sets intersect to form the set $\{s \rightarrow t,\ \neg t\}$.

Should we be skeptical and remove the union of the three minimal sets? Should we be credulous and remove one element from each nogood as long as it is at the third rank?

Perhaps we should focus on the *intersection* of the three nogoods. In this case, credulous reasoning might remove *either* $s \rightarrow t$ or $\neg t$, whereas skeptical revision would remove *both* beliefs. This is where one might reason that presence in multiple nogoods reduces credibility. This reasoning can backfire, though.

> Imagine the many failed attempts to make an airplane during the birth of aviation. Each failed flight was preceded by the formation of a set of base beliefs from which the inventor inferred that his plane would fly. After his contraption came crashing to the earth, the inventor had an inconsistent set of beliefs (a nogood). Thus, every failed flight produced a nogood. It is a fair bet that the belief "Gravity exists" was present in *all* the nogoods, but that certainly does not mean it should be removed from the belief base in an attempt to restore consistency.

Hence, a belief's presence in multiple nogoods does not *guarantee* that the belief is the best choice for removal to restore consistency. It should be noted, however, that the belief "Gravity exists" would most likely have a higher rank than other beliefs common to most or all the nogoods in the airplane example—

i.e., beliefs about how gravity, speed and construction would affect the flight characteristics of the plane—so "Gravity exists" would not be likely to be selected as a culprit.

3.4.4 Dealing with non-linear issues

This discussion of belief change issues was presented to emphasize the complexities of culprit selection when the beliefs are *not* ordered linearly and/or when nogoods intersect. Developing a *perfect* culprit selection function is beyond the scope of this dissertation—perhaps, beyond the scope of a lifetime of research.

A major goal of this research, however, is to provide an operation (the operation of reconsideration) that can improve the way an *existing* reasoning system maintains its belief base. This is dependent on the culprit selection strategy that is used *by that system* to restore consistency to an inconsistent base. The next section discusses six different strategies (that is, six of the *many* strategies) for restoring consistency in a belief base where the beliefs are ordered in a total pre-order. I then apply reconsiderastion to a pre-existing example that contrasts these strategies. The results show (at least for this one example) how reconsideration can, in most cases, improve a pre-order of base beliefs—and, in some cases, improve the base to an optimal state.

3.5 Belief Change in a Total Pre-order: Six Strategies Used by SATEN

3.5.1 Introduction

As mentioned earlier, SATEN [Williams & Sims2000] is an "extraction and belief revision engine" that is available over the internet. When performing an extraction or a revision, the user must choose one of six available adjustment strategies that select which beliefs should be removed (if necessary). Essentially, the adjustment strategies can be considered selection functions [Alchourrón, Gärdenfors, & Makinson1985] or in some cases incision functions [Hansson1994, Hansson1999], and an extraction (the extraction of a consistent belief base from the current base) is an operation of kernel consolidation [Hansson1994].

In the SATEN system, beliefs are accompanied by a degree of credibility between 0 and 1, where a higher degree means a belief is more preferred (more important). These degrees provide a ranking (1,2,3,...) where the highest degree in the base equates to a ranking of 1. Beliefs at the highest degree are also considered to be at the first or highest rank. Note that “rank is inversely proportional to importance” [Williams & Sims2000]. A *cut* of the base at a specific rank is the set of beliefs at that rank and above: e.g., a cut at rank 3 would include all beliefs at ranks 1, 2, and 3.

SATEN does no “track-keeping” or storage of derived beliefs. Its belief revision strategies are based on algorithms that (1) look at the degrees of credibility of the beliefs in the base and (2) calculate inconsistencies in the base at the time that revision is needed. In some cases, the information needed to perform kernel contraction or kernel consolidation is also calculated at the time that revision is done, but some of the strategies used do not perform kernel belief change operations.

The six adjustment strategies used by SATEN are explained briefly below with a mix of quotes from [Williams & Sims2000] followed by further clarification. They can produce pairwise distinct results when looking at an example of extraction (consolidation) on a base of ten beliefs at just four different degrees of credibility.[11] The example is actually shown as a base of nine of the ten beliefs revised by the tenth belief, $\neg a$. This is equivalent to adding $\neg a$ and performing contraction.

Table 3.4 shows the original base of nine beliefs (in the first row of base beliefs) and the results of revision by $\neg a$ under each adjustment strategy (in the second row of base beliefs). Table 3.5 also shows these same rows as its top two rows of belief bases.

In the explanations of the adjustment strategies that follow, I have added the reasons for each different revision result. To simplify later discussion, I am using semi-revision (implemented as addition followed by consolidation: $(B + \neg a)!$). Because $\neg a$ is entering the base at the highest level and is not inconsistent with

[11] This example was found (and is available for testing) on SATEN’s website: <http://magic.it.uts.edu.au/systems/saten.html>.

Belief Base	Rank	Degree	Standard	Maxi-adjustment	Hybrid	Global	Linear	Quick
B	1	.95	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$
	2	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
	3	.40	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f
	4	.20	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$
$(B + \neg a)!$	1	.95	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$
	2	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
	3	.40	f	e, f	$\neg b \vee \neg d$, e, f	e, f		d, e, f
	4	.20		$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$		$\neg g \vee \neg b$, $\neg d \vee g$	

Table 3.4: This table shows the basic SATEN example that compares its six adjustment strategies when revising a base B by the belief $\neg a$. The differences are explained in the text. Revision by $\neg a$ is performed by the addition of $\neg a$ followed by consolidation. This table is included in Table 3.5 in larger print.

the other belief at that level, this is equivalent to external revision $((B + \neg a) \sim a)$, which produces the same result in this system as internal revision $((B \sim a) + \neg a)$.

My intent is not to insist that the reader completely understand each strategy and its implementation, but rather that the reader be comfortable with the concept that (1) the strategies produce different results and (2) reconsideration can improve (and possibly optimize) a base after a series of belief change operations—even if the base beliefs are *not* in a linear ordering. I address point (1) in this section and point (2) in Section 3.6.

3.5.2 Standard Adjustment

From the standard adjustment description in [Williams & Sims2000]:

> Standard Adjustment [Williams1994a, Williams1995, Williams1994b] is based on the standard epistemic entrenchment construction [Gärdenfors & Makinson1988, Gärdenfors1988] for Belief Revision.

Essentially, standard adjustment moves down each rank testing for consistency with all higher ranks. When a rank is found that is inconsistent with the higher ranks, the beliefs at that rank and all weaker beliefs are removed from the base. Then, any removed beliefs that are *derivable* from the beliefs that were retained are returned to the base.

In the example shown in Table 3.4, inconsistency is found at the third rank. Of all the beliefs at Ranks 3 and 4, only f is derivable from the beliefs at Ranks 1 and 2. So all beliefs ranked at level 3 and 4 are removed—except for f.

A standard adjustment of a linear ordering is unlikely to produce an optimal result, because every belief at the first inconsistent ranking and below is removed, unless it is derivable from the beliefs ranked above the first inconsistency. Even if these weaker, unsupported beliefs are not in any nogoods (and, thus, unrelated to any inconsistenty), they are removed.

Consolidation using standard adjustment is *not* an example of kernel consolidation, because beliefs other than those in nogoods are removed.

3.5.3 Maxi-Adjustment

From the maxi-adjustment description in [Williams & Sims2000]:

> Maxi-adjustment is based on maximal inertia [Williams1997]. Maxi-adjustment proceeds from the top of the ranking and moves down it rank by rank. At each rank it deletes all the beliefs that are inconsistent with other beliefs at that rank and above. [It does this by removing] the union of all subsets S of the beliefs on that rank having the properties (1) S is inconsistent with ... all the information ranked higher in the base than S; and (2) no proper subset of S satisfies (1). This strategy keeps many of the beliefs in the theory base, but is also quite slow as it has to examine all subsets of each rank. Performance is improved if the agent is more discerning. [The] algorithm is anytime [Williams1997] so the longer the agent has to revise the better the result.

Essentially, when maxi-adjustment finds a rank j that is inconsistent with the beliefs in the ranks above it (1 through $j-1$), it calculates the subset at rank j of every nogood for the set of beliefs spanning ranks 1 through j. As mentioned in [Williams & Sims2000], this requires computing and testing all subset combinations at rank j which is computationally expensive (see [Williams & Sims2000] for complexity analysis).

The reference in the quote above to a discerning agent improving performance means the following: the more ranks the agent can spread the base beliefs across, the smaller the number of beliefs at any one rank resulting in fewer subset combinations to be tested in the inconsistent ranks. The *most* discerning agent presents a linear ordering of beliefs, which requires no subset testing at all and would produce the optimal base as defined in Section 3.2.2 (the base with the highest bitmap value).

The anytime aspect of the algorithm refers to the fact that the calculation of the new base begins at the highest rank and works its way down the rankings, iteratively building up a new base from the strongest rank (rank 1) to the weakest rank. If the algorithm is stopped at any point before completion, the current rank being processed, call it j, is dismissed, and the new base consists of what is *known* to be correct: the ranking of beliefs from rank 1 to rank $j-1$. Unfortunately, beliefs ranked below rank $j-1$ are lost. This is an anytime algorithm, because a partial solution is always available (the base made up of ranks 1 through $j-1$) and this partial solution improves as the algorithm continues to run.

The result of maxi-adjustment equates to skeptical reasoning with no attention to nogood intersections. It is as if the nogoods for the beliefs at that rank and stronger were calculated, and the minimal elements of each nogood were then removed.

In the example shown in Table 3.4, inconsistency is found at the third rank. The inconsistent subsets at that rank are $\{a \vee d,\ \neg b \vee \neg d\}$ and $\{\neg b \vee \neg d,\ d\}$, so $\{a \vee d,\ \neg b \vee \neg d,\ d\}$ is removed from the base. This leaves e and f at rank 3, and rank 4 does not produce an inconsistency.

Consolidation using maxi-adjustment *is* an example of kernel consolidation, because only beliefs in nogoods are removed from the base.

3.5.4 Hybrid Adjustment

From the hybrid adjustment description in [Williams & Sims2000]:

> Hybrid Adjustment is a combination of standard adjustment and maxi-adjustment. [Standard adjustment] is computationally "easier" than maxi-adjustment, however the main shortcoming of [standard adjustment] is that it maintains beliefs only if there is an explicit reason to keep them, whilst maxi-adjustment removes beliefs only if there is an explicit reason to remove them. Hybrid adjustment performs a standard adjustment which computes the core beliefs to be retained, and then it performs a maxi-adjustment which is used to regather as many beliefs as possible that were discarded during the adjustment phase.
>
> The hybrid adjustment strategy represents a combination of the standard adjustment and maxi-adjustment strategies. When revising with a, hybrid adjustment first finds every belief b in the base such that $\neg a \vee b$ is in the base, and is at the same rank as $\neg a$ (this is the adjustment step—it is removing every belief that "explicitly fails to be a consequence of the cut above $\neg a$), and then performs a maxi-adjustment on what is left. So in other words maxi-adjustment is used to recoup as much of the ranking as the time set aside for the revision permits. This strategy is comparable to maxi-adjustment with regard to the number of beliefs retained, and though somewhat faster on average, has the same worst-case time expenditure as the maxi-adjustment procedure.

Although hybrid adjustment can sometimes return a few more beliefs to the base than maxi-adjustment, they produce the most similar results. In the example shown in Table 3.4, inconsistency is yet again found at the third rank, but hybrid adjustment differs from maxi-adjustment by first removing the two beliefs $a \vee d$ and d. After their removal, rank 3 is no longer inconsistent, so $\neg b \vee \neg d$ can remain in the base along with e and f and the beliefs at rank 4.

As with maxi-adjustment, hybrid adjustment of a linear ordering would produce the optimal base.

Consolidation using hybrid adjustment *is* also an example of kernel consolidation, because the maxi-adjustment phase insures that only beliefs in nogoods are removed from the base.

3.5.5 Global Adjustment

From the Global adjustment description in [Williams & Sims2000]:

> Global Adjustment is similar to a maxi-adjustment except that it takes all beliefs in the ranking into consideration when computing the minimally inconsistent subsets, instead of proceeding rank by rank. It removes the least ranked beliefs "causing" the inconsistency.

Global adjustment is the only adjustment strategy that does *not* proceed down the ranking of beliefs rank by rank. Instead it computes the *complete* nogoods for the entire base (as if the beliefs were not ranked at all). This takes *considerably* longer than the other adjustment strategies, even much longer than the subset calculations that maxi-adjustment performs at a single rank. Once the nogoods are computed, the minimal elements of *each* nogood are removed from the base.

This is similar to consolidation by safe contraction[12] [Alchourrón & Makinson1985]. But, safe contraction has a second step much like standard adjustment does. After safe contraction removes all minimal culprits (like global adjustment), any removed beliefs that are derivable from the remaining base beliefs are returned to the base.

[12] Recall that revision is performed by adding $\neg a$ and then consolidating. The way to perform consolidation using safe contraction is to contract by falsum: $(B + \neg a)! = ((B + \neg a) \sim \perp)$, where $\sim$ is the operation of safe contraction.

In the example shown in Table 3.4, global adjustment computes the following nogoods (with minimal elements underlined):

NG1= $\{\neg a,\ a \vee b,\ \underline{\neg b \vee \neg d},\ \underline{a \vee d}\}$,

NG2= $\{\neg a,\ a \vee b,\ \underline{\neg b \vee \neg d},\ \underline{d}\}$,

NG3= $\{\neg a,\ a \vee b,\ d,\ \underline{\neg g \vee \neg b},\ \underline{\neg d \vee g}\}$,

NG4= $\{\neg a,\ a \vee b,\ a \vee d,\ \underline{\neg g \vee \neg b},\ \underline{\neg d \vee g}\}$.

The minimal (underlined) elements are then removed. Note that the ranking *is* used to determine the minimal elements for each nogood. This is skeptical reasoning with no regard for intersecting nogoods *or* for the removal of non-minimal elements in a nogood: e.g., the removal of $a \vee d$ and d does not prevent the removal of the minimal elements in NG3 and NG4 (if it did, it would produce the same results as maxi-adjustment).

Global adjustment of a linear ordering will produce an optimal base if none of the nogoods intersect; otherwise, an optimal result is unlikely.

Although global adjustment is slower and produces less desirable results than maxi-adjustment, it does compute the entire set of nogoods for this base, and (as I will show in the next chapter) that information might be beneficial to save.

Consolidation using global adjustment *is* an example of kernel consolidation, because it very specifically calculates the nogoods and only removes elements (the minimal ones) from those nogoods.

3.5.6 Linear Adjustment

From the linear adjustment description in [Williams & Sims2000]:

> Linear Adjustment is similar to maxi-adjustment except that it removes all the beliefs at ranks which contain inconsistencies with beliefs above them [Nebel1994]. In the case where there is only one belief at each rank, then linear and maxi-adjustment are identical.

> [This] strategy looks at the successive cuts of the theory-base and, at every rank at which the cut is inconsistent, removes the entire rank from the base. This is quite time-efficient, and keeps more beliefs in the base than standard adjustment, but, unlike hybrid adjustment and maxi-adjustment, is fairly ad-hoc about the beliefs to be kept in the theory base.

In the example shown in Table 3.4, linear adjustment removed all the beliefs at the third rank. Because of that removal, rank 4 is not inconsistent with the remaining base.

Consolidation using linear adjustment is *not* an example of kernel consolidation, because beliefs other than those in nogoods may be removed.

3.5.7 Quick Adjustment

From the quick adjustment description in [Williams & Sims2000]:

> Quick Adjustment is identical to maxi-adjustment except that rather than removing all beliefs it randomly chooses a minimal number of beliefs from each set of inconsistent beliefs in this way the inconsistency is removed.
>
> [This] strategy is an attempt to capture the judiciousness of maxi-adjustment without the associated time expenditure. Once a given rank of beliefs is discovered to cause a contradiction, quick adjustment begins adding beliefs from the left end of that rank until a contradiction is discovered. It then removes these beliefs, one at a time, keeping track of which ones eliminate the contradiction when they are removed. It is these "culprit" beliefs that are then removed from the theory base. This process is then iterated. This strategy has the advantage of improving performance by reducing the amount of processing at each rank, and also emulates the output of maxiadjustment much of the time. However, its behaviour depends upon the order of the inputs on a given level of the theory base, and hence is somewhat ad hoc. The system could

be easily extended to generate more than one ranking and used to perform credulous reasoning. Instead of randomly choosing an alternative, all alternative revised rankings could be created.

In the example shown in Table 3.4, quick adjustment removes $a \vee d$ and $\neg b \vee \neg d$ at rank 3. This eliminated the inconsistency at that rank; but, by not removing d, rank 4 is inconsistent and quick adjustment chooses to remove both beliefs at that rank.

Quick adjustment is not guaranteed to remove *only* one element from each active inconsistent subset when it reaches an inconsistent rank. It is also not guaranteed to remove the intersection (if it exists) of those sets. However, depending on the order that the beliefs are listed at the inconsistent rank, both of those results are a possibility. When a different order was used at rank 3, specifically placing $\neg b \vee \neg d$ to the right of both $a \vee d$ and d, then quick adjustment removed only $\neg b \vee \neg d$ from rank 3. This is an example of removing one belief from an inconsistent subset *and* an example of removing the intersection of two inconsistent subsets. During this testing, I was unable to find any list order for the rank 3 beliefs that resulted in quick adjustment removing both $a \vee d$ and d but retaining $\neg b \vee \neg d$, which would have allowed the beliefs at rank 4 to be retained. Additionally, no list order at rank 4 could produce a quick adjustment that removed only one of the rank 4 beliefs.

Quick adjustment on a linear ordering will produce the optimal base. Consolidation using quick adjustment *is* also an example of kernel consolidation, because only beliefs underlying the inconsistency are removed and those beliefs are elements of the nogoods.

3.5.8 Preparing to Test Reconsideration

In the next section, I test reconsideration using this same example to see whether it can improve the state of a belief base that is *not* ordered linearly.

3.6 Reconsideration on a Total Pre-order Using SATEN Strategies

3.6.1 Introduction

To test how reconsideration works with the six adjustment strategies just discussed, I chose to use the same example that is used to compare them. I revise the already revised bases by the belief a at a higher degree of credibility than the previously added $\neg a$; then I see if reconsideration improves the belief base. The degree for a is .98, whereas the degree for $\neg a$ is .95. In effect, I am "undoing" the revision by $\neg a$. As before, revision is implemented by addition followed by consolidation, where the consolidation is performed by each of the six adjustment strategies.

If the adjustment strategies could adhere to the Recovery postulate, the revision by a would produce a base that is a superset of the original base B. This is because $\neg a$ was not an element in the closure of B originally: if $\neg a \notin Cn(B)$, then $((B+\neg a)!+a)! = (((B+\neg a) \sim a)+a) \sim \neg a = (((B \sim a)+\neg a) \sim \neg a)+a = (B \sim a)+a$. Therefore if Recovery holds, then $B \subseteq Cn((B \sim a)+a)$, and, therefore, $B \subseteq Cn(((B+\neg a)!+a)!)$. However, as mentioned before, foundations-oriented base belief change operations cannot adhere to Recovery, and the six SATEN strategies are no exception.

3.6.2 Performing a Second Revision: ($((B+\neg a)!+a)!$)

The results of this second revision ($((B+\neg a)!+a)!$) are shown in the third row of belief bases in Table 3.5.

Recovery does not hold for any of these strategies; because, for each resulting base, there is at least one element from the original base, B, that is not derivable from that base. Both $\neg b \vee \neg d$ and d were beliefs in B; but the base produced by quick adjustment does not entail $\neg b \vee \neg d$, and the bases produced by the other strategies do not entail d. Therefore, none of the six strategies adhere to recovery.

Four of the strategies merely remove the belief $\neg a$ during revision by a; those adjustment strategies are maxi-adjustment, hybrid, global, and quick. This is consistent with the kernel consolidation nature of all four strategies: since the only nogood in the base being consolidated is $\{a,\ \neg a\}$, and all strategies remove the minimal elements of a nogood, all remove only $\neg a$. Standard and linear adjustments differ in their results for this revision, because they are *not* kernel operations.

Standard removes $\neg a$; it retains $a \vee b$ and $a \vee f$, because they follow from the consistent cut $\{a\}$; it removes f, because f is *not* derivable from the consistent cut $\{a\}$ (it was retained previously, because it follows from $\neg a$ and $a \vee f$, which were in the consistent cut from the revision by $\neg a$).

Linear adjustment removes all the beliefs at the inconsistent rank 2, so $a \vee b$ is removed along with the contradictory $\neg a$.

3.6.3 Reconsideration: $(\ ((B + \neg a) + a)!\ )$

The goal of this test was to see if reconsideration could improve the knowledge state that was produced by a series of revisions, even thought the beliefs are ordered in a total pre-order as opposed to a linear ordering, and even though some of the adjustment strategies are not kernel operations.

I implemented reconsideration by consolidating the set of all base beliefs: $((B + \neg a) + a)!$. The consolidation operation was performed by all six adjustment strategies, and the results are shown in the last row of Table 3.5. To picture the entire set of base beliefs, $((B + \neg a) + a)$, look at B in the top row of bases in the table and imagine the following: $\neg a$ alongside $a \vee b$ at .95 and a at a rank above those two beliefs at .98.

Three Strategies Produced the Optimal Base

For three of the strategies, reconsideration produced the optimal base[13] of $B + a$, which has the Recovery aspect of containing the original base in its closure: $B \subseteq Cn(\ ((B + \neg a) + a)!\ )$. In fact, the base, itself,

[13]Pre-orders may not have a unique optimal base for the operation of $((B + \neg a) + a)!$, where $a \succ \neg a$. For this example, since $\neg a \notin Cn(B)$, the optimal base is $B + a$.

Table showing a base, B, revised by $\neg a$ (.95), then revised by a (.98), and then after Reconsideration is performed.
Columns show different adjustment strategies producing varied results for revision and reconsideration.

Belief Base	Degree	Standard	Maxi-adjustment	Hybrid	Global	Linear	Quick
B	.95	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$
	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
	.40	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f
	.20	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$
$(B + \neg a)!$	.95	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$	$\neg a$, $a \vee b$
	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
	.40	f	e, f	$\neg b \vee \neg d$, e, f	e, f		d, e, f
	.20		$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$		$\neg g \vee \neg b$, $\neg d \vee g$	
$((B + \neg a)! + a)!$	.98	a	a	a	a	a	a
	.95	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$		$a \vee b$
	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
	.40		e, f	$\neg b \vee \neg d$, e, f	e, f		d, e, f
	.20		$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$		$\neg g \vee \neg b$, $\neg d \vee g$	
$((B + \neg a) + a)!$	.98	a (improved)	a (optimal)	a (optimal)	a (unchanged)	a (improved)	a (optimal)
	.95	$a \vee b$	$a \vee b$	$a \vee b$	$a \vee b$		$a \vee b$
	.90	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$	$a \vee f$
Reconsideration	.40	$a \vee d$	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f	$a \vee d$, $\neg b \vee \neg d$, d, e, f
	.20		$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$		$\neg g \vee \neg b$, $\neg d \vee g$	$\neg g \vee \neg b$, $\neg d \vee g$

Table 3.5: This table compares revision and reconsideration on a total pre-order of beliefs using six different adjustment strategies. Revision by a belief is performed by the addition of a belief followed by consolidation. Reconsideration is consolidation on all base beliefs ($B^{\cup}$). The "Rank" column contains the beliefs' degrees of credibility. See text for further discussion.

explicitly contains the original base: $B \subseteq (\ ((B + \neg a) + a)! \ )$. This means that the revision by $\neg a$ was completely reversed. The three strategies are maxi-adjustment, hybrid adjustment, and quick adjustment. These all worked, because they found the inconsistency at rank 2 (due to $\neg a$ at .95 conflicting with a at .98) and removed $\neg a$ (since they are kernel oriented). After that, no further inconsistencies existed.

Reconsideration on a linear ordering using any one of these three strategies will produce an optimal base. Reconsideration on a non-linear ordering using any one of these strategies is *not guaranteed* to produce optimal results. It is nice to see that it can happen, though.

Reconsideration Using Global Adjustment Never Improves the Base

Reconsideration using global adjustment is the fourth kernel operation, but it does not produce an optimal base. This is because it always removes at least one minimal element from every nogood. Even though $\neg a$ is present in all the nogoods, and its removal is sufficient to remove all inconsistencies, $\neg a$ is not the *minimal* element in all the nogoods, so other beliefs are removed (the same beliefs and for the same reasons as those discussed in Section 3.5.5).

Since global adjustment always removes the minimal elements of all nogoods, reconsideration using global adjustment will *never* improve on the results of a series of revisions performed using global adjustment. There are even cases where reconsideration will remove more beliefs (moving away from an optimal result).

Example: Given two beliefs $\neg a$ and b at .90, revision by a at .95 followed by revision by $a \vee \neg b$ at .98 results in the belief base $\{a \vee \neg b, \ a, \ b\}$. Reconsideration on all the beliefs, removes both beliefs at .90, because they contradict $a \vee \neg b$ at .98; so the resulting base is $\{a \vee \neg b, \ a\}$, which is a proper subset of a consistent base and, therefore, sub-optimal.

Reconsideration is not necessary unless beliefs have been removed. In the above example, the first adjustment was optimal, and the second revision did not require further retractions, so reconsideration need not have been called.

Reconsideration Improves the Bases for the Standard and Linear Adjustment Examples

As seen in Table 3.5, reconsideration performed using standard adjustment improved the belief base in this example by returning $a \vee d$ (along with $a \vee b$ and $a \vee f$) to the base because it is derivable from the consistent cut $\{a\}$.

Likewise, reconsideration performed using linear adjustment improved the belief base because the beliefs with a degree of credibility of .40 did not get removed. The linear result is only suboptimal because the belief $a \vee b$ was removed along with $\neg a$ when all beliefs with a degree of .95 were removed.

Linear adjustment reconsideration on a linear ordering of beliefs will always produce an optimal base. Linear adjustment reconsideration on a non-linear ordering of beliefs is *not* guaranteed to produce an optimal base.

As mentioned before, the standard adjustment strategy will rarely yield an optimal base, no matter how the beliefs are ordered.

The sub-optimality of Standard and Linear Adjustments occurs largely due to the fact that their algorithms are not incision functions and do not perform kernel contraction—elements that do *not* underly an inconsistency may get removed during consolidation solely because they have the same (in both Standard and Linear Adjustment) or weaker (Standard Adjustment, only) degrees of credibility as some contradictory belief that *must* be removed. Specifically, these strategies do not adhere to the postulate for base contraction called Relevance (discussed in Section 2.2.4) and the postulate for kernel contraction called Core-Retainment (see Theorem 2.2.1), because they may remove beliefs that are not *relevant* to (i.e., they do not underly) any inconsistency.

3.7 Conclusions

Reconsideration provides a way for belief change operations on a finite set of base beliefs to improve the knowledge state by optimizing the belief base with the help of hindsight. The result is the return of previously retracted beliefs reminiscent of the Recovery postulate. This occurs without altering the set of base beliefs that have been considered by the reasoning agent.

For reconsideration to optimize a belief base, the incision function used to determine what beliefs should be removed during kernel consolidation must be able to maximize the value of the belief base bitmap for a consistent base. This is easy to achieve if the beliefs are linearly ordered, but all that is required is knowledge of the least element in each nogood of base beliefs. Defining an optimal base as the maximal bitmap for a consistent base (when a linear ordering is possible) is a technique supported in belief revision research [Williams & Sims2000, Booth *et al.*2005].

Section 3.2.4 shows that a system that can implement σ-liberation can also implement reconsideration and vice versa. This is assuming (1) a linear ordering of beliefs and (2) both systems can implement the same ordering. I have not proved whether a non-linear ordering with least elements of the nogoods is sufficient for performing σ-liberation, and I leave this for future work.

Kernel consolidation of a finite belief base has adherence results for Optimized-recovery (OR) that are preferred over the adherence results for the traditional Recovery postulate for base contraction. Thus, reconsideration imparts a Recovery aspect to belief bases that was previously lacking. Although σ-liberation retraction was never intended to adhere to Recovery, if (1) contractions are *for consistency maintenance only* and (2) the base for each new ordering is determined, then it adheres identically as well as kernel contraction adheres to OR).

For non-linear orderings of beliefs, there is less guarantee that reconsideration can produce an optimal base. In fact, what *defines* an optimal base is less clear for non-linear orderings. What I do show is that, for an arbitrary example comparing six different decision functions, a series of revisions on a total-pre-order results in a sub-optimal belief base which can be improved (in all but one case) and possibly optimized (as seen in three/half of the cases) by performing reconsideration. It is interesting to note that one optimization occurred using a skeptical approach (maxi-adjustment), and another occurred using a credulous approach (quick adjustment). These are not guarantees that reconsideration will always optimize a base that has a non-linear ordering, but they do support using reconsideration as a background process to *possibly* improve the reliability of a base.

A complete analysis of reconsideration for non-linear orderings is reserved for future work.

In the next chapter, I present an algorithm for implementing reconsideration in any truth maintenance system (TMS) where the set of nogoods of base beliefs is known.

Chapter 4

Dependency-Directed Reconsideration

This chapter contains material that was published in [Johnson & Shapiro2005b]. It has been altered and expanded for clarity and includes full algorithms and proofs (although some proofs are in the Appendix due to their length).

4.1 Introduction, Motivation and Assumptions

The preceding chapter presented the operation of reconsideration on the knowledge state triple $\langle B, B^{\cup}, \succeq \rangle$. The discussion included the purpose of reconsideration, benefits, similarities to other belief change operations, and examples of its use. Reconsideration can be implemented in several ways. Some examples are:

- examine *all* beliefs in $B^{\cup}$ in order (from most to least preferred) to determine which can be in the optimal base[1]
- assume all non-culprit beliefs are in the base and only examine the culprit beliefs (in descending order

[1]This is similar to the definition for determining the base for some σ in the belief liberation research (cf. Section 2.2.7) or the way that SATEN would extract a consistent base from a linear ordering using either maxi-adjustment, hybrid adjustment, linear adjustment or quick adjustment—but *not* standard or global adjustments (cf. Section 3.5).

of preference) to see which ones can *also* be in the base

- assuming B is consistent, examine just those beliefs in X (in descending order of preference) to see which can return. Recall that $X = B^{\cup} \setminus B$. This is the only naive algorithm that can be performed on the current belief base and incrementally improve it.

The examples just mentioned are progressively less naive.

In this chapter, I present an algorithm that assumes that, in addition to maintaining the three elements of the knowledge state triple described in Chapter 3 ($\langle B, B^{\cup}, \succeq \rangle$), a system can (1) identify and store inconsistent sets (nogoods [de Kleer1986, Forbus & de Kleer1993]) as well as (2) maintain connections between a base belief and any nogoods containing that belief.

This is a reasonable expectation for any track-keeping system, such as the justification-based truth maintenance system (JTMS) [Doyle1979] and the assumption-based truth maintenance system (ATMS) [de Kleer1986, Martins1983, Martins & Shapiro1988], which were discussed in Section 2.4.[2]. These truth maintenance systems already identify inconsistent sets when performing truth maintenance to eliminate contradictions [de Kleer1986, Forbus & de Kleer1993].

However, a nogood need not be *minimally* inconsistent, so I use the term NAND-set to refer to a set of base beliefs that an *implemented system* identifies as *minimally* inconsistent.[3] Note that, unless the system's reasoning is complete, there is no guarantee that a NAND-set is truly minimal, only that the *system* has not determined that any proper subset of that NAND set is inconsistent.

As a generalization of the binary NAND (standing for NOT AND), a consistent base *cannot* derive the conjunction (AND) of all the elements of a NAND-set. If all *minimally inconsistent* NAND-sets are known, the goal of eliminating inconsistencies can be achieved by insuring that no NAND-set is a subset of the

[2] Although minimally inconsistent nogoods are identified by SATEN during global adjustment (cf. Section 3.5), the process uses SAT testing and is computationally impractical for a large knowledge base.

[3] I thank my advisor, Stuart C. Shapiro, for suggesting NAND-set as the name such a set.

current belief base—i.e., if at least one element in a NAND-set N is removed from the belief base, the inconsistency that is attributed to N is eliminated.

Example 4.1.1 An example of how a TMS can identify a NAND-set after detecting an inconsistency can be seen by referring to Figure 2.1 on page 45. The derivation of both q and $\neg q$ indicates an inconsistency. A JTMS would trace down the paths of the derivation tree to determine the NAND-set formed by the base beliefs underlying the derivation of $\perp$. An ATMS would union the origin sets for q and $\neg q$ ($\{p,\ p \rightarrow q\}$ and $\{s,\ s \rightarrow t,\ t \rightarrow \neg q\}$, respectively) to determine the NAND-set. In either case, the NAND-set is $\{p,\ p \rightarrow q,\ s,\ s \rightarrow t,\ t \rightarrow \neg q\}$.

The best way to improve the computational expense of reconsideration over the naive algorithms listed above is to reduce the number of retracted base beliefs whose removal is to be reconsidered. By storing and using the NAND-sets detected in $B^{\cup}$ dependency-directed reconsideration can greatly reduce the number of beliefs examined for return to the base.

4.2 Dependency-Directed Reconsideration (DDR)

4.2.1 The DDR Process

If consolidation results in removing some base belief, p, from the base, this might allow a currently disbelieved *weaker* base belief, q, to return to the base; e.g., q can return if the reasons for disbelieving it depended on p being believed. Specifically, a belief can return if it either (1) does *not* raise an inconsistency or (2) if any inconsistencies raised can be resolved by retracting *only* weaker beliefs. The DDR algorithm processes retracted beliefs in two ways:

1. if a belief was retracted, DDR uses the NAND-sets to determine which weaker beliefs should be

examined for *possible* return to the base and inserts them onto a global *priority queue*[4]

2. if a retracted belief q is on the queue to be considered for possible return, DDR determines *whether* q can return, and, *if so*, executes the return of q to the base while retracting some (possibly empty) set of weaker conflicting beliefs in order to maintain consistency.

Because a retracted belief points (by way of its NAND-sets) to the removed beliefs to be examined for possible return to the base, DDR is "dependency-directed" (a phrase coined in [Stallman & Sussman1977]).

4.2.2 Defining a Knowledge State

Definition 4.2.1 A DDR knowledge state is a five-tuple: $\langle B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\rangle$, where :

- B is a set of base beliefs that are currently believed (the current base)
- $B^{\cup}$ is the set of *all* base beliefs, believed or not. Therefore, $B \subseteq B^{\cup}$.
- $I \subseteq \{N \mid N \subseteq B^{\cup} \text{ and } N \vdash \perp\}$, where $(\nexists N, N' \in I) : N' \subsetneq N$.
- $\succeq$ is a linear preference ordering on the beliefs in $B^{\cup}$
- Q is a priority queue consisting of a sequence, possibly empty, of tagged beliefs (written as $\langle p_1, \tau_1\rangle, \langle p_2, \tau_2\rangle, \ldots, \langle p_n, \tau_n\rangle$) s.t.
  - $p_i \in B^{\cup} \setminus B$
  - $i \neq j \ \Rightarrow \ p_i \neq p_j$
  - $p_i \succ p_{i+1}\ ,\ \ 1 \leq i < n$
  - $\tau \in \{justout, in?, both\}$

For convenience, let X be the set of currently disbelieved base beliefs and X will be called the *exbase*: $X =_{def} B^{\cup} \setminus B$. Note that if $\langle p_i, \tau_i\rangle \in Q$, then $p_i \in X$.

[4]Note that a priority queue is different from a standard queue. A standard queue is a first-in-first-out data structure—much like a line for tickets at the cinema. Think of a priority queue as a high school cafeteria line where one might step into line in a place befitting one's rank within the current makeup of the line—highest rank can enter the front of the line, middle ranked step in line in the middle, and so forth.

Unless otherwise stated, for this chapter, I assume the notational default that (for $i \in \{0,1,2,\ldots\}$) any DDR knowledge state referred to as KS_i is the five-tuple $\langle B_i, B_i^{\cup}, I_i, \succeq_i, Q_i \rangle$ and that $X_i = B_i^{\cup} \setminus B_i$.[5] For a knowledge state referred to as KS, the five-tuple elements are B, $B^{\cup}$, I, $\succeq$, and Q, respectively, and $X = B^{\cup} \setminus B$. These elements may also be referred to with a subscript of the knowledge state that they comprise: e.g., B can also be written as B_{KS}, and B_1 could also be called B_{KS_1}. This will remain unchanged for the duration of this chapter. I also use the shorthand expression $p \in Q$ to stand for $\exists\tau$ s.t. $\langle p,\tau \rangle \in Q$. Additionally, the deductive closure of a knowledge state is defined as the closure of its base: $Cn(KS) =_{def} Cn(B)$.

4.2.3 Defining a NAND-set

Elements in I are called NAND-sets.

Definition 4.2.2 A set N is a NAND-set IFF $N \in I$.

4.2.4 Defining Complete-I

The DDR knowledge state KS has a component called I, which is a set of NAND-sets where no element of I can be a proper subset of another element in I. One cannot assume that the NAND-sets in I are minimally inconsistent; nor are they guaranteed to represent all inconsistencies in $B^{\cup}$.

An ideal reasoner has the ability to reason both soundly and completely. Therefore, it knows all minimally inconsistent sets of beliefs. Given a set of base beliefs $B^{\cup}$, Complete-I($B^{\cup}$) is defined as the set of all minimally inconsistent sets in $B^{\cup}$:

Definition 4.2.3 Complete-I($B^{\cup}$) $=_{def} \{N \mid N \subseteq B^{\cup}, N \vdash \perp \wedge (\forall N' \subset N) : N' \nvdash \perp\}$. Because every KS has an associated $B^{\cup}$, Complete-I(KS) $=_{def}$ Complete-I($B^{\cup}$). Note that the elements of Complete-I(KS) are the

[5]For example: $KS_1 = \langle B_1, B_1^{\cup}, I_1, \succeq_1, Q_1 \rangle$.

minimally inconsistent NAND-sets in $B^{\cup}$.

If a system with an ideal reasoner implemented DDR through the use of a DDR knowledge state, it would be expected that $I = \text{Complete-I}(KS)$.

The information in I can be represented in a graph connecting base beliefs to the NAND-sets they are in. For an example, see Figure 4.2 on page 164. In this example, $I = \text{Complete-I}(KS)$.

4.2.5 Defining Culprits

The definitions and observations in this section all assume a DDR knowledge state KS containing the elements $B, B^{\cup}, I, \succeq$, and Q.

Because there is a linear ordering over all beliefs in $B^{\cup}$, there must be a minimal (least) belief in any subset of $B^{\cup}$:

Definition 4.2.4 Given a non-empty set $S \subseteq B^{\cup}$, where $B^{\cup}$ is ordered by $\succeq$,

Weakest$(S, \succeq) =_{def}$ the belief p such that $p \in S$ and $(\forall q \in S) : q \succeq p$.

The weakest belief in a set is the least preferred belief in that set:

Observation 4.2.5 *For some set* $S \subseteq B^{\cup}$, p = *Weakest*$(S, \succeq) \equiv (\forall q \in S) : q \neq p \Rightarrow q \succ p$.[6]

The weakest element of a NAND-set is its culprit:

Definition 4.2.6 If $S \in I$, then Culprit$(S, \succeq) =_{def}$ Weakest$(S, \succeq)$.

A NAND-set's culprit belief is less preferred than any other belief in that NAND-set:

Observation 4.2.7 *For some set* $S \in I$, p = *Culprit*$(S, \succeq) \equiv (\forall q \in S) : q \neq p \Rightarrow q \succ p$.

[6]The $\equiv$ can be read "if and only if" and is used to indicate that the two boolean expressions it separates have the same boolean value. It can also be written using the symbol $\Leftrightarrow$ or the text IFF, and I choose to use *all* these forms. The form used at any given time is determined by my perception of which form best suits the flow of the statement that I am expressing—it is purely subjective and, quite probably, inconsistent.

NAND-sets can be inserted into a priority queue that is ordered by their culprits (non-increasing order of credibility). Since more than one NAND-set can have the same culprit, this ordering is *not* linear. I refer to a priority queue of NAND-sets as a *NAND-set-Q*.

Definition 4.2.8 Given a set of NAND-sets $I' \subseteq I$, NAND-set-Q$(I', \succeq) =_{def} S_1, S_2, ..., S_n$, for all $S_i \in I'$, where Culprit$(S_j, \succeq) \succeq$ Culprit$(S_{j+1}, \succeq)$ for $1 \leq j < n$.

A knowledge state is consistent if its base is consistent:

Definition 4.2.9 Consistent$(KS) \equiv B_{KS} \nvdash \perp$.

The culprits of a knowledge state are the weakest (least) elements of its NAND-sets:

Definition 4.2.10 Culprits$(KS) = \{p \mid (\exists S \in I) : p =$ Weakest$(S, \succeq)\}$.

For any belief $p \in B^{\cup}$, the NAND-sets that contain p as their culprit are p's *IC-NAND-sets* (IC stands for *is-culprit*) and those NAND-sets that contain p but with a different belief as their culprit are p's *NC-NAND-sets* (NC stands for *not-culprit*):

Definition 4.2.11 Given a belief $p \in B^{\cup}$:

IC-NAND-sets$(p, KS) = \{S \mid (S \in I)$ and $p =$ Culprit$(S, \succeq)\}$.

NC-NAND-sets$(p, KS) = \{S \mid (S \in I), p \in S,$ and $p \neq$ Culprit$(S, \succeq)\}$.

4.2.6 Definitions Relevant to Knowledge State Consistency and Optimality

The definitions and observations in this section all assume a DDR knowledge state KS containing the elements $B, B^{\cup}, I, \succeq$, and Q, and the exbase $X = B^{\cup} \setminus B$.

An *active* set is a set that is a subset of the current belief base (i.e., all its base beliefs are currently believed):

Definition 4.2.12 Active$(A, KS) \equiv A \subseteq B$.

Observation 4.2.13 *If any NAND-set is active, then B is inconsistent:* $(\exists S \in I)\ Active(S, KS) \Rightarrow B \vdash \perp$.

Observation 4.2.14 *If $I =$ Complete-I(KS), the base of the knowledge state KS is inconsistent IFF at least one of its NAND-sets is active:* $I =$ *Complete-I(KS)*$\Rightarrow (B \vdash \perp \equiv (\exists S \in I) : Active(S, KS))$

A *precarious* NAND-set is a NAND-set that has all but one of its base beliefs in the current belief base (i.e., only one of its beliefs is *not* currently believed):

Definition 4.2.15 Given the NAND-set $S \in I$, Precarious$(S, KS) \equiv (|S \cap X| = 1)$.

A DDR knowledge state is Safe-per-I if none of its NAND-sets are Active.

Definition 4.2.16 Safe-per-I$(KS) \equiv (\forall S \in I) :\ \neg$Active$(S, KS)$.

A DDR knowledge state is Safe-per-I, if and only if its base is Safe-per-I: Safe-per-I$(B) \equiv$ Safe-per-I(KS).

Observation 4.2.17 *Given a DDR knowledge state KS, where $I =$ Complete-I(KS):*

Safe-per-I$(KS) \equiv$ *Consistent(KS).*

MustOut(p, KS) indicates that the belief p is the culprit (weakest element) of some NAND-set whose other beliefs are currently believed. Note that MustOut does *not* indicate whether p is currently believed or not, merely that it *must* not be believed if consistency is desired (given the other elements in the current base).[7]

Definition 4.2.18 MustOut$(p, KS) \equiv (\exists S \in I) :\ S \setminus \{p\} \subseteq B$ and $p =$ Culprit$(S, \succeq)$.

The following three definitions are about a disbelieved belief p (i.e., $p \in X$).

A belief p is designated as BadOut in a knowledge state *KS* if it is disbelieved and it is not the case that MustOut(p, KS). This means that for any precarious NAND-set $S \subseteq I$ containing p, p is *not* the culprit for S.

[7] It is possible that multiple NAND-sets are active in B and that there is a currently believed culprit, q, that is *MustOut* and that shares the only NAND-set that is designating p as MustOut. If this is the case, once q is removed, p would no longer be MustOut and can remain in the base without threatening consistency. DDR recognizes this—whether p is already disbelieved or whether q and p are removed together—and DDR would end up returning p to the base (barring any other obstacle to its return).

Definition 4.2.19 $\text{BadOut}(p, KS) \equiv p \in X \ \wedge \ \neg\text{MustOut}(p, KS)$.

A belief p is designated as JustifiedOut in a knowledge state KS if it is disbelieved, and it *is* the case that MustOut(p, KS). This means that there is some precarious NAND-set $S \subseteq I$ that has p as a culprit.

Definition 4.2.20 $\text{JustifiedOut}(p, KS) \equiv p \in X \ \wedge \ \text{MustOut}(p, KS)$.

The following observation is a tautology expressed using the binary connective *XOR* (*exclusive or*), meaning one or the other but not both. A base belief that is not in the current base is either BadOut or JustifiedOut, but it cannot be both.

Observation 4.2.21 $(\forall p \in X)$*:* *BadOut*(p, KS) *XOR* *JustifiedOut*(p, KS) *.*

The value of MustOut, BadOut, or JustifiedOut for any belief in a DDR knowledge state is not affected by the makeup of the priority queue. In other words: two DDR knowledge states, KS and KS_1, that differ *only* in their priority queues have identical bases and exbases, and each individual belief's designation as MustOut, BadOut, or JustifiedOut is the same for KS as it is for KS_1.

Observation 4.2.22 *Given two DDR knowledge states* KS *and* KS_1*, where* $B = B_1$*,* $B^{\cup} = B_1^{\cup}$*,* $I = I_1$*,* $\succeq = \succeq_1$*, and* $Q \neq Q_1$*:* $X = X_1$ *and* $(\forall p)$*:* *MustOut*$(p, KS) \equiv$ *MustOut*(p, KS_1)*, BadOut*$(p, KS) \equiv$ *BadOut*(p, KS_1)*, and JustifiedOut*$(p, KS) \equiv$ *JustifiedOut*(p, KS_1) *.*

4.2.7 DDR Belief Credibility and Base Credibility, Ordering, and Optimality

The definitions and observations in this section all assume a DDR knowledge state KS containing the elements $B, B^{\cup}, I, \succeq$, and Q.

A numerical value for credibility of a belief $p_i \in B^{\cup}$ is as defined in Definition 3.2.1: The credibility value for any belief $p_i \in B^{\cup} = p_1, p_2, \ldots, p_n$ is: $\text{Cred}(p_i, B^{\cup}, \succeq) =_{def} 2^{n-i}$.

It is still the case that $p_i \succ p_j$ IFF Cred$(p_i, B^{\cup}, \succeq)$ $>$Cred$(p_j, B^{\cup}, \succeq)$. As discussed in Chapter 3 (page 71), these credibility values are merely a numerical representation of the linear ordering of the beliefs that simplifies the definitions and proofs of belief base and knowledge state optimality.

The credibility of a DDR base is dependent on I as well as where its beliefs fall in the ordering of $B^{\cup}$.

Definition 4.2.23 Given a DDR knowledge state KS containing the elements B, I, and $B^{\cup} = p_1, p_2, \ldots, p_n$ as ordered by $\succeq$: Cred$(B, B^{\cup}, I, \succeq) =_{def} \sum_{p_i \in B} 2^{n-i}$ (the bit vector indicating the elements in B) when Safe-per-I(KS). Otherwise, when $\neg$Safe-per-I(KS), Cred$(B, B^{\cup}, I, \succeq) = -1$.

This means that, as in Chapter 3, the credibility value of a consistent belief is equivalent to the credibility value of its singleton set: Cred$(p_i, B^{\cup}, \succeq)$= Cred$(\{p_i\}, B^{\cup}, I, \succeq)$. And the credibility value of a *consistent* set is equivalent to the sum of the credibility values of its beliefs (also equivalent to the sum of the credibility values of its singleton subsets): Cred$(B, B^{\cup}, I, \succeq)$ = $\sum_{p_i \in B}$Cred$(p_i, B^{\cup}, \succeq)$.

A linear ordering over DDR bases (overloading the relation $\succeq$) is defined similarly to that in Chapter 3 (Def 3.2.4):

Definition 4.2.24 Given two DDR knowledge states, KS and KS_1, where $B^{\cup} = B_1^{\cup}$, $I = I_1$, and $\succeq = \succeq_1$:

$B \succeq B_1$ IFF Cred$(B, B^{\cup}, I, \succeq) \geq$ Cred$(B_1, B^{\cup}, I, \succeq)$.

Likewise, $B \succ B_1$ IFF Cred$(B, B^{\cup}, I, \succeq) >$ Cred$(B_1, B^{\cup}, I, \succeq)$.

Note that $B \succ B_1$ implies $B \succeq B_1$.

If I is complete, the preference ordering over bases is also consistency-oriented—i.e., ordered by $\succeq_{\perp}$ and/or $\succ_{\perp}$, as defined in Def 3.2.4.

Observation 4.2.25 *If I = Complete-I(KS), then $B \succeq B' \Rightarrow B \succeq_{\perp} B'$ and $B \succ B' \Rightarrow B \succ_{\perp} B'$.*

A DDR base is Optimal-per-I if it is the maximal subset of $B^{\cup}$ provided it is Safe-per-I:

Definition 4.2.26 Optimal-per-I(B, KS) $\equiv$ $\forall B' \subseteq B^{\cup}$, $B \succeq B'$.

Note that a base that is Optimal-per-I may possibly be inconsistent, because it is only Safe-per-I, but *not* guaranteed to be consistent.

An optimal (Optimal-per-$\perp$) DDR base is defined similarly to the Optimal-per-$\perp$ base in Chapter 3 (cf. Definition 3.2.6):

Definition 4.2.27 Optimal-per-$\perp(B,KS) \equiv \forall B' \subseteq B^{\cup}$, $B \succeq_{\perp} B'$.

Therefore, Optimal(B,KS)=$_{def}$ Optimal-per-$\perp(B,KS)$.

A base that is Optimal-per-$\perp$ is guaranteed to be consistent.

If $I =$ Complete-I(KS), then Optimal-per-I is equivalent to Optimal.

Observation 4.2.28 *$I =$ Complete-I(KS) $\Rightarrow$ (Optimal-per-I$(B,KS) \equiv$ Optimal-per-$\perp(B,KS)$).*

4.2.8 Using Q

The priority queue, Q, is used to store beliefs that need to be processed by DDR. These beliefs are processed in order from most credible to least credible.

The uses of the tags are:

- Any belief p that is removed from the base is inserted into Q with the tag *justout*. This pair will be processed *in turn* to determine if any *weaker* base beliefs that are disbelieved should be considered for possible return to the base.
- If the processing of some $\langle p, justout \rangle \in Q$ determines that some $q \in X$ might be eligible for return to the base, then $\langle q, in? \rangle$ is inserted into Q (unless q is already on Q with a tag of *justout*, in which case q's tag is changed to *both*). After $\langle p, justout \rangle$ is processed, it is removed from Q.
- Processing some $\langle q, in? \rangle \in Q$ determines whether q can *actually* return to the base or not. A belief can return either (1) if it does *not* raise an inconsistency or (2) if any inconsistencies raised can be resolved by retracting *only* weaker beliefs. After $\langle q, in? \rangle$ is processed, it is removed from Q.

- If $\langle p, both \rangle \in Q$ then it is first processed to see if p can return to the base (as if it were tagged as *in*?). Only if p *cannot* return to the base, is it then processed as if it were tagged *justout*. After $\langle p, both \rangle$ is processed, it is removed from Q.

4.2.9 Functions for the Priority Queues: Q and *NAND-set-Q*

Definition 4.2.29 The list below comprises the functions for *any* priority queue, QUEUE, (whether it is a queue of belief-tag pairs or of NAND-sets) that I will be using in this dissertation. Although most are well known in the computer science field, I define them here to insure their use is understood.

- **Empty**(QUEUE) returns *true* if QUEUE is empty, else *false*. This is non-destructive.
- **First**(QUEUE) returns the first element in QUEUE (non-destructively), unless Empty(QUEUE), in which case it returns *false*.
- **Rest**(QUEUE) returns a queue just like QUEUE but without its first element (non-destructively). If Empty(QUEUE), it returns *false*.
- **Insert**(ELEMENT, QUEUE,$\succeq$) destructively alters QUEUE and returns the modified queue (MOD-QUEUE) that now contains ELEMENT in its proper place per $\succeq$. If ELEMENT is already on QUEUE, it simply returns QUEUE. Note: ($\forall e \neq$ELEMENT): $e \in$ MOD-QUEUE $\equiv e \in$ QUEUE.
- **Pop**(QUEUE) destructively removes the first element in QUEUE, and returns it. The return is identical to that for First(QUEUE), but in this case, the first element has been actually removed from the queue—modifying the queue to be equivalent to Rest(QUEUE). If Empty(QUEUE), it returns *false*. Regarding the elements in the modified queue (MOD-QUEUE): ($\forall e \neq$First(QUEUE)): $e \in$ MOD-QUEUE $\equiv e \in$ QUEUE.
- **Popped**(QUEUE) destructively removes the first element in QUEUE, then returns the modified queue (MOD-QUEUE). The return is identical to that for Rest(QUEUE), but the operation is destructive. If Empty(QUEUE), it returns *false*. Note: ($\forall e \neq$ First(QUEUE)): $e \in$ MOD-QUEUE $\equiv e \in$ QUEUE.

Definition 4.2.30 Additional functions for a knowledge state priority queue Q include:

- **New-Q( )** returns a DDR knowledge state queue that is empty.
- **Bel**$(Q) =_{def} \{p \mid \langle p, \tau \rangle \in Q\}$.
- **Find**$(p, Q, \succeq)$ returns the tuple $\langle p, \tau \rangle$ non-destructively.
- **Change-Tag**$(p, Q, \tau, \succeq)$ destructively alters the queue Q by changing the tag for p to τ, and returns the new queue. If $p \notin \text{Bel}(Q)$, it returns Q. Note: $(\forall q \neq p, \forall \tau) : \langle q, \tau \rangle \in Q \equiv \langle q, \tau \rangle \in Q_1$.
- **Merge-Qs**$(Q_1, Q_2, \succeq)$ non-destructively merges two priority queues (of the type used in a DDR knowledge state) into one priority queue, provided they have no beliefs in common, and they are both ordered by the same linear ordering $\succeq$. Merge-Qs returns the resulting merged queue.
- **DDR-Q-Insert** $(p, Q, \succeq)$ destructively alters the priority queue Q (changing it from Q_{pre} to Q_{post}) so that p is in the priority queue Q_{post} with a tag to indicate that it should be considered for possible return to a base. The resulting tag will depend on whether p is in Q_{pre} and, if so, with what tag:

  if $\langle p, in? \rangle \in Q_{pre}$, **then** $Q_{post} = Q_{pre}$

  if $\langle p, both \rangle \in Q_{pre}$, **then** $Q_{post} = Q_{pre}$

  if $\langle p, justout \rangle \in Q_{pre}$, **then** $Q_{post} = \text{Change-Tag}(p, Q_{pre}, both, \succeq)$.

  if $p \notin Q_{pre}$, **then** $Q_{post} = \text{Insert}(\langle p, in? \rangle, Q_{pre}, \succeq)$.

  DDR-Q-Insert returns the altered queue. When a belief is inserted into a priority queue using DDR-Q-Insert, I refer to it as being "DDR-Q-Inserted" into the queue.

4.2.10 DDR Invariants

For DDR to work, there are two requirements:

- the knowledge state priority queue, Q, must always be properly maintained, and
- the knowledge state KS must also be properly maintained *whenever the DDR algorithm is running*.

Helper Predicate: Protected-by-Q

If a belief in the DDR knowledge state is designated as "BadOut", it must be *protected* by some presence in the priority queue that will insure that the belief can return to the base.[8] This protection happens in at least one of the following ways:

1. The belief is on Q with a tag of either *in*? or *both*.

2. At least one of its IC-NAND-sets, N, has at least one non-culprit belief also in X, and every non-culprit belief in $N \cap X$ is on Q with a tag of either *justout* or *both*.[9]

This concept of being protected is formalized in the definition below:

Definition 4.2.31 Given a DDR knowledge state *KS* with the elements $B, B^{\cup}, I, \succeq$, and Q,

Protected-by-Q$(p, KS) \equiv$

Q1	(**or** $\langle p, in?\rangle \in Q$	;;; p protects itself
Q2	$\langle p, both\rangle \in Q$	;;; p protects itself
Q3	$\exists S$ s.t (**and** $S \in I$	;;; p protected by one of its IC-NAND-sets
Q4	Weakest$(S, \succeq) = p$	
Q5	$(S \setminus \{p\}) \cap X \neq \emptyset$	
Q6	$\forall q \in ((S \setminus \{p\}) \cap X)$:	
Q7	(**or** $\langle q, justout\rangle \in Q$	
Q8	$\langle q, both\rangle \in Q$))))	

[8]The BadOut belief will return to the base, unless (prior to that return) changes to the knowledge state result in its status changing to "JustifiedOut".

[9]Item 2 insures that either: (a) all more credible beliefs in that NAND-set are returned to the base making p "JustifiedOut"; or (b) one of the non-culprit beliefs is processed as *justout* during DDR which results in p getting a tag of either *in*? or *both* (as determined by DDR-Q-Insert), which will result in its return to the base (provided it is still "BadOut" when it is processed).

DDR Invariant Predicates: Proper-Q and Proper-KS

A priority queue, Q, for a knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ is considered to be properly maintained if every belief p that is "BadOut" is protected by that Q.

Definition 4.2.32 Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q,

Proper-Q(KS) $\equiv (\forall p)$: BadOut(p, KS) $\Rightarrow$ Protected-by-Q(p, KS).

A properly maintained knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ has no active elements of I and has a properly maintained priority queue Q:

Definition 4.2.33 Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q,

Proper-KS(KS) $\equiv$

P1 (**and** Safe-per-I(KS)

P2 Proper-Q(KS))

Note: Proper-Q is always maintained, because it is computationally expensive to regain[10]—it is essential to the efficiency of DDR. Proper-KS, however, can lapse as long as DDR is not being performed. This means that although the queue is maintained, the base of a knowledge state may become inconsistent due to active NAND-sets. Before DDR can be run, however, the system must insure (by retracting beliefs, if necessary) that no NAND-set is active.

Theorems Using Proper-Q and Proper-KS

A proper DDR knowledge state with a complete I is consistent (has a consistent base).

Theorem 4.2.34 *Given a DDR knowledge state KS,*

Proper-KS$(KS) \wedge I =$ *Complete-I(KS)* $\Rightarrow$ *Consistent(KS).*

[10]Proper-Q can be regained efficiently in the case where there has been a re-ordering of the base beliefs as discussed in Section 4.5.1, provided the old ordering/culprit information is still available to reference and the changes to the ordering are relatively minor.

Proof.

(Proof-by-Refutation): Given Proper-KS(KS) and $I =$ Complete-I(KS), assume $\neg$Consistent(KS). Consistent(KS) $\equiv B \nvdash \perp$ (Def 4.2.9). Therefore, $\neg$Consistent(KS) $\equiv B \vdash \perp$. Therefore, $B \vdash \perp$. Proper-KS(KS) $\equiv$ Safe-per-I(KS) $\wedge$Proper-Q(KS)) (Def 4.2.33). Therefore, Safe-per-I(KS). Safe-per-I(KS) $\equiv (\forall S \in I) :$ $\neg$Active(S, KS) (Def 4.2.16). Therefore, $(\forall S \in I) :$ $\neg$Active(S, KS). Complete-I(KS) $= \{S \mid S \subseteq B^{\cup}$, $S \vdash \perp$, and $(\forall S' \subsetneq S) : S' \nvdash \perp\}$ (Def 4.2.3). Because $B \vdash \perp$, either (1) $(\forall B' \subsetneq B)[B' \nvdash \perp]$ XOR (2) $(\exists B' \subsetneq B)[B' \vdash \perp \wedge (\forall B'' \subsetneq B')[B'' \nvdash \perp]]$. In other words, since B is inconsistent, either it is minimally inconsistent or one of its subsets (take such a subset to be called B') is minimally inconsistent (per set theory). If (1) is true, then $B \in I$, because $I =$ Complete-I(KS). But, that would mean Active(B, KS) (Def 4.2.12)—causing a contradiction with $(\forall S \in I) :$ $\neg$Active(S, KS). If (2) is true, then $B' \in I$, because $I =$ Complete-I(KS). But, again we'd have Active(B', KS) (Def 4.2.12)—and a contradiction. Therefore, (Proper-KS(KS) $\wedge I =$ Complete-I(KS)) $\Rightarrow$ Consistent(KS). □

If a DDR knowledge state has a properly maintained priority queue that is empty, then all retracted base beliefs must be designated as JustifiedOut.

Theorem 4.2.35 *Given the DDR knowledge state KS (containing the elements $B, B^{\cup}, I, \succeq$, and Q), Proper-Q(KS), and Empty(Q), then $(\forall p \in X)$: JustifiedOut(p, KS), recalling that $X = B^{\cup} \setminus B$.*

Proof.

(Proof-by-refutation) Given the DDR knowledge state KS (containing the elements $B, B^{\cup}, I, \succeq$, and Q), Proper-Q(KS) and Empty(Q), assume $(\exists p \in X) :$ $\neg$JustifiedOut(p, KS). $(\forall p \in X)$: BadOut(p, KS) XOR JustifiedOut(p, KS) (Obs:4.2.21) . Therefore, $(\exists p \in X) :$ BadOut(p, KS). Since Proper-Q(KS), Protected-by-Q(p, KS) (Def 4.2.32). But, based on the definition for Protected-by-Q (Def 4.2.31), there must be some element on Q "protecting" p—at least one of the following statements must be true:

- $\langle p, in?\rangle \in Q$ (line Q1)
- $\langle p, both\rangle \in Q$ (line Q2)
- $(\exists q \in X):\ \langle q,\ justout\rangle \in Q \vee \langle q, both\rangle \in Q$ (lines Q7 or Q8 to satisfy lines Q3-8)

If any of the just listed statements were true, then ¬Empty(Q). Since Empty(Q) is a given, we have a contradiction. Therefore, $(\forall p \in X)$: JustifiedOut(p, KS). □

If a DDR knowledge state has a properly maintained priority queue, and one belief's tag is changed from *justout* to *both*, then the new knowledge state also has a properly maintained queue.

Theorem 4.2.36 *Given the DDR knowledge state KS (containing the elements $B, B^{\cup}, I, \succeq$, and Q), such that Proper-Q(KS), $\langle p, justout\rangle \in Q$, and KS_1 where $B_1 = B, B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq$, and Q_1 = Change-Tag($p, Q, both, \succeq$): Proper-Q(KS_1).*

The proof for this theorem is in the Appendix in Section A.1.1. Briefly described, the only way that p's presence in Q affects Proper-Q(KS) is if $\langle p, justout\rangle$ satisfies line Q7 in Definition 4.2.31 above for some belief b, where Protected-by-Q(b, KS).[11] If so, $\langle p, both\rangle$ will equally satisfy line Q8 for that same b. Recall that BadOut(b, KS) ≡ BadOut(b, KS_1), because all elements of the knowledge states KS and KS_1 are identical *except for* their queues (cf. Obs 4.2.22). Since all other belief pairs in Q are also in Q_1, all other requirements for Proper-Q(KS_1) are met in the same way that they are met when determining Proper-Q(KS).

For any DDR knowledge state KS where Proper-Q(KS), $p \in X$, and $p \notin Q$: inserting p into the queue will result in a new knowledge state KS_1 s.t. Proper-Q(KS_1).

Theorem 4.2.37 *Given the DDR knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$, Proper-Q(KS), $p \in X$, $p \notin Q$, and $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1 \leftarrow Insert(\langle p, \tau\rangle, Q, \succeq)\rangle$, where $\tau \in \{justout, in?, both\}$: Proper-Q($KS_1$).*

[11] I choose to use b for an arbitrary belief rather than p, which is already used in this discussion, or q, which is used in lines Q6 through Q8 of Definition 4.2.31.

The proof for this theorem is in the Appendix in Section A.1.2. Briefly described, once the conditions for Proper-Q(KS) are met, any other belief $p \notin Q$ can be inserted into the queue without violating the Proper-Q designation—provided the definition of a priority queue (Def 4.2.1) is not violated—i.e., $p \in X$, its tag $\tau \in \{justout, in?, both\}$, and it is inserted in order based on $\succeq$. Essentially, all the elements in KS that satisfied the requirements for Proper-Q(KS) are also contained in KS_1, therefore Proper-Q(KS_1). Note that this depends on KS and KS_1 differing in their queue elements only—all other elements of their DDR knowledge state tuples must be identical.

Based on the previous two theorems, a designation of Proper-Q is maintained through DDR-Q-Insertion.

Theorem 4.2.38 *Given $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$, $p \in X$, and $KS' = \langle B, B^{\cup}, I, \succeq,$ DDR-Q-Insert$(p, Q, \succeq)\rangle$, then Proper-Q(KS) $\Rightarrow$ Proper-Q(KS').*

Proof.

Based on the definition for DDR-Q-Insert (Def 4.2.30):

1. **if** $\langle p, in?\rangle \in Q$, **then** $Q' = Q$
2. **if** $\langle p, both\rangle \in Q$, **then** $Q' = Q$
3. **if** $\langle p, justout\rangle \in Q$, **then** $Q' = \text{Change-Tag}(p, Q, both, \succeq)$
4. **if** $p \notin Q$, **then** $Q' = \text{Insert}(\langle p, in?\rangle, Q, \succeq)$.

Cases (1) and (2) result in KS' = KS, therefore Proper-Q(KS) $\Rightarrow$ Proper-Q(KS'). Case (3): Proper-Q(KS) $\Rightarrow$ Proper-Q(KS') from Theorem 4.2.36. Case (4): Proper-Q(KS) $\Rightarrow$ Proper-Q(KS') from Theorem 4.2.37. □

When altering a knowledge base by merely popping a JustifiedOut first belief off the priority queue, the designation of Proper-Q is not lost if either (1) the belief is tagged as *in*? or (2) all retracted culprits of its NC-NAND-sets are in Q with a tag if *in*? or *both*.

Theorem 4.2.39 *Given* $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$, *Proper-Q(KS)*, $\langle p, \tau\rangle$ = *First(Q), JustifiedOut(p, KS), and* $KS' = \langle B, B^{\cup}, I, \succeq,$ *Popped(Q)*$\rangle$,

then we know that Proper-Q(KS′), provided one of the following is true:

1. $\tau = in?$ *or*

2. $(\forall q \in X) : ((\exists N \in \textit{NC-NAND-sets}(p, KS))\ q = \textit{Culprit}(N, \succeq)) \Rightarrow (\langle q, in?\rangle \in Q \vee \langle q, both\rangle \in Q)$.

Proof.

JustifiedOut(p, KS) and $KS' = \langle B, B^{\cup}, I, \succeq, \text{Popped}(Q)\rangle$ (given). $(\forall q \neq p, \forall \tau) : \langle q, \tau\rangle \in Q \equiv \langle q, \tau\rangle \in Q'$ (Definition of Popped: Def 4.2.29). Since JustifiedOut(p, KS), we know $p \in X$ and $\neg$BadOut(p, KS) (Def 4.2.20, Obs 4.2.21). $(\forall q)$:BadOut(q, KS) $\equiv$ BadOut(q, KS') (Obs 4.2.22). Proper-Q(KS) $\equiv$ $(\forall q)$: BadOut(q, KS) $\Rightarrow$ Protected-by-Q(q, KS) (Def 4.2.32). $(\forall q)$: BadOut(q, KS) $\Rightarrow$ Protected-by-Q(q, KS) (Proper-Q(KS) is given).

1. If $\tau = in?$, then $(\forall q \neq p)$: p is not involved in supporting Protected-by-Q(q, KS), because a belief with tag $in?$ can only "protect" itself (Def 4.2.31). Therefore, $(\forall q \neq p)$: Protected-by-Q(q, KS) $\equiv$ Protected-by-Q(q, KS') (all elements are protected in the same way in both KS, and KS'). Thus, Proper-Q(KS') (Def 4.2.32).

2. $(\forall q \in X) : (\exists N \in \text{NC-NAND-sets}(p, KS))\ q = \text{Culprit}(N, \succeq) \Rightarrow (\langle q, in?\rangle \in Q \vee \langle q, both\rangle \in Q)$. Since the case for $\tau = in?$ is already proved, assume $\tau \in \{both, justout\}$. I must prove that p's removal from Q will not prevent any BadOut belief from being Protected-by-Q. As in case (1) above, if p is not involved in supporting Protected-by-Q(q, KS), then Protected-by-Q(q, KS) $\equiv$ Protected-by-Q(q, KS'). Let q be an arbitrary belief such that BadOut(q, KS) and $\langle p, \tau\rangle \in Q$ is involved in "protecting" q. Since BadOut(q, KS), $q \in X$ (Def 4.2.19). Since $\neg$BadOut(p, KS), $q \neq p$. Therefore, the only protection p can offer is if $(\exists S \in I) : \{p, q\} \subseteq S, q =$Weakest($S, \succeq$), and $(\forall s \in ((S \setminus \{q\}) \cap X)) : \langle s, justout\rangle \in Q \vee \langle s, both\rangle \in Q$ (Def 4.2.31). Take such an S. Then we know q = Culprit($S, \succeq$) and $S \in$ NC-

NAND-sets(p,KS)) (Def 4.2.6,Def 4.2.11). Therefore, $\langle q, in?\rangle \in Q \vee \langle q, both\rangle \in Q$. And, $\langle q, in?\rangle \in Q' \vee \langle q, both\rangle \in Q'$, because the only change from Q to Q' is the removal (pop) of p. So, Protected-by-Q(q,KS') (Def 4.2.31). Therefore, $(\forall q)$: BadOut(q,KS') $\Rightarrow$ Protected-by-Q(q,KS'). Thus, Proper-Q(KS') (Def 4.2.32). □

Given a DDR knowledge state KS, where Proper-Q(KS), returning any retracted base belief p to the base results in a new knowledge state KS_1 where Proper-Q(KS_1), provided $p \notin Q_1$. Recall that $X = B^{\cup} \setminus B$.

Theorem 4.2.40 *Given the DDR knowledge states $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ and $KS_1 = \langle B_1, B^{\cup}, I, \succeq, Q_1\rangle$, where Proper-Q(KS), $p \in X$, $p \in B_1$, $p \notin Q_1$ and $(\forall q \neq p, \forall \tau) : (q \in B \equiv q \in B_1) \wedge (\langle q,\tau\rangle \in Q \equiv \langle q,\tau\rangle \in Q_1)$; then following statement is true: Proper-Q(KS_1).*

The proof for this theorem is extensive and can be found in the Appendix in Section A.1.3.

Given a DDR knowledge state KS, where Proper-Q(KS), removing a set of base beliefs $R \subseteq B$ from the base results in a new knowledge state KS_1 where Proper-Q(KS_1), provided every belief in R is in the queue with the tag *both*.

Theorem 4.2.41 *Given the DDR knowledge states $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ and $KS_1 = \langle B_1, B^{\cup}, I, \succeq, Q_1\rangle$, where Proper-Q(KS), $R \subseteq B$, $R \cap B_1 = \emptyset$, and $(\forall q \notin R, \forall \tau, \forall r \in R) : (q \in B \equiv q \in B_1) \wedge (\langle q,\tau\rangle \in Q \equiv \langle q,\tau\rangle \in Q_1) \wedge \langle r, both\rangle \in Q_1$; then following statement is true: Proper-Q(KS_1).*

Proof.

This proof uses the definition for Protected-by-Q (Def 4.2.31) and refers to the lines in that definition as $Q\#$. The goal is to show that $(\forall b)$: BadOut(b,KS_1) $\Rightarrow$ Protected-by-Q(b,KS_1), and, therefore Proper-Q(KS_1) (Def 4.2.32). Because Proper-Q(KS), it is known that $(\forall b)$: BadOut(b,KS) $\Rightarrow$ Protected-by-Q(b,KS) (Def 4.2.32). For any $r \in R$, if BadOut(r,KS_1), then $\langle r, both\rangle \in Q_1$ satisfies line $Q2$. For any other belief $b \in (X_1 \setminus R)$, we know that $b \in X$ (premise).[12] Let b be any arbitrary belief such that $b \in X_1 \setminus R$. If BadOut(b, KS_1), then (1) BadOut(b, KS) XOR (2) JustifiedOut(b, KS) (Obs 4.2.21). Regarding case(1), if Protected-by-Q(b, KS) is satisfied using line $Q1$ or $Q2$, then Protected-by-Q(b, KS_1) is likewise satisfied. Otherwise, for case(1), Protected-by-Q(b, KS) is satisfied through some set $S \in I$ as described in lines $Q3$-$Q8$. Take such an S. If $R \cap S = \emptyset$, then Protected-by-Q(b, KS_1) is satisfied in the same way as Protected-by-Q(b, KS)—recall that all beliefs in X remain in X and experience no tag changes (premise). If $R \cap S \neq \emptyset$, then $(\forall r \in R \cap S)[\langle r, both\rangle \in Q_1]$ and $(\forall p \in (S \cap X) \setminus (R + b))[\langle p, justout\rangle \in Q_1 \vee \langle p, both\rangle \in Q_1]$ (again, because the elements in X and Q are also in X_1 and Q_1). This means that Protected-by-Q(b, KS_1) is satisfied through that same S that satisfies Protected-by-Q(b, KS) through lines $Q3$-$Q8$. Case (2) states that JustifiedOut(b, KS); so, BadOut(b, KS_1) means there must be an $S \in I$, where $b = Culprit(S, \succeq)$, $(S \setminus \{b\}) \subseteq B$ (i.e., $(S \setminus \{b\}) \cap X = \emptyset$), but $R \cap S \neq \emptyset$ (i.e., $(S \setminus \{b\}) \cap X \neq \emptyset$). In this case, the premise that $(\forall r \in R \cap S)[\langle r, both\rangle \in Q_1]$ is sufficient to insure that Protected-by-Q(b, KS_1) holds—by satisfying line $Q8$ of the $Q3$-$Q8$ condition. Therefore, $(\forall b)$: BadOut$(b, KS_1) \Rightarrow$ Protected-by-Q(b, KS_1). Therefore, Proper-Q(KS_1) (Def 4.2.32). □

Given a DDR knowledge state KS, where Proper-Q(KS), any p that is on the priority queue with a tag of *both* and is also designated as JustifiedOut, can have its tag changed to *justout* and the resulting knowledge state will retain the Proper-Q designation.

Theorem 4.2.42 *Given the DDR knowledge states $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ and $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1\rangle$, where $\langle p, both\rangle \in Q$, JustifiedOut$(p, KS)$, $\langle p, justout\rangle \in Q_1$, and $(\forall q \neq p, \forall \tau) : \langle q, \tau\rangle \in Q \equiv \langle q, \tau\rangle \in Q_1$; then the following statement is true: Proper-Q(KS_1).*

Proof.

Again, I will reference lines of the definition for Protected-by-Q (Def 4.2.31), using the convention Q#,

[12] Recall that for any KS, $X =_{def} B^{\cup} \setminus B$ (Def 4.2.1).

where # is the number of the line. Given the premise Proper-Q(KS), it is known that $(\forall b)$: BadOut$(b, KS) \Rightarrow$ Protected-by-Q(b, KS) (Def 4.2.32). Regarding any p where $\langle p, both\rangle \in Q$ and JustifiedOut(p, KS), it is clear that $\neg$BadOut(p, KS) (Obs 4.2.21). For any other $b \neq p$, BadOut$(b, KS) \Rightarrow$ Protected-by-Q(b, KS). BadOut$(b, KS) \equiv$ BadOut(b, KS_1), because B is the base for both KS and KS_1 (premise). Therefore, BadOut$(b, KS_1) \Rightarrow$ Protected-by-Q(b, KS). If Protected-by-Q(b, KS) is satisfied using line Q1 or Q2, then Protected-by-Q(b, KS_1) is likewise satisfied. Otherwise, Protected-by-Q(b, KS) is satisfied through some set $S \in I$ as described in lines Q3-Q8. Take such an S. If $p \notin S$, then Protected-by-Q(b, KS_1) is satisfied in the same way as Protected-by-Q(b, KS)—recall that all beliefs in X remain in X and experience no tag changes (premise). If $p \in S$, then $\langle p, both\rangle \in Q$ contributes to satisfying Protected-by-Q(b, KS) at line Q8. Similarly, $\langle p, justout\rangle \in Q_1$ contributes to satisfying Protected-by-Q(b, KS_1) at line Q8. As before, $(\forall s \in (S \cap X) \setminus \{b, p\}) : [\langle s, justout\rangle \in Q_1 \lor \langle s, both\rangle \in Q_1]$, because $(\forall s \in (S \cap X) \setminus \{b, p\}) : [\langle s, justout\rangle \in Q \lor \langle s, both\rangle \in Q]$, and the beliefs in $X \setminus \{p\}$ and their tags are not altered. And, therefore, $(\forall b)$: BadOut$(b, KS_1) \Rightarrow$ Protected-by-Q(b, KS_1). Thus, Proper-Q(KS_1) (Def 4.2.32). $\square$

4.2.11 Knowledge State Preference Ordering and the Impact of Q

Given a DDR knowledge state with the elements $B, B^{\cup}, I, \succeq$, and Q, the impact that the priority queue Q can have on KS (*QImpact*) is proportional to the credibility of its first tagged belief (Def 3.2.1).

Definition 4.2.43 Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, if Empty(Q), QImpact(KS) $=_{def}$ 0;

otherwise, given First$(Q) = \langle p, \tau\rangle$, then QImpact$(KS) =_{def}$ Cred$(p, B^{\cup}, \succeq)$.

Assuming a knowledge state has a consistent base and a non-empty queue, popping the first element off the queue results in a knowledge state with a reduced QImpact value.

Observation 4.2.44 *Given the DDR knowledge states KS and* KS_1*, where* $B_1 = B, B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq$ *,* Q_1 *=Popped(Q), and ¬Empty(Q):   QImpact*(KS) *>QImpact*(KS')*.*

The impact of Q is incorporated into the determination of a knowledge state's relative credibility, though that credibility is largely determined by the credibility of its belief base (Cred$(B, B^{\cup}, I, \succeq)$: Def 4.2.23). Note that two knowledge states can be compared only if they have the same $B^{\cup}$ and $\succeq$.

Definition 4.2.45 Given a knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, if ¬Safe-per-I(KS), then Cred($B, B^{\cup}, I, \succeq$)= -1, and $\nexists KS'$ s.t. $KS \succeq_{KS} KS'$. Otherwise, Safe-per-I(KS), and given a second knowledge state KS_1 where $B_1^{\cup} = B^{\cup}, I_1 = I$, and $\succeq_1 = \succeq$, and Proper-Q(KS) and Proper-Q(KS_1):

- If $B_1 = B$ and QImpact(KS) = QImpact(KS_1), then $KS \succeq_{KS} KS_1$ and $KS_1 \succeq_{KS} KS$. Their queues may differ, as long as the first elements in their queues are the same.
- $KS \succ_{KS} KS_1$ if either
  - $B \succ B_1$ OR
  - $B = B_1$ and QImpact(KS) $<$ QImpact(KS_1).
- If $KS \succ_{KS} KS_1$, then $KS \succeq_{KS} KS_1$.

A preference between knowledge states can also be defined in terms of consistency:

Definition 4.2.46 Given a knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, if ¬Consistent(KS), then Cred($B, B^{\cup}, \succeq$)= -1, and $\nexists KS'$ s.t. $KS \succeq_{\perp} KS'$. Otherwise, Consistent(KS), and given a second knowledge state KS_1 where $B_1^{\cup} = B^{\cup}, I_1 = I$, and $\succeq_1 = \succeq$, and Proper-Q(KS) and Proper-Q(KS_1):

- If $B_1 = B$ and QImpact(KS) = QImpact(KS_1), then $KS \succeq_{\perp} KS_1$. Their queues may differ, as long as the first elements in their queues are the same.
- $KS \succ_{\perp} KS_1$ if either
  - $B \succ_{\perp} B_1$ OR
  - $B = B_1$ and QImpact(KS) $<$ QImpact(KS_1).

- If $KS \succ_{\perp} KS_1$, then $KS \succeq_{\perp} KS_1$.

Popping the first element off the non-empty queue of a knowledge state that is Safe-per-I results in a strictly more preferred knowledge state, provided *both* states are Proper-Q.

Observation 4.2.47 *Given the DDR knowledge states KS and KS_1, where $B_1 = B, B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq, Q_1 =$Popped(Q), Safe-per-I(KS), Proper-Q(KS), Proper-Q(KS′), and ¬Empty(Q): $KS' \succ_{KS} KS$.*

Popping the first element off the non-empty queue of a knowledge state that is *consistent* results in a strictly more preferred knowledge state, provided *both* states are Proper-Q.

Observation 4.2.48 *Given the DDR knowledge states KS and KS_1, where $B_1 = B, B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq, Q_1 =$Popped(Q), Consistent(KS), Proper-Q(KS), Proper-Q(KS′), and ¬Empty(Q): $KS' \succ_{\perp} KS$.*

Since QImpact(*KS*) is determined by First(*Q*), different queues with identical first elements will have the same QImpact value. $\succeq_{KS}$ is a *preorder*—it is *not linear*.[13] The QImpact value of a knowledge state is an indication of the *impact* the priority queue may have on the beliefs in the base when DDR processes the knowledge state towards the optimal base. The higher the preference of the top element, the greater the number of beliefs (and the greater their possible preference) that might be moved in or out of the base to make it optimal.

The QImpact value of a knowledge state is not only a measure of how much DDR might affect the base of the knowledge state, it is also a reverse measure of the confidence in the current base and reasoning performed using that base. The higher the QImpact value, the lower the confidence that the base is optimal. Any reasoning that is grounded in a set of base beliefs that are *all* more preferred to the top element in *Q* is reasoning performed in an optimal belief space and will survive the DDR process. Actually, reasoning

[13]Consider KS_1 and KS_2, where $B_1 = B_2$, $Q_1 \neq Q_2$, but First(Q_1) = First(Q_2) (only the beliefs need be equal here, not the tags). In this case, $KS_1 \neq KS_2$, but $KS_1 \succeq KS_2$ and $KS_2 \succeq KS_1$.

with non-culprit base beliefs is always safe, since they never are retracted from the base (assuming $I =$ Complete-I(KS) and no new input forces them out). Reasoning with any beliefs derived using some culprit base belief that is not more credible than the top element in Q is *potentially* faulty reasoning with regard to the optimal base (the base that is optimal w.r.t. $B^{\cup}$ and $\succeq$).

Observation 4.2.49 *Given the DDR knowledge states KS and KS_1, where $B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq$, and Safe-per-I(KS):* $KS \succ_{KS} KS_1 \equiv ((B \succ B_1) \vee (B = B_1 \wedge QImpact(KS_1) > QImpact(KS)))$.

Observation 4.2.50 *Given the DDR knowledge states KS and KS_1, where $B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq$, and Safe-per-I(KS):* $((B \succeq B_1) \wedge (QImpact(KS_1) \geq QImpact(KS))) \Rightarrow KS \succeq_{KS} KS_1$.

A knowledge state KS is Optimal-per-I (w.r.t. $B^{\cup}$ and $\succeq$) if it is preferred over all other knowledge states with the same $B^{\cup}$ and $\succeq$:

Definition 4.2.51 Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q,

Optimal-per-I(KS) $\equiv (\forall KS_1) : (B_1^{\cup} = B^{\cup} \wedge I_1 = I \wedge \succeq_1 = \succeq) \Rightarrow KS \succeq KS_1$.

A knowledge state KS is optimal (a.k.a. Optimal-per-$\perp$) w.r.t. $B^{\cup}$ and $\succeq$ if it is preferred (per $\perp$) over all other knowledge states with the same $B^{\cup}$ and $\succeq$:

Definition 4.2.52 Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q,

Optimal-per-$\perp$(KS) $\equiv (\forall KS_1) : (B_1^{\cup} = B^{\cup} \wedge I_1 = I \wedge \succeq_1 = \succeq) \Rightarrow KS \succeq_{\perp} KS_1$.

Based on Observation 4.2.28, if $I =$ Complete-I(KS), then a knowledge state that is Optimal-per-I is also Optimal-per-$\perp$:

Observation 4.2.53 *Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, $I =$ Complete-I(KS) $\Rightarrow$ (Optimal-per-I(KS) $\equiv$ Optimal-per-$\perp$(KS)).*

If a DDR knowledge state is Proper-KS and has an empty queue, then it is Optimal-per-I.

Theorem 4.2.54 *Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, where Proper-KS(KS): Empty(Q) $\Rightarrow$ Optimal-per-I(KS)*

Proof.

Proper-KS(KS) and Empty(Q) (given, premise). Therefore, Safe-per-I(KS) (Def 4.2.33). Proper-Q(KS) $\wedge$ Empty(Q) $\Rightarrow$ $(\forall p \in X)$: JustifiedOut(p, KS) (Thm 4.2.35). $(\forall p \in X, \exists S \in I): \; S \setminus \{p\} \subseteq B \; \wedge \; p =$ Culprit$(S, \succeq)$ (Def 4.2.18,Def 4.2.20) . Therefore, $(\forall p \in X, \exists S \in I): \; S \subseteq (B \cup \{p\}) \; \wedge \; p =$ Culprit$(S, \succeq)$. Therefore, $(\forall p \in X) : (B \cup \{p\}) \vdash \perp$ (Def of I: Def 4.2.1). If $S \in I$ and $p =$ Culprit$(S, \succeq)$, then $(\forall q \in S)$: $q \neq p \Rightarrow q \succ p$ (Def 4.2.4,Def 4.2.6). Therefore, $((B \cup \{p\}) \setminus \{q \mid p \succ q\}) \vdash \perp$. Therefore, $(\nexists B' \subseteq B^{\cup})$: Cred$(B', B^{\cup}, I, \succeq)>$Cred$(B, B^{\cup}, I, \succeq)$, because any belief in X added to B makes the resulting base ¬Safe-per-I (resulting in a credibility of -1) unless a *stronger* belief is removed (resulting in a lower credibility than B)—so no change to the base can increase its credibility. Therefore, $(\forall B' \subseteq B^{\cup}) : B \succeq B'$. Empty($Q_{KS}$)$\Rightarrow$ QImpact(KS) = 0 (Def 4.2.43). ¬Empty(Q_{KS}) $\wedge$ $\langle p, \tau \rangle$ = First(Q_{KS}) $\Rightarrow$ QImpact(KS) = Cred$(p, B^{\cup}, \succeq)$ (Def 4.2.43). Therefore, $(\forall KS')$: QImpact(KS')$\geq$ QImpact(KS). Therefore, ($\forall KS'$ where $B^{\cup}_1 = B^{\cup} \wedge I_1 = I \wedge \succeq_1 = \succeq$) : $B \succeq B' \wedge$ QImpact(KS')$\geq$ QImpact(KS). Therefore, ($\forall KS'$ where $B^{\cup}_1 = B^{\cup} \wedge I_1 = I \wedge \succeq_1 = \succeq$) : $KS \succeq_{KS} KS'$ (from Obs 4.2.50). Optimal-per-I(KS) $\equiv$ $(\forall KS_1) : (B^{\cup}_1 = B^{\cup} \wedge I_1 = I \wedge \succeq_1 = \succeq) \Rightarrow KS \succeq KS_1$ (Def 4.2.51). Therefore, Optimal-per-I(KS). Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, where Proper-KS(KS): Empty(Q) $\Rightarrow$ Optimal-per-I(KS). □

If a DDR knowledge state is Proper-KS and has an I that is complete and an empty queue, then it is Optimal-per-$\perp$.

Theorem 4.2.55 *Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, where Proper-KS(KS) and I = Complete-I(KS): Empty(Q) $\Rightarrow$ Optimal-per-$\perp$(KS)*

Proof.

Proper-KS(KS), $I =$ Complete-I(KS), and Empty(Q) (given, premise). Proper-KS(KS) $\wedge$ Empty(Q) $\Rightarrow$ Optimal-per-I(KS) (Theorem 4.2.54). $I =$ Complete-I(KS) $\Rightarrow$ (Optimal-per-I(KS) $\equiv$ Optimal-per-$\perp$(KS)) (Obs 4.2.53). Therefore, Optimal-per-$\perp$(KS). $\square$

4.2.12 DDR Knowledge State Belief Change Operations

Introduction

The operations that can be performed on (and change) a DDR knowledge state, KS, are expansion, consolidation, semi-revision, reconsideration and optimized-addition. These are similar to the operations on a knowledge state triple that were discussed in Chapter 3 (see Section 3.2.3), but the operation of contraction is notably absent. This omission is discussed immediately below.

The operators used to represent these operations will be the same as those used on the knowledge state presented in Chapter 3—and in some cases the same as those operations on a simple belief base—but the context in which each operator is used should disambiguate its usage.

The main belief change operations performed on a DDR knowledge state involve changes to its belief base that are defined in terms of only two *base* belief change operations: expansion ($+$) and/or kernel consolidation (!) [Hansson1994]. When the base of a DDR knowledge state is altered, however, there may be changes to other elements of the tuple as well.

All changes made to the priority queue during these operations are to insure that the resulting knowledge state (KS_1) satisfies Proper-Q(KS_1)—assuming that the starting knowledge state (KS) similarly satisfied Proper-Q(KS).

Knowledge State Contraction Performed for Consistency Maintenance Only

There are two reasons for not defining an operation of contraction for a knowledge state implemented for use with DDR.

First, I assume that any system implemented to include reconsideration is designed with optimality as a high priority. When the operation of contraction is performed outside of consistency maintenance, there is a potential for producing a sub-optimal base. As explained in [Chopra, Georgatos, & Parikh2001] "Agents do not lose beliefs without a reason: to drop [i.e., retract] the belief ... p is to *revise* by some information that changes our reasoning." Extending this concept, reconsideration supports the idea that beliefs should not *remain* lost (retracted) without a reason.[14] Contraction occurs as a by-product of the consistency maintaining operations of consolidation, reconsideration, and optimized-addition.

Second, the algorithm for performing dependency-directed reconsideration (DDR) could adjust to any user initiated contraction as long as the contracted belief (along with any other contracted beliefs) is inserted into the priority queue Q with a tag of *both* at the time of contraction. If the contraction was justified, the belief will remain contracted when DDR is performed. If it does not need to be removed for consistency maintenance, the contracted belief will be returned to the base by DDR.[15] Whether the user initiated contraction holds or not, the result is the optimal base which is optimal w.r.t. $B^{\cup}$ and $\succeq$ and unaffected by the contraction. Hence, since DDR eliminates the effects of any contraction, I choose to eliminate it as a possible operation for a system implementing DDR.

[14] Contraction was included in the knowledge state triple belief change operations discussed in Chapter 3 to facilitate the discussion of improved Recovery for belief bases and the comparison of reconsideration to the operations in the Belief Liberation literature.

[15] These DDR predictions assume no other factors occur after the contraction or during DDR (such as the return or removal of some other belief that affects the contracted belief) to change the status of the contraction.

DDR Knowledge State Expansion

The operation of expansion on a DDR knowledge state *KS* (containing $B, B^{\cup}, I, \succeq$, and Q) is the addition of some belief p (along with the preference ordering information regarding that belief, $\succeq_p$), without regard to any inconsistencies that may arise from that addition. Expansion of *KS* by the pair $\langle p, \succeq_p \rangle$ is written as $KS + \langle p, \succeq_p \rangle$ and is the knowledge state version of belief base expansion.[16] The result is a new knowledge state, KS_1 containing the following elements:

- $B_1 = (B + p)$, where $+$ is the operation of belief base expansion
- $B_1^{\cup} = B^{\cup} + p$, where $+$ is the operation of belief base expansion. Note that $X_1 = X$.
- $I_1 = I \cup I'$, where $I' \subseteq \{N \mid N \subseteq B_1^{\cup}$ and $N \vdash \perp\}$ and $(\nexists N, N' \in I_1) : N' \subsetneq N$.
- $\succeq_1$ is $\succeq$ adjusted to include the preference information $\succeq_p$—which positions p relative to other beliefs in $B^{\cup}$, while leaving the order of other beliefs in $B^{\cup}$ unchanged. The resulting ordering is the transitive closure of these orderings.[17] If this re-ordering results in altering the culprit for a NAND-set, a corresponding change is reflected in Q_1, see below.
- Q_1 is similar to Q with the following changes, which insure adherence to the definition of a priority queue and Proper-Q(KS_1): (1) if p was in Q and is now in B_1, then it is *not* in Q_1; (2) $(\forall q \in X)$: if the re-ordering results in q being moved up from the culprit position in a NAND-set to a non-culprit position, then q must be in the queue with a tag indicating it should be considered for *possible* return to the base during DDR.[18]

---

[16] I choose to overload the expansion operator, $+$, because it is clear from the context which expansion operation is being performed—base expansion or DDR-KS expansion. Similarly, I will also be overloading the operators for consolidation (!) and semi-revision ($+_!$).

[17] I assume that if $p \in B^{\cup}$, the location of p in the sequence might change—i.e., its old ordering information is removed before adding $\succeq_p$ and performing closure—but all other beliefs remain in their same relative order.

[18] This insertion would be performed by the procedure DDR-Q-Insert which determines whether the tag for q should be *in?* or *both*. How this assures Proper-Q(KS_1) is discussed in detail in Section 4.5.1 which covers restoring Proper-Q after a re-ordering.

DDR Knowledge State Consolidation

$KS!$ is consolidation of the DDR KS which produces KS_1, where $B_1 = B!$ (recall that $!$ is the operation of belief base kernel consolidation), and Q_1 is similar to Q but adjusted so that: $(\forall p)$ If $p \in B$ and $p \notin B_1$, then $\langle p, justout \rangle \in Q_1$ (assuming JustifiedOut(p, KS_1)). Any beliefs in $B \setminus B_1$ that are not guaranteed to be JustifiedOut should have tags of *both* in the queue (to assure Proper-Q(KS_1)). $B_1^{\cup} = B^{\cup}, I_1 = I$, and $\succeq_1 = \succeq$.

As discussed in Chapter 3, the linear ordering of the beliefs in the base B make it possible (and desirable) for $B!$ to be optimal w.r.t. B and $\succeq$. Note, again, that the optimality is w.r.t. B, not $B^{\cup}$. See Section 3.2.3 for the full discussion.

DDR Knowledge State Kernel Semi-Revision

The operation of kernel semi-revision on a DDR knowledge state KS is the addition of a belief p (along with the preference ordering information regarding that belief, $\succeq_p$) followed by the consistency maintenance operation of consolidation. It is written as $KS +_! \langle p, \succeq_p \rangle$ and is the knowledge state version of Hansson's belief base kernel semi-revision [Hansson1997]. The result is a new knowledge state (KS_1) with the following elements:

- $B_1 = B +_! p$, where $+_!$ is belief base semi-revision: $B +_! p = (B + p)!$, and $!$ is performed using the information in $\succeq_1$. Recall: p is not *guaranteed* to be in B_1
- $B_1^{\cup} = B^{\cup} + p$ (and recall that $X_1 = B_1^{\cup} \setminus B_1$)
- $I_1 = I \cup I'$, where $I' \subseteq \{N \mid N \subseteq B_1^{\cup}$ and $N \vdash \perp\}$ and $(\nexists N, N' \in I_1) : N' \subsetneq N$.
- $\succeq_1$ is $\succeq$ adjusted to include the preference information $\succeq_p$ as described above for knowledge state expansion.

- Q_1 is similar to Q with the following alterations:[19]

  1. if $p \in Q$ and $p \in B_1$, then $p \notin Q_1$
  2. $(\forall q \in X)$: if the re-ordering results in q being moved up from the culprit position in a NAND-set to a non-culprit position, then q must be DDR-Q-Inserted into the queue as discussed for DDR knowledge state expansion.
  3. $(\forall b \in B)$: if $b \in X_1$, then $\langle b, \mathit{justout} \rangle \in Q_1$, provided JustifiedOut($p, KS_1$)—if the system cannot guarantee JustifiedOut(p, KS_1), then the tag for b is *both* (to assure Proper-Q(KS_1)).
  4. If $p \in X_1$, then $\langle p, \mathit{justout} \rangle \in Q_1$

DDR Knowledge State Reconsideration

$KS!_{\cup}$ is reconsideration of the knowledge state KS which produces KS_1, where $B_1 = B^{\cup}!$ (where $!$ is the operation of belief base kernel consolidation), $B_1^{\cup} = B^{\cup}, I_1 = I, \succeq_1 = \succeq$ and Q_1 is empty.

The following theorem is the DDR version of Observation 3.2.15. It states that, assuming a maximizing global decision function for consolidation (that finds the maximal consistent subset of the base being consolidated), reconsideration results in the optimal base (and, thus, the optimal knowledge state), regardless of the makeup of the current belief base (recall that the current base is B, *not* $B^{\cup}$) at the time that reconsideration is performed. In other words, the result of reconsideration on a knowledge state is independent of the makeup of the current base of that knowledge state.

Theorem 4.2.56 *Given* any *DDR knowledge state KS containing the elements* $B, B^{\cup}, I$, $\succeq$, *and Q, where a maximizing global decision function is used for consolidation (one that finds the maximal consistent subset of the base being consolidated) : Optimal-per-*$\perp$*(*$KS!_{\cup}$*).*

Proof.

Given that $KS_1 = KS!_{\cup}$, $B_1 = B^{\cup}!$, by the definition of reconsideration. If consolidation uses a maximizing

[19]The first three alterations come from the expansion and consolidation changes to the priority queue already discussed above.

global decision function that finds the maximal consistent subset of the base being consolidated, $(\forall B' \subseteq B^{\cup})$: $B^{\cup}! \succeq_{\perp} B'$ (Obs 3.2.8). Therefore, $(\forall B' \subseteq B^{\cup}) : B_1 \succeq_{\perp} B'$. Empty($Q_1$) by definition, therefore QImpact(KS_1) = 0 (Def 4.2.43). So, $(\forall KS' = \langle B', B^{\cup}, I, \succeq, Q' \rangle) : KS_1 \succeq_{\perp} KS'$ (Def 4.2.24, Def 4.2.45). Thus, Optimal-per-$\perp$(KS_1) (Def 4.2.52). □

DDR Knowledge State Optimized-Addition

The operation of optimized-addition on a DDR knowledge state KS is the addition of a belief p (along with the preference ordering information regarding that belief, $\succeq_p$) followed by the optimizing operation of reconsideration. It is written as $KS +_{\circlearrowleft} \langle p, \succeq_p \rangle$ and is expansion followed by reconsideration. The result is a new DDR knowledge state, KS_1 with the following elements:

- $B_1 = B^{\cup} +_! p$, where $+_!$ is belief base semi-revision: $(B^{\cup} +_! p = (B^{\cup} + p)!)$, and ! is performed using the information in $\succeq_1$. Note, however, that the new base is a result of semi-revision of $B^{\cup}$ by p (not semi-revision of B by p)—this is the difference between *semi-revision* of a DDR KS and *optimized-addition* to a DDR KS.

  Recall: p is not *guaranteed* to be in B_1, and B_1 is optimal w.r.t. $B_1^{\cup}$ and $\succeq_1$.

- $B_1^{\cup} = B^{\cup} + p$

- $I_1 = I \cup I'$, where $I' \subseteq \{N \mid N \subseteq B_1^{\cup} \text{ and } N \vdash \perp\}$ and $(\nexists N, N' \in I_1) : N' \subsetneq N$.

- $\succeq_1$ is $\succeq$ adjusted to include the preference information $\succeq_p$ as described above for knowledge state expansion.

- Q_1 is empty.

4.3 DDR Algorithms

4.3.1 Introduction

The algorithms discussed in this section assume that the system has a knowledge state *KS* that contains $B, B^{\cup}, I, \succeq$, and Q.

In the following algorithms for implementing DDR, all changes to the system knowledge state are made *ONLY* during the helper procedure Update-KS (which is called in line 18 of the DDR algorithm). Additionally, because this update takes place at the end of each pass through the DDR loop, the current knowledge state during pass n through the DDR loop reflects all the changes made during during the previous $n-1$ DDR loop iterations.

DDR is an anytime algorithm, because it can be stopped at any time without loss of information or damage to the current knowledge state. Although I discuss many aspects of anytime algorithms (with regards to DDR) later in this chapter[20], a brief definition is: “Anytime algorithms [[Dean & Boddy1988]] are algorithms whose quality of results improves gradually as computation time increases” [Zilberstein1996], and the *current* result is available at any point during the algorithm’s execution.

I assume that each line of an algorithm (which include primitive changes to the knowledge state) can be successfully performed to completion, except in the case where it is a call to a helper algorithm (specifically, one of the helper algorithms defined in this section). In this case, every line in that helper algorithm is assumed to be successfully performed to completion. The proofs for the theorems that accompany these algorithms show that if the DDR process is halted *between* any line of the DDR algorithm or the helper algorithms, no anytime benefits are sacrificed, and the status of Proper-KS is maintained.

If the execution of any primitive change to the knowledge state is interrupted, it is possible that the

[20]Cf. Section 4.5.1.

knowledge state may become inconsistent and/or no longer satisfy the Proper-Q requirements. It is up to the system designers to implement safeguards to reduce the likelihood of this happening and/or implement ways for the system to "self-correct" if halted mid-process. I draw special attention to the last line of the Add-Remove algorithm (page 156) as an example of a line resulting in a change to the knowledge state that should *not* be interrupted when only partially completed.

4.3.2 A General Diagram of a System Using DDR

Figure 4.1 is a diagram that illustrates how a system can expand its base beliefs (from KS_0 to KS_1), make belief change decisions in series that may result in a non-optimal state (KS_2, and then KS_3), then use DDR to restore optimality (KS_4). It also shows the DDR process being stopped in the process of improving the belief base credibility (improving from KS_5 but stopped at non-optimal KS_6). More belief change decisions are made that reduce the credibility (KS_7 and KS_8 are less credible than KS_6 *relative* to the optimal knowledge state), and then DDR is called to finish optimizing the knowledge state resulting in KS_9.

4.3.3 Helper-Functions/Procedures

The following procedures and functions are used by the DDR algorithm to gradually improve the credibility of the belief base. To distinguish the knowledge state at the beginning of a procedure from the knowledge state at the end of the procedure, I refer to them as KS_{pre} and KS_{post}, respectively.

- Safe-Return (p, KS) returns the set of weaker beliefs that *must* be removed from the base to eliminate any contradictions caused by returning the retracted belief p to the base.
- KS-Add-Remove(KS, p, R) alters the base by returning p to the base and removing all beliefs in R. It also updates the queue accordingly: $p \notin Q_{post}$ and $(\forall r \in R) : \langle r, justout\rangle \in Q_{post}$
- Process-Justout(KS, p) removes p from the queue and DDR-Q-Inserts retracted culprits of p's NC-NAND-sets into the queue to be considered for possible return.

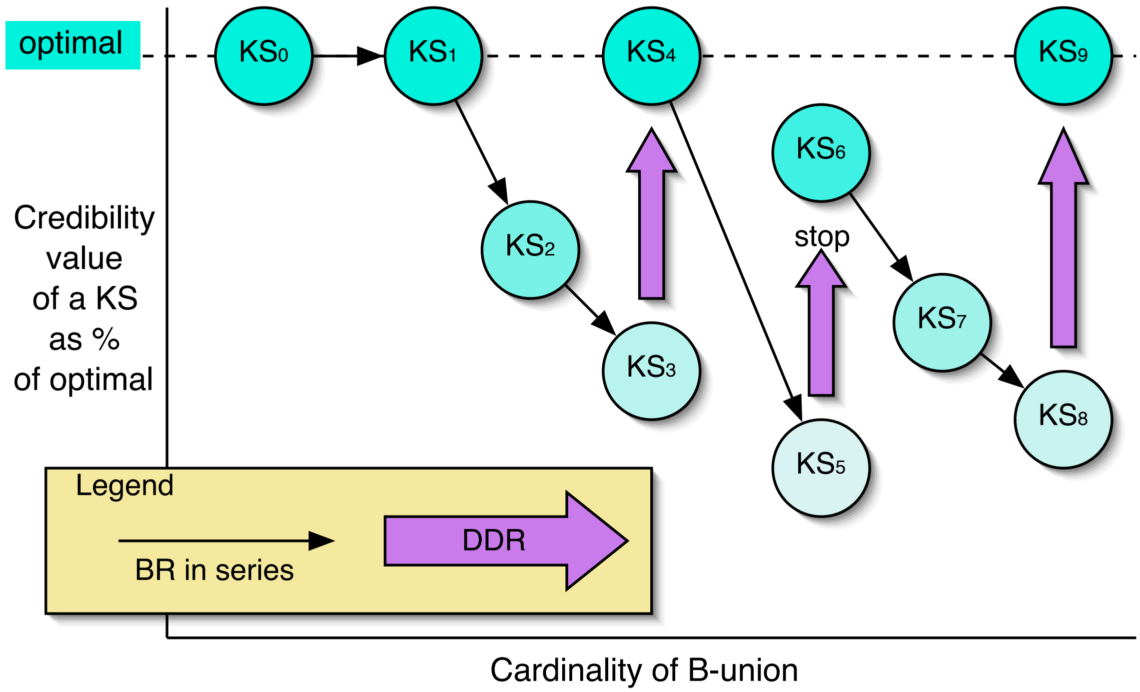

Figure 4.1: A diagram illustrating an overview of a system using DDR. The system can call DDR to optimize the base of a knowledge state (optimal w.r.t. $B^{\cup}$, $\succeq$, and I, for the knowledge state: see KS_3 optimized to KS_4), or it can be stopped during the process (KS_6) so that further reasoning and/or belief change operations can be performed. When convenient, DDR can be recalled and will incrementally improve the credibility of the base until it reaches the optimal knowledge state w.r.t. the current $B^{\cup}$, $\succeq$, and I. DDR *never* makes a change that results in a less credible belief base.

- Update-KS$(KS, update, p, R)$ updates the knowledge state KS to a more credible state. The following values for *update* trigger the related update operation:
  - *update*= *PopQ*: the only change to the knowledge state is the removal the first element in the priority queue;
  - *update*= *ProcJustout*: calls Process-Justout(KS, p);
  - *update*= *AddRem*: calls KS-Add-Remove(KS, p, R).

Note that Safe-Return is the only function, and it does not change the knowledge state. The procedures all change some element of the knowledge state (the parameter KS which is passed by reference).

The only changes by these procedures/functions and DDR that affect the elements of a DDR knowledge state are (occasionally) changes to the base (B) and changes to the priority queue (Q). Although the top element on the queue is removed from the queue with every pass through the DDR loop, the queue may be altered in other ways as well—weaker culprit beliefs may be placed on the queue...

- if they are to be considered for possible return to the base, because the top element is no longer in the base or
- if they are removed from the base so that the top element can return.

The remaining knowledge state five-tuple elements ($B^{\cup}$, I, and $\succeq$) are *unaffected* during the entire DDR process.

For all functions/procedures just discussed, *all parameters are passed by reference*. For example, a pointer to the knowledge state is passed to the function/procedure being called. This way the procedures can make alterations to the knowledge state directly. This saves the computational expense of duplicating the knowledge state or any of its parts when calling procedures. Likewise, a belief that is represented as p in the text may be implemented as a data structure with information to assist locating within or inserting into ordered lists. Similarly, the assignment of the elements of the knowledge state tuple to their respective variables within a procedure (e.g., $\langle B, B^{\cup}, I, \succeq, Q\rangle \leftarrow_{ref} KS$) is also a value copy of the pointer assignment. I emphasize this by the $\leftarrow_{ref}$ symbol, and it insures that any change to Q or B is reflected in KS. Additionally, recall that many other operations that are called are destructive: Pop, Popped, Insert and DDR-Q-Insert.

In the pseudo-code of the algorithms, I use the following conventions:

- $\leftarrow$ is used for assignment by value (e.g., *update*$\leftarrow$*PopQ*)
- $\leftarrow_{ref}$ is used to emphasize pointer assignment (e.g., $\langle B, B^{\cup}, I, \succeq, Q\rangle \leftarrow_{ref} KS$)
- ;;; is used to indicate a comment.

The helper functions and procedures begin on the next page.

Safe-Return (p,KS)

Safe-Return (p,KS) returns a set of beliefs (R), where R contains the beliefs to be removed from $B+p$ for consistency maintenance—it is possible for R to be empty. KS is unaffected by this function.

function Safe-Return(p,KS) ;;; returns a set of beliefs

$\langle B, B^{\cup}, I, \succeq, Q \rangle \leftarrow_{ref} KS$

$R \leftarrow_{ref} \{\ \}$;;; initializing R

$NQ \leftarrow_{ref}$ *NAND-set-Q*(NC-NAND-sets(p,KS) , $\succeq$) ;;; p's NC-NAND-sets in a queue

loop until Empty(NQ)

$N \leftarrow_{ref}$ Pop(NQ) ;;; First element in NQ removed into N

if $(N \setminus \{p\} \subseteq B$ AND $N \cap R = \emptyset)$ **then**

$R \leftarrow_{ref} R \cup \{$Culprit$(N,\succeq)\}$;;; culprit for N inserted into R

endif

end loop

return R

Theorem 4.3.1 *(Proof in Appendix, Section A.2.1) Given KS and p and the following:*

- *Preconditions for Safe-Return(p,KS): $KS = KS_{pre}$ is a DDR knowledge state containing the elements $B, B^{\cup}, I, \succeq$, and Q; Proper-KS(KS_{pre}); BadOut(p, KS_{pre}); $\langle p, \tau \rangle =$ First(Q_{pre}), where $\tau \in \{in?, both\}$.*
- *$R =$ Safe-Return(p,KS).*

then the following postconditions hold:

- *Postconditions for Safe-Return(p,KS) that returns a set of beliefs R: $KS = KS_{post} = KS_{pre}$; Proper-KS(KS); BadOut($p,KS$); $R \subseteq B$; $(\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$; $\sum_{p_i \in ((B+p)\setminus R)} Cred(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} Cred(p_j, B^{\cup}, \succeq)$; $(\forall r \in R,) : p \succ r$; $(\forall r \in R, \exists S \in I) : r =$ Culprit$(S,\succeq) \wedge S \subseteq (((B+p) \setminus R) + r)$.*

KS-Add-Remove(KS, p, R)

KS-Add-Remove(KS, p, R) destructively alters the knowledge state KS (also referred to as KS_{pre}) to be a new knowledge state (KS_{post}) by making the following changes: adding p to the base, removing the set of beliefs R from the base, and adjusting the priority queue to reflect these changes.

procedure KS-Add-Remove(KS, p, R),

$\langle\, B\,,\, B^{\cup}\,,\, I\,,\, \succeq\,,\, Q\,\rangle \leftarrow_{ref} KS$

$Q_1 \leftarrow_{ref}$ New-Q( )

for each r in R **do**

$Q_1 \leftarrow_{ref}$ Insert($\langle r, justout\rangle$, Q_1 , $\succeq$) $\rangle$;;; $\leftarrow_{ref}$ for clarity, Insert is destructive.

endfor

$KS \leftarrow_{ref} \langle\, B \leftarrow_{ref} (B \cup \{p\}) \setminus R,\ B^{\cup}\,,\, I\,,\, \succeq\,,$ Popped(Merge-Qs($Q, Q_1, \succeq$))$\rangle$

Theorem 4.3.2 *(Proof in Appendix, Section A.2.2) Given KS, p, R and the following*

- *Preconditions for KS-Add-Remove(KS, p, R): $KS = KS_{pre}$ is a DDR knowledge state; Proper-KS(KS_{pre}); BadOut(p, KS_{pre}); $\sum_{p_i \in ((B+p)\setminus R)} Cred(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} Cred(p_j, B^{\cup}, \succeq)$; $R \subseteq B_{pre}$; $\langle p, \tau\rangle =$ First(Q_{pre}), where $\tau \in \{in?, both\}$; $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$; and $(\forall r \in R) : p \succ r$, and $(\exists S \in I) : r = Culprit(S, \succeq) \wedge S \subseteq (((B_{pre} + p) \setminus R) + r)$.*
- *KS_{post} is the knowledge state resulting from KS-Add-Remove(KS, p, R)*

then the following postconditions hold:

- *Postconditions for KS-Add-Remove(KS, p, R): KS_{post} is a DDR knowledge state; Proper-KS(KS_{post}); $B_{post} = (B_{pre} \cup \{p\}) \setminus R$; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; $(\forall r \in R)$: JustifiedOut(r, KS_{post}); QImpact(KS_{pre})$>$ QImpact(KS_{post}) ; $B_{post} \succ B_{pre}$; $KS_{post} \succ_{KS} KS_{pre}$; $p \notin Q_{post}$; $(\forall r \in R) : p \succ r \wedge \langle r, justout\rangle \in Q_{post}$; $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau\rangle \in Q_{pre}) \equiv (\langle b, \tau\rangle \in Q_{post}))$.*

Process-Justout(KS, p)

Process-Justout(KS, p) destructively alters the knowledge state KS (also referred to as KS_{pre}) to be a new knowledge state (KS_{post}) by making the following changes: its queue is updated s.t. $p \notin Q_{post}$ and all disbelieved culprits of p's NC-NAND-sets are inserted into the queue by the function DDR-Q-Insert.

procedure Process-Justout (KS, p)

$\langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle \leftarrow_{ref} KS$

for each $N \in$ NC-NAND-sets(p, KS)

$q \leftarrow_{ref}$ Culprit(N)

if $q \notin B$ **do** ;;; Note: $p \succ q$

$Q \leftarrow_{ref}$ DDR-Q-Insert$(q, Q, \succeq)$;;; $\leftarrow_{ref}$ for clarity, DDR-Q-Insert is destructive.

endif

endfor

$KS \leftarrow_{ref} \langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,$ Popped(Q) $\rangle$

Theorem 4.3.3 *(Proof in Appendix, Section A.2.3) Given KS and p and the following*

- *Preconditions for Process-Justout (KS, p): $KS = KS_{pre}$ is a DDR knowledge state; Proper-KS(KS_{pre}); JustifiedOut(p, KS_{pre}); and $\langle p, \tau \rangle =$ First(Q_{pre}), where $\tau \in \{justout, both\}$;*
- *KS_{post} is the knowledge state resulting from Process-Justout (KS, p);*

then the following postconditions hold:

- *Postconditions for Process-Justout* (KS,p)*:* KS_{post} *is a DDR knowledge state; Proper-KS(*KS_{post}*) ;* $B_{post} = B_{pre}$;$B^{\cup}_{post} = B^{\cup}_{pre}$;$I_{post} = I_{pre}$;$\succeq_{post} = \succeq_{pre}$; *QImpact(*KS_{pre}*)*$>$ *QImpact(*KS_{post}*) ;* $KS_{post} \succ_{KS} KS_{pre}$; Q_{post} *resembles* Q_{pre} *with the following changes:*
  - $p \notin Q_{post}$, *and*
  - $(\forall N \in$*NC-NAND-sets*$(p,KS), \exists q =$*Culprit*$(N,\succeq))$: $q \in X \Rightarrow \langle q, in?\rangle \in Q_{post} \vee \langle q, both\rangle \in Q_{post}$

  Note: $(\forall N \in$ *NC-NAND-sets*$(p,KS), \forall b \in (X \setminus \{p\}), \forall \tau)$: $b \neq$*Culprit*$(N,\succeq) \Rightarrow ((\langle b,\tau\rangle \in Q_{pre}) \equiv (\langle b,\tau\rangle \in Q_{post}))$.

The proof is supplied in the Appendix in Section A.2.3.

Update-KS$(KS, update, p, R)$

Update-KS$(KS, update, p, R)$ destructively alters the knowledge state KS (also referred to as KS_{pre}) to be a new knowledge state (KS_{post}) which is like the input KS_{pre} but altered depending on the value of *update*.

procedure Update-KS $(KS, update, p, R)$

$\langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle \leftarrow_{ref} KS$

Case *update* =

PopQ:

$KS \leftarrow_{ref} \langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ \text{Popped}(Q)\ \rangle$

ProcJustout:

Process-Justout(KS, p)

AddRem:

KS-Add-Remove(KS, p, R)

Theorem 4.3.4 *(Proof in Appendix, Section A.2.4) Given KS, update, p, and R and the following*

- *Preconditions for Update-KS(KS, update, p, R): $KS = KS_{pre}$ is a DDR knowledge state;*

  Proper-KS(KS_{pre}); $\langle p, \tau\rangle =$ First(Q_{pre}); update $\in \{PopQ, ProcJustout, AddRem\}$, and

  - *If update $=$ PopQ: JustifiedOut(p, KS_{pre}); and $\tau = in?$.*
  - *If update $=$ ProcJustout: JustifiedOut(p, KS_{pre}); and $\tau \in \{justout, both\}$.*
  - *If update $=$ AddRem: BadOut(p, KS_{pre}); $R \subseteq B_{pre}$; $\tau \in \{in?, both\}$;*

    $\sum_{p_i \in ((B+p)\setminus R)} Cred(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} Cred(p_j, B^{\cup}, \succeq)$;

    $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$; *and*

    $(\forall r \in R) : p \succ r$, *and* $(\exists S \in I) : r = Culprit(S, \succeq) \wedge S \subseteq (((B_{pre} + p) \setminus R) + r)$;

- *KS_{post} is the knowledge state resulting from Update-KS (KS, update, p, R);*

then the following postconditions hold:

- *Postconditions for Update-KS(KS, update, p, R): KS_{post} is a DDR knowledge state;*

  $p \notin Q_{post}$; QImpact(KS_{pre})$>$ QImpact(KS_{post}); $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$;

  Proper-KS(KS_{post}); $KS_{post} \succ_{KS} KS_{pre}$. Also:

  - *If update $=$ PopQ : $B_{post} = B_{pre}$.*
  - *If update $=$ ProcJustout : $B_{post} = B_{pre}$; ($\forall N \in$NC-NAND-sets$(p, KS), \exists q =$Culprit$(N, \succeq)$): $q \in X \Rightarrow (\langle q, in?\rangle \in Q_{post} \vee \langle q, both\rangle \in Q_{post})$; ($\forall N \in$ NC-NAND-sets$(p, KS), \forall b \in X \setminus \{p\}, \forall \tau)$:*

    $b \neq$*Culprit*$(N, \succeq) \Rightarrow ((\langle b, \tau\rangle \in Q_{pre}) \equiv (\langle b, \tau\rangle \in Q_{post}))$.
  - *If update $=$ AddRem : $B_{post} = (B_{pre} \cup \{p\}) \setminus R$; $(\forall r \in R) : p \succ r \wedge$ JustifiedOut(r, KS_{post}); $(\forall r \in R)$: $\langle r, justout\rangle \in Q_{post}$; and $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau\rangle \in Q_{pre}) \equiv (\langle b, \tau\rangle \in Q_{post}))$.*

The proof for Theorem 4.3.4 is in the Appendix in Section A.2.4.

If Update-KS($KS, update, p, R$) alters the base of the knowledge state, the only addition can be the first belief in the queue, and, any retracted beliefs must be a culprit beliefs that are weaker than the belief that was added.

Corollary 4.3.5 *Given* $KS = \langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle$, $\langle p, \tau\rangle$ *= First(Q), and* KS' *is the knowledge base resulting from Update-KS(*$KS, update, p, R$*), then if* $B' \neq B$ *the following is true:*

- $(B' \setminus B) = \{p\}$;
- $(\forall r \in (B \setminus B'), \exists N \in I) : r = \mathit{Culprit}(N, \succeq) \wedge p \succ r.$ *(Thm 4.3.4)*

4.3.4 DDR Algorithm

DDR(KS) takes a DDR knowledge state ($KS = \langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle$, also referred to as KS_{pre}) as input, and improves it incrementally (with each pass throught the loop), until the queue is empty. For the purpose of discussing this algorithm, the starting knowledge state is referred to as KS_{pre} and the final knowledge state is referred to as KS_{post}. Only the base B and the priority queue Q are affected by DDR. $B^{\cup}, I$, and $\succeq$ are *not* affected by DDR.

The DDR algorithm is presented in its entirely on the following page.

```
procedure DDR(KS)
loop until Empty(Q_KS)
    ⟨p,τ⟩ ←ref First(Q_KS)
    if1 (τ = in? or τ = both) , then
        can-return ← BadOut(p,KS)
        if2 can-return , then                                   ;;; BadOut(p,KS)
            R ←ref Safe-Return(p,KS)
            update ← 'AddRem
        else2                                                   ;;; JustifiedOut(p.KS)
            if3 τ = both , then
                update ← 'ProcJustout
            else3                                               ;;; τ = in?
                update ← 'PopQ
            endif3
        endif2
    else1                                                       ;;; τ = justout
        update ← 'ProcJustout
    endif1
    Update-KS(KS,update,p,R)        ;;; DESTRUCTIVE—alters the DDR knowledge state KS
end loop
```

Theorem 4.3.6 *(Proof in Appendix, Section A.3.) Given the DDR knowledge state KS and the following:*

- *Preconditions for DDR(KS): $KS = KS_{pre}$ is a DDR knowledge state; Proper-KS(KS_{pre}).*
- *KS_{post} is the knowledge state resulting from calling DDR(KS) and running it to completion.*

then the following conditions hold:

- *Postconditions for DDR(KS): KS_{post} is a DDR knowledge state; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; $I =$ Complete-I(KS_{pre}) $\equiv$ $I =$ Complete-I(KS_{post}); Proper-KS(KS_{post}); Empty(Q_{post}); $KS_{post} \succeq KS_{pre}$.*
- *Loop conditions for DDR(KS): Let KS_{top} be the DDR knowledge state at the top of the DDR loop (just after line 1). And let KS_{bot} be the DDR knowledge state that results from KS_{top} being processed by the DDR loop (just after line 18). For each pass through the DDR loop: Proper-KS(KS_{top}) and Proper-KS(KS_{bot}); QImpact(KS_{top})$>$ QImpact(KS_{bot}); $KS_{bot} \succ_{KS} KS_{top}$. Additionally, if $\langle p, \tau \rangle$ = First(Q_{top}) and $B_{top} \neq B_{bot}$, then*
  - $(B' \setminus B) = \{p\}$;
  - $(\forall r \in (B \setminus B'), \exists N \in I) : r =$*Culprit*$(N, \succeq) \wedge p \succ r$.

The proof for Theorem 4.3.6 is in the Appendix in Section A.3.

Note that each pass through the loop in the DDR algorithm processes the top element of the dynamically changing priority queue. Each element is processed in turn and how it is processed depends on its tag, τ.

If $\tau = justout$, then the belief p was retracted at some earlier time and the DDR process will insert onto the queue the retracted culprits of its NC-NAND-sets to be considered for possible return to the base. If $\tau = in?$, then DDR determines whether the belief can return to the base and, if so, returns it (with appropriate consistency maintenance). If $\tau = both$, then the belief is first processed as if its tag is *in*?; only if the belief cannot return to the base, will DDR, then, process it as if it has the tag *justout*.

The result of DDR performed to completion is the Optimal-per-I knowledge state.

Theorem 4.3.7 *Given that the DDR knowledge state $KS = KS_{post}$ (containing the elements $B, B^{\cup}, I, \succeq$, and Q) is the knowledge state resulting from DDR(KS_{pre}) running to completion, Proper-KS(KS_{pre}) $\Rightarrow$ Optimal-per-I(KS_{post}).*

Proof.

Proper-KS(KS_{pre}) (premise). KS is the result of DDR(KS_{pre}) running to completion. Therefore, Empty(Q_{post}) and Proper-KS(KS_{post}) (Thm 4.3.6). $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$ (Thm 4.3.6). For any knowledge state KS, Proper-KS(KS): Empty(Q) $\Rightarrow$ Optimal-per-I(KS) (Thm 4.2.54). Therefore, Optimal-per-I(KS_{post}).

□

The result of DDR performed to completion when $I =$ Complete-I(KS) is the Optimal-per-$\perp$ knowledge state.

Theorem 4.3.8 *Given that the DDR knowledge state $KS = KS_{post}$ (containing the elements $B, B^{\cup}, I, \succeq$, and Q) is the knowledge state resulting from DDR(KS_{pre}) running to completion, Proper-KS(KS_{pre}) $\wedge$ $I =$ Complete-I(KS_{pre}) $\Rightarrow$ Optimal-per-$\perp$(KS_{post}).*

Proof.

Proper-KS(KS_{pre}) and $I =$ Complete-I(KS_{pre}) (premise). KS is the result of DDR(KS_{pre}) running to completion. Proper-KS(KS_{pre}) $\Rightarrow$ Optimal-per-I(KS_{post}) (Thm 4.3.7). Therefore, Optimal-per-I(KS_{post}). $I =$ Complete-I(KS_{pre}) $\equiv$ $I =$ Complete-I(KS_{post}) (Thm 4.3.6). Therefore, $I =$ Complete-I(KS_{post}).

$I =$ Complete-I(KS) $\Rightarrow$ (Optimal-per-I(KS) $\equiv$ Optimal-per-$\perp$(KS)) (Obs 4.2.53).

Therefore, Optimal-per-$\perp$(KS_{post}). □

An example illustrating the DDR process begins on the next page.

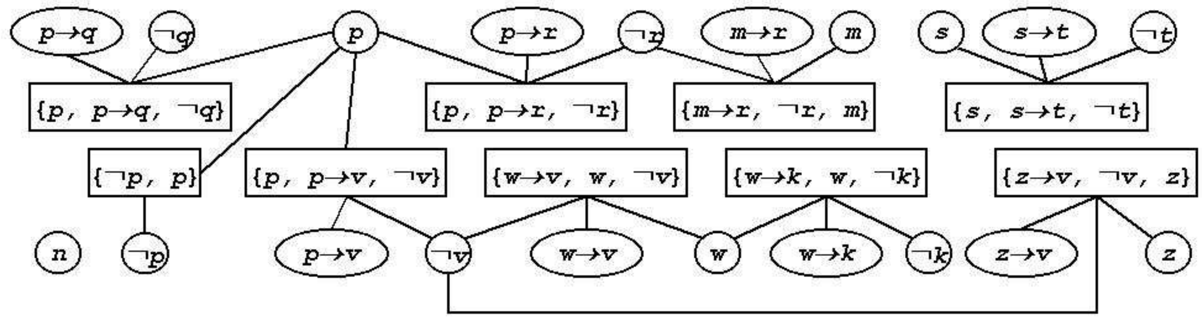

Figure 4.2: A graph showing the elements of $B^{\cup}$ (circles/ovals) of a knowledge state, $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$, connected to their NAND-sets (rectangles), where $B^{\cup} = \neg p, p, p{\rightarrow}q, p{\rightarrow}r, m{\rightarrow}r, s, s{\rightarrow}t, w{\rightarrow}v, w{\rightarrow}k, p{\rightarrow}v, z{\rightarrow}v, n, \neg q, \neg r, w, \neg v, m, z, \neg t, \neg k$ (in decreasing order of preference w.r.t. $\succeq$). In this example, $I =$ Complete-I(KS).

4.4 DDR Example

Example 4.4.1 This example uses the knowledge state KS shown in the nearby Figure 4.2, where $B^{\cup} = \neg p, p, p{\rightarrow}q, p{\rightarrow}r, m{\rightarrow}r, s, s{\rightarrow}t, w{\rightarrow}v, w{\rightarrow}k, p{\rightarrow}v, z{\rightarrow}v, n, \neg q, \neg r, w, \neg v, m, z, \neg t, \neg k$ and $I =$Complete-I(KS). Consider a knowledge state KS_1 where $B^{\cup}_1 = B^{\cup} \setminus \{\neg p\}$, $\succeq_1$ is like $\succeq$ without the preference information for $\neg p$, $I_1 =$Complete-I(KS_1), and Empty(Q_1). The base of KS_1 (the optimal base w.r.t. $B^{\cup}_1$ and $\succeq_1$) is $B_1 = p, p{\rightarrow}q, p{\rightarrow}r, m{\rightarrow}r, s, s{\rightarrow}t, w{\rightarrow}v, w{\rightarrow}k, p{\rightarrow}v, z{\rightarrow}v, n, w, m, z$.[21]

Now consider the optimized addition of $\neg p$ to KS_1 ($KS_1 +_! \langle \neg p, \succeq_{\neg p}\rangle$), with the ordering information $\succeq_{\neg p} = \neg p \succ p$. $KS_1 +_! \langle \neg p, \succeq_{\neg p}\rangle$ forces the retraction of p (to eliminate the contradiction of the NAND-set $\{\neg p, p\}$) to form KS_2 whose base is $B_2 = \neg p, p{\rightarrow}q, p{\rightarrow}r, m{\rightarrow}r, s, s{\rightarrow}t, w{\rightarrow}v, w{\rightarrow}k, p{\rightarrow}v, z{\rightarrow}v, n, w, m, z$. At this point, note that $I =$ Complete-I(KS_2) as mentioned in the caption of the figure.

Most systems stop here (with a sub-optimal base) because they focus on maintaining consistency but do not review previous belief change decisions to see if they could be improved using hindsight. The set of retracted base beliefs (X_2) is $\{p, \neg q, \neg r, \neg v, \neg t, \neg k\}$, $B^{\cup}_2 = B^{\cup}$ (from Figure 4.2), and $\succeq_2 = \succeq$. It is clear

[21] B_1 must be optimal—both Optimal-per-$\perp$ and Optimal-per-I—because Empty(Q_1). Cf. Theorems 4.2.54 and 4.2.55.

that B_2 is sub-optimal, because the removed belief $\neg q$ can be returned to the belief base without raising an inconsistency: $(B_2 + \neg q) \succeq B_2$.

The optimal base w.r.t. $B^{\cup}$ and $\succeq$, would be: $B = \neg p, p \rightarrow q, p \rightarrow r, m \rightarrow r, s, s \rightarrow t, w \rightarrow v, w \rightarrow k, p \rightarrow v, z \rightarrow v, n, \neg q, \neg r, w, z$. DDR can be performed on KS_2 (with the result being $KS_{2\cup!}$) to produce this optimal base. When $KS_1 +_! \langle \neg p, \succeq_{\neg p} \rangle$ produces KS_2, then B_2 is as described above, and Q_2 contains the single pair $\langle p, justout \rangle$. DDR on KS_2 goes through the following steps:

1. Process $\langle p, justout \rangle$. This DDR-Q-Inserts the retracted culprits of p's NC-NAND-sets $(\neg q, \neg r, \neg v)$ into the queue and pops p (and its tag) off the queue. The belief base is unaffected.

   Result: $KS_3 = \langle B_2, B_2^{\cup}, I, \succeq, Q_3 \rangle$, where $Q_3 = \langle \neg q, in? \rangle, \langle \neg r, in? \rangle, \langle \neg v, in? \rangle$

2. Process $\langle \neg q, in? \rangle$. Since $\neg q$ can return to the base without raising an inconsistency (because p is now removed), it is returned to the base and popped off the queue.

   Result: $KS_4 = \langle (B_3 \cup \{\neg q\}), B_2^{\cup}, I, \succeq, Q_4 = \langle \neg r, in? \rangle, \langle \neg v, in? \rangle \rangle$

3. Process $\langle \neg r, in? \rangle$. Because p is now removed, $\neg r$'s IC-NAND-set will not become active if $\neg r$ is returned to the base. The NC-NAND-set for $\neg r$, however, will become active, so its culprit (m) gets placed into R by the function Safe-Return and is removed from the base when $\neg r$ is returned (during the procedure Add-Remove). Note: $\neg r \succ m$, so this is considered a good trade: $B_4 \succ B_3$. The queue is adjusted to reflect the changes: $\neg r$ is popped off the queue, and m is inserted with a tag of *justout*.

   Result: $KS_5 = \langle (B_4 \cup \{\neg r\}) \setminus \{m\}, B_2^{\cup}, I, \succeq, Q_5 \rangle$, where $Q_5 = \langle \neg v, in? \rangle, \langle m, justout \rangle$.

4. Process $\langle \neg v, in? \rangle$. Although the removal of p makes one of $\neg v$'s IC-NAND-sets no longer precarious, the other IC-NAND-set remains precarious, which means that its culprit, $\neg v$, cannot return to the base or it will become active. The only change made to the knowledge state is popping $\neg v$ off the queue.

   Result: $KS_6 = \langle B_5, B_2^{\cup}, I, \succeq, Q_6 = \langle m, justout \rangle \rangle$.

5. Process $\langle m, justout \rangle$. The only IC-NAND-set for m is precarious, so m cannot return to the base; so m

is popped off the queue, the queue is empty, and DDR exits.

Result $KS_7 = \langle B_6, B_2^{\cup}, I, \succeq, Q_7 \rangle$, where Empty($Q_7$) and $B_7 = B$, which is optimal w.r.t. $B^{\cup}$ and $\succeq$. Note: $B^{\cup} = B_7^{\cup}$.

In step 4, once DDR determines that $\neg v$ cannot return to the base (due to its being the culprit for the precarious NAND-set $\{w \rightarrow v, w, \neg v\}$), there is no need to examine $\neg v$'s NC-NAND-set containing z. This shows how NC-NAND-sets for a belief that might return are only examined *after* the examination of its IC-NAND-sets indicates that it truly *can* return to the base. Note, also, that if $\neg v$'s IC-NAND-set containing w happened to be the first IC-NAND-set examined when processing $\neg v$, the examination of $\neg v$'s other IC-NAND-set would have been pruned.[22]

The set containing $\neg k$ would also not be examined or affected, even though it is distantly connected to p. The fact that $\neg k$ never gets looked at is an example of how the NAND-sets to be examined during DDR are determined dynamically—i.e., the examination of *some* beliefs during DDR is dependent on the *result* of DDR's processing of a stronger, connected belief. This saves time and computational expense over first selecting *all* NAND-sets connected to p and, then, examining them in turn.

The NAND-set containing s would also be ignored by DDR, because it is not connected to p in any way. This last case is representative of the possibly thousands of unrelated NAND-sets for a typical belief base which *would* be checked during a naive operation of reconsideration, but are ignored by DDR.

An implemented run of the above example is in the Appendix in Section A.5 on page 289.

[22]I have not developed any criteria for determining an efficient order when processing IC-NAND-sets to see if a belief can return to the base. I am unsure whether one exists; each check for a precarious state has a complexity of $O(1)$ (assuming each NAND-set N keeps a count of the number of its elements in X: 1 means precarious, 0 means active), so any attempt to control the order of examination would probably increase the complexity; and, I currently assume that the number of IC-NAND-sets for any one belief is typically too small to make any such analysis worth the effort. I mention this here, because there may be uses in the future where such an analysis would be beneficial.

4.5 Discussion

4.5.1 Benefits of DDR

In this section, the discussion of knowledge state optimality carries the following caveat:

1. if I =Complete-I(KS), then optimality equates to *true* optimality: Optimal-per-$\perp$ (Def 4.2.52).
2. if I $\neq$Complete-I(KS), then optimality can only refer to Optimal-per-I (Def 4.2.51)
3. the optimality designated by KS-Optimal (defined in Chapter 5: Def 5.5.6 on page 206) is equivalent to Optimal-per-I.

DDR Produces Optimality

If run to completion, DDR(KS) produces an optimal knowledge state. It is Optimal-Per-I (Theorem 4.3.7), and if $I =$ Complete-I(KS), then it is also Optimal (Optimal-per-$\perp$) (Theorem 4.3.8).

DDR is an Anytime Algorithm

The key elements of an anytime algorithm [Dean & Boddy1988] are that (1) the answer is available at any point and (2) the quality of the answer must improve as a function of time.

DDR starts with the *current* knowledge state, KS, which is required to be properly maintained. With each pass through the algorithm loop, the knowledge state for the system gets updated with any changes dictated by the processing of the top element in the queue. If DDR is interrupted, the system knowledge state is the same as it was at the start of that single loop pass — incorporating all the improvements of the previous passes through the DDR loop. Thus, the answer—the most recent answer—is available *at any point* (1). Recall, however, that, although DDR and its helper procedures/function can be interrupted, no individual line of the code is left uncompleted due to an interruption. The line most likely to be implemented in several steps (which *must NOT* be only partially completed) is the last line in the procedure Add-Remove.

Let $KS_{top} \leftarrow KS$ at the top of the DDR loop (just after line 1). Let $KS_{bot} \leftarrow KS$ just after the knowledge state update is performed at line 18. For each pass through the DDR loop: $KS_{bot} \succ KS_{top}$ (Thm 4.3.6). Therefore, (2) is satisfied. Additionally, these incremental improvements towards optimality (whether Optimal-per-I/KS-Optimal or truly optimal/Optimal-per-$\perp$) are both measurable and recognizable (using Cred($B, B^{\cup}, I, \succeq$) and QImpact(KS))—desirable attributes for an anytime algorithm [Zilberstein1996].

DDR offers diminishing returns with time

With each pass through the DDR loop, the changes to the base and exbase involve less and less preferred beliefs. From Thorem 4.3.6, we can see that during any *single* pass through the DDR loop, the queue changes from Q_{top} to Q_{bot}:

- the only belief that can possibly move from the exbase to the base is the most credible belief in Q_{top};
- each belief that moves from the base to the exbase must be a culprit belief that is strictly weaker than the first belief in Q_{top};
- and the first belief on Q_{bot} is strictly weaker than the first belief on Q_{top}.

Thus, each pass through the loop manipulates weaker and weaker beliefs, until the queue is empty.

Any beliefs more credible than the first belief in Q will (for the current $B^{\cup}$ and $\succeq$) remain in their current set (the base or the exbase)—it is the set they will inhabit in the optimal knowledge state (the Optimal-per-I/KS-Optimal knowledge state; and, if $I =$ Complete-I(KS), the Optimal-per-$\perp$ knowledge state).

Diminishing returns is another attribute mentioned in [Zilberstein1996].

DDR is interruptable, preemptable, interleavable, and still optimizing

Whenever DDR is interrupted, the system is left with a knowledge state KS whose Q has been properly maintained. This can be seen from the proofs for DDR algorithm and its helper algorithms—at all times, throughout the lines of the algorithms, the current knowledge state is Proper and therefore it must have a properly maintained queue (Proper-KS(KS) $\Rightarrow$ Proper-Q(KS); Def 4.2.33). Performing DDR to completion on KS will still result in the optimal knowledge state (w.r.t. $B^{\cup}$ and $\succeq$)—again if I =Complete-I(KS), then the resulting knowledge state is *truly* optimal (Optimal-per-$\perp$), otherwise the knowledge state is Optimal-per-I/KS-Optimal. Therefore, it is interruptable and preemptable; features also recommended by [Dean & Boddy1988] and [Zilberstein1996].

If semi-revision is performed on the knowledge state KS after DDR is stopped, we can still call DDR on the new knowledge state (KS_1). This is because the semi-revision process maintains a proper priority queue and a consistent base. DDR(KS_1), if run to completion, will still result in an optimal state (optimal w.r.t. the *current* $B_1^{\cup}$ and ordering, $\succeq_1$ at the time that DDR is re-started).[23] Therefore, DDR is also both interleavable and optimizing—two bonus features that are neither required nor expected from an anytime algorithm.

DDR's dynamic selection of beliefs to process reduces computational expense

As discussed in the example above, the examination of *some* beliefs during DDR is dependent on the *result* of DDR's processing of a stronger, connected belief.

The belief $\neg k$ is not considered for possible return to the base, even though it is distantly connected to p. Only the ordering keeps $\neg k$ from returning to the base. If the order of $\neg v$ and w were switched in the linear ordering so that $\neg v \succ w$, then

- $\neg v$ would return to the base along with the simultaneous removal of the weaker w, which would be placed on the queue with a tag of *justout*;
- processing the retracted w would place the weaker $\neg k$ onto the queue with a tag of *in?*; and
- processing $\neg k$ would result in its return to the base.

[23] And, again, if I =Complete-I(KS), then the resulting knowledge state is *truly* optimal (Optimal-per-$\perp$), otherwise the knowledge state is Optimal-per-I/KS-Optimal.

Recall that Wassermann's anytime algorithm (cf. Section 2.2.6) was a method for selecting the relevant belief set upon which to perform a belief change operation. For example, contraction by p would only affect beliefs "related" (e.g., logically connected) to p—first, this set of beliefs was selected, then the contraction operation was performed on that set. But, that set would *include* $\neg k$ (unless the anytime algorithm was stopped in progress).

Though both Wassermann's algorithm and DDR eliminate the involvement of unconnected or unrelated beliefs (like s), DDR's dynamic selection of beliefs further reduces the set of beliefs that are processed during reconsideration. For systems that do *not* maintain NAND-sets, DDR is not an option. A more naive algorithm for reconsideration would be used and Wassermann's algorithm for focusing on a relevant subset would be very useful.

Using the DDR queue as a measure of confidence

Because of the diminishing returns, the base elements more credible than the top belief in the queue are also elements of the optimal base. Because only culprits of the current $B^{\cup}$ can be retracted from the base (recall that contraction is for consistency maintenance, only), *all* non-culprit base beliefs are in the optimal base. This means that we can use the DDR queue as an indicator of when we are reasoning with a recognizable subset of the optimal base—even if we have yet to attain that optimal base.

Observation 4.5.1 *Given a knowledge state KS, where* $\langle p,\tau\rangle =$ *First(Q), and* $B' \subseteq B$ *s.t.* $(\forall b \in B') : b \succ p \lor b \notin$ *Culprits(KS), then* $(\forall d) : B' \vdash d \;\Rightarrow\; KS\overset{!}{\cup} \vdash d$.

Using DDR to Handle Preference Re-ordering

If the preference ordering of an optimal DDR knowledge state *KS* is altered, the base of that knowledge state may not be optimal w.r.t. the new ordering. DDR can re-optimize a base when the preference ordering is changed.

Given a DDR knowledge state *KS* where Proper-KS(*KS*) and a new ordering $\succeq_1$ for the beliefs in $B^{\cup}$, a new DDR knowledge state $KS_1 = \langle B, B^{\cup}, I, \succeq_1, Q_1 \rangle$ can be formed so that Proper-KS(KS_1). The only change (other than from $\succeq$ to $\succeq_1$) is to the priority queue. Q_1 is similar to Q, except for the following additions: $(\forall q \in X)$: if $\exists N \in I$ s.t. $q =$Culprit($N, \succeq$) and $q \neq$Culprit($N, \succeq_1$), then $q \in Q_1$ with a tag of *in?* or *both* as determined by DDR-Q-Insert.

Theorem 4.5.2 *Let us consider two DDR knowledge states KS and* KS_1*, where Proper-KS(KS),* $B = B_1$*,* $B^{\cup} = B_1^{\cup}$*,* $I = I_1$*,* $\succeq \neq \succeq_1$*, and* Q_1 *is similar to Q with the following changes:* $(\forall q \in X)$: *if* $\exists N \in I$ *s.t.* $q =$*Culprit(*$N, \succeq$*) and* $q \neq$*Culprit(*$N, \succeq_1$*), then* $q \in Q_1$ *with a tag of* in? *or* both *as determined by DDR-Q-Insert. Note:* $\forall q, \forall \tau : \neg(q = \mathit{Culprit}(N, \succeq) \wedge q \neq \mathit{Culprit}(N, \succeq_1)) \Rightarrow \langle q, \tau \rangle \in Q \equiv \langle q, \tau \rangle \in Q_1$*. Based on these given assumptions, Proper-(*KS_1*).*

Proof.

Because Proper-KS(*KS*), any belief that was designated as BadOut in *KS* was Protected-by-Q in *KS*. Any belief that was *already* designated as BadOut remains Protected-by-Q, because either

1. it retains a tag of *in*? or *both* (it satisfied lines $Q1$ or $Q2$),
2. it remains the culprit of some NAND-set whose more credible retracted beliefs are tagged with *justout* or *both*, or
3. is is bumped out of the culprit position of the NAND-set just described, and it is DDR-Q-Inserted into the priority queue, which protects it.

Any removed belief that *becomes* designated as BadOut due to the re-ordering[24] is in Q_1 with a tag of either *in*? or *both*, which allows it to be considered for possible return to the base. Therefore, Proper-Q(KS_1).

[24]To *become* BadOut, it must have been Justified-Out before the re-ordering. If it remains the culprit of a NAND-set designating it as JustifiedOut, then it is still JustifiedOut. Therefore, to now be BadOut, the new ordering must have moved it *out* of the culprit position for at least one of its NAND-sets. In which case, it will have been DDR-Q-Inserted into the priority queue, which protects it.

Since Proper-KS(KS), $(\forall S \in I) : \neg\text{Active}(S, KS)$. Since $B_1 = B$ and $I_1 = I$, we also know that $(\forall S \in I_1) : \neg\text{Active}(S, KS_1)$. Thus, Safe-per-I($KS_1$) Therefore, Proper-KS($KS_1$), and DDR running to completion will optimize the base w.r.t. $B^{\cup}$ and the new preference ordering. □

If the analysis to determine what beliefs have shifted out of a culprit spot proves too computationally expensive, a simpler algorithm is to DDR-Q-Insert *all* elements of X into the queue. This might be more appropriate if the re-ordering resembles a re-shuffling of all the beliefs in the ordering as opposed to just a few beliefs sliding up or down in the order.

Either adjustment for dealing with a re-ordering saves steps over consolidation of $B^{\cup}$ using $\succeq_1$—including consolidation by reviewing of all NAND-sets in order (using a NAND-set-Q) to determine which culprits should be removed—by not looking at sections of the base that have been unaltered by the re-ordering.[25] Only a reordering *within* an inconsistent set and affecting a removed culprit could possibly effect a change in the base. But that change could then result in other adjustments to the base. DDR will also catch any changes that are perpetuated by these initial changes in the same way that it dealt with cascading changes in Example 4.4.1—when the removal of p was later followed by the assertion of $\neg r$ with the simultaneous retraction of m.

4.5.2 DDR Optimized-Addition Differs from Other Two-step Belief-change Algorithms

DDR optimized-addition is a two-step belief change operation much like those discussed in Section 3.1.4: saturated kernel contraction [Hansson1994], safe contraction [Alchourrón & Makinson1985], standard adjustment [Williams1994a], and hybrid adjustment [Williams & Sims2000]. DDR optimized-addition returns beliefs to the base (possibly exchanging them with less credible beliefs that become retracted) after first ex-

[25] This is because $| I | \geq |\text{Culprits}(KS_1)| \geq | X |$.

panding the base by adding some belief. However, it also differs from those operations in several key ways. The differences are

- the beliefs that return to the base start out in the exbase at the start of the DDR optimized-addition operation, whereas the two-step operations mentioned in Section 3.1.4 are "returning" beliefs that start out in the base—they are only "removed" from the base during the first step of the belief change operation;
- the second step of DDR-optimized addition *is* DDR, which can be interrupted—although that stops the DDR-optimized addition process, DDR can be recalled later to finish optimizing the knowledge state;
- if run to completion, DDR-optimized-addition will produce the Optimal-per-I knowledge state w.r.t. $B^{\cup}$ and $\succeq$; and, if I =Complete-I(KS), the the knowledge state would also be Optimal-per-$\perp$.

4.5.3 Issues Regarding Implementing DDR in Existing KR&R Systems

Adding DDR to an Existing TMS

Making an existing truth maintenance system[26] DDR-capable involves very little additional computational load. The reasoning system is unaffected. The contradiction handling system must add a few steps: maintaining the priority queue and the detected NAND-sets with links to and from their elements.

I assume the TMS already has some technique for determining the culprit in each NAND-set; I merely require that the technique is consistent (resulting in a partial ordering over the culprits—no cycles in the ordering). Any arbitrary linear ordering over $B^{\cup}$ that is consistent with this pre-established partial ordering will work with the DDR algorithm and will produce the same final and optimal results (though the intermediate changes to the base may differ based on which linear ordering is chosen).

[26] By truth maintenance system, I am referring to systems that already compute NAND-sets: such as JTMS, ATMS and LTMS as discussed in [Forbus & de Kleer1993] and the system described in [Martins & Shapiro1988]. Cf. Section 2.4.

Recall that DDR can be started, stopped, and continued at any time. If DDR is performed only during "down times" (times when reasoning and belief change are not being performed) there will be no perceivable effect (no slowing of the reasoning process) during regular, daily operation; the only difference to the user is that of reasoning with a more credible base.

If it is determined that DDR *must* be performed prior to executing any further reasoning or belief change operations, then this decision implies that the cost of DDR is obviously preferred over that of reasoning with a less credible base, and DDR is computationally less expensive than a naive batch consolidation over *all* base beliefs in an effort to determine *the* optimal base.

Reconsideration for Non-TMS Implementations

Implementing DDR in a system that does not already detect NAND-sets might be computationally expensive. Such a system does *not* compute NAND-sets for some reason that may include the resulting computational load; thus, adding NAND-set detection and maintenance might be disadvantageous. These systems may operate with more restricted domains or logics, however, where a naive implementation of reconsideration would be helpful (provided the computational load is not prohibitive).[27]

4.5.4 DDR as an Alternative to Iterated Belief Revision

DDR offers a solution (at least for TMS systems) to many of the issues explored by researchers in *iterated belief revision* (iterated BR)—issues regarding belief change operations performed in series [Williams1994b, Darwiche & Pearl1997, Chopra, Ghose, & Meyer2002]. Research in the area of iterated BR focuses mainly on two different areas. I mention both areas below, but the focus of this section is on the second of the two.

The first iterated BR research area deals with the need for the decision function used by belief revision

[27] See SATEN examples in Section 3.5.

operations to be a global function. Belief change theories [Alchourrón, Gärdenfors, & Makinson1985] initially focused on belief change operations that were being performed on a deductively closed set of beliefs using a decision function specific to *that* set. Once the set was altered (through retraction or addition), the decision function no longer was applicable to the new set. One focus of iterated BR research is how to define global decision functions that can make BR decisions for various knowledge states. An example of a global decision function is the optimizing restriction for my DDR KS consolidation operation: to maximize the credibility of the knowledge state belief base w.r.t. a linear ordering of those base beliefs.[28] Defining a global decision function is essential to being able to address the second issue dealt with by iterated BR.

The second area of iterated BR research is an attempt to define and improve the results of a *series* of belief change operations in order to reduce the negative side-effects of operation order [Darwiche & Pearl1997, Chopra, Ghose, & Meyer2002]. This is the same goal for which DDR was developed, and thus this area is the focus of this sub-section. Typically, iterated BR has approached this issue by defining the effect that a specific series of operations should have on a set of beliefs. Some complications associated with this approach are:

- Where do you stop in the determination of a series? After two operations? Three?
- What is done in the meantime—handle the first of a series then adjust when the second comes along and you recognize the series' pattern? What preceded the first operation? What if patterns overlap? What if a pattern is interupted by another pattern then continued at a later time?
- Must effects be defined for all possible combinations of operations?

DDR handles all these complications. DDR allows operations to be performed individually (as a series of operations), but it also allows the repair that hindsight offers. Additionally, DDR can optimize following

[28]This is merely one example of a global decision function, but any such function that can produce a linear ordering on the culprits of $B^{\cup}$ can be utilized by the DDR algorithm, provided that ordering is independent of the makeup of the base of the knowledge state.

a series of operations interrupted in the middle by another series—something that iterated BR has yet to tackle.

One example of the DDR operations adhering to an iterated BR axiom is optimized addition performed in series adhering to axiom (R3) in [Chopra, Ghose, & Meyer2002] as discussed in Section 3.3.3—the discussion was in the chapter on reconsideration, but applies equally to DDR (a specific algorithm for performing reconsideration). In DDR terms, (R3) states that $\neg p \notin Cn(KS) \Rightarrow Cn(KS) \subseteq Cn((KS +_{\overset{!}{\cup}} \neg p) +_{\overset{!}{\cup}} p)$, where recency determines order (i.e., $p \succ \neg p$, because p was added most recently). As discussed in Chapter 3, however, DDR not only recovers the original belief *space* after the two optimized additions, but the original belief *base* would also be contained in the *final base*.

Another example is (C3) in [Darwiche & Pearl1997] which can be restated to say: $q \in Cn(KS +_{\overset{!}{\cup}} p) \Rightarrow q \in Cn((KS +_{\overset{!}{\cup}} q) +_{\overset{!}{\cup}} p)$, again, assuming recency ordering.

Note that iterated BR assumes recency ordering of beliefs (i.e., the more recent the belief, the higher its preference). DDR allows orderings that differ from recency to produce alternate results that are hard to imagine in an iterated BR form.

4.5.5 DDR as a CSP

The optimization of a DDR knowledge state can be framed as a boolean Constraint Satisfaction Problem (CSP) to obtain a knowledge base $KS = \langle B, B^{\cup}, I, \succeq, Q \rangle$ [Russell & Norvig2003]. The beliefs in $B^{\cup}$ are the *variables* of the CSP. Their *domain* of possible values consists of: *true* (in the base) and *false* (in the exbase). Therefore, this is a boolean *CSP*, because the domain contains only two possible “values”.

The NAND-sets represent *multiary hard constraints*—multiary, because they are multi-variable constraints (constraints affecting more than one variable and of varying sizes); hard, because they *must* be adhered to.

Figure 4.2 is a *constraint hyper-graph*, where the boxes represent constraints and circles/ovals represent variables. The lines connect the constraints to their variables. e.g., NAND-set $\{\neg p, p\}$ is the constraint $\neg(\neg p \wedge p)$. Note: These constraints can yield multiple bases that are consistent, but not necessarily optimal.

There are two ways to optimize the result:

(1) Require Cred$(B, B^{\cup}, I, \succeq)$ be implemented as an *objective function*, which must be maximized.

(2) Add a *hierarchy* of *unary constraints*: $(\forall p \in B^{\cup}) : p = true$, where the ordering of these constraints is identical to the ordering of the beliefs in $B^{\cup}$—$(\forall p_i, p_j \in B^{\cup}) : p_i = true \succ p_j = true$ iff $p_i \succ p_j$.

When new beliefs are added to the knowledge state, we may derive new NAND-sets. In this case, the number and type of constraints may always be increasing. This is called an *iterated* CSP.

4.6 In Summary

DDR is a TMS-friendly algorithm, which performs reconsideration by examining only a dynamically determined subset of base beliefs that are actually affected by changes made since the last base optimization process. DDR is an efficient, anytime, belief base optimizing algorithm that eliminates operation order effects. If the reasoner is ideal—i.e., it stores all the minimal NAND-sets in $B^{\cup}$—then the result of running DDR to completion is a knowledge state that is Optimal (a.k.a. Optimal-per-$\perp$). Chapter 5 contains a formalization of a non-ideal system and how that system can use DDR to become optimal w.r.t. its limited knowledge. Chapter 6 presents the extended case study of such an implementation of DDR in an existing TMS system.

Chapter 5

A Deductively Open Belief Space

This chapter contains material that was originally published in [Johnson & Shapiro2001]. It has since been expanded and altered for clarity and to include concepts for applying the DDR algorithms to a Deductively Open Belief Space.

5.1 Introduction

Implemented knowledge representation and reasoning, KR&R, systems run into danger when they attempt to implement traditional belief change theories. These theories are defined in terms that assume (or require) that the reasoning agent (or system) can perform ideal reasoning—i.e., it can reason completely and instantaneously [Alchourrón, Gärdenfors, & Makinson1985, Nebel1989, Hansson1993, Hansson1999]. Resource limitations may cause a real world system that guarantees consistency to fail: logic complexities and/or memory or time restrictions may prevent complete consistency checking. This is especially true of systems with a large knowledge base and a complex logic and reasoning system. Due to this size and complexity, implemented systems cannot guarantee deductive closure, completeness, or decidability in a resource-limited world. These limitations prevent real world systems from being able to guarantee that they accurately implement all of the belief change theories that have been developed assuming the reasoning agent is ideal.

One example of this kind of limitation occurs when a new belief is added to the belief space for the purpose of reasoning with that belief. For a system with a large base and complex logic, there must be a trade-off between using this new information and adhering to the belief change concepts discussed in Chapter 2. Either

- reasoning with the new belief must be delayed until the base is checked and adjusted for consistency (and it may be that such a check is not guaranteed to be complete, though it must be sound[1]),

  or

- the system must reason with a possibly inconsistent base.

In Chapter 4, I defined optimality for an ideal agent—one that reason's completely, where $I =$ Complete-I(KS), and DDR run to completion guarantees that the resulting knowledge state is Optimal-per-$\perp$. I also showed that, for a non-ideal agent, I cannot be guaranteed to be complete, so DDR run to completion can only guarantee that the resulting knowledge state is Optimal-per-I. In other words, it is possible that, even if Optimal-per-I(KS), KS may be inconsistent if $(\exists S \in \text{Complete-I}(KS) \setminus I) : S \subseteq B$.

If this is the case, one must consider what happens when further reasoning from the current base exposes a previously undetected inconsistency (i.e., discovers some $S_{\subseteq B} \vdash \perp$ that was previously undetected when reasoning was performed). DDR can also optimize a deductively open belief space after new inconsistencies are discovered—making it optimal w.r.t. the current ordering ($\succeq$) and whatever inconsistencies are *now* currently known. This is discussed in Section 5.5.2.

Another question arises from this discussion, however. A knowledge state KS can be inconsistent even though it is Optimal-per-I. There is no term for this in the literature, though it can be referred to as "consistent as far as is known." Optimal-per-I might reflect the concept of "optimal as far as is known." The purpose

[1] Checking for consistency is *sound* if every inconsistency found is truly an inconsistency; it is *complete* if *all* inconsistencies are found.

of this chapter is to provide the terminology that bridges the gap between consistency and inconsistency for implemented systems that cannot guarantee complete reasoning.

In the following sections, I present a knowledge state formalization that incorporates the facts that

1. its belief space is not guaranteed to be deductively closed,
2. it does not guarantee that it can determine if a belief is derivable (it is not complete),
3. it cannot guarantee to know all ways that a belief can be derived,
4. and, therefore, it cannot guarantee to know all minimally inconsistent sets of its base beliefs.

This knowledge state defines a deductively *open* belief space (DOBS) that, at any given time, consists of its currently asserted *explicit* beliefs. These beliefs are comprised of the set of base beliefs that are currently asserted (B) and only those derivations that have been performed *up to that point* that follow from B. Even when the belief *base* is not growing (i.e., no new input is being added), the belief *space* can grow as more beliefs are derived. I use this formalization to redefine some of the belief change terminology that is discussed in Chapter 2 so that it can apply to a DOBS.

Then, Section 5.5 discusses how the DDR algorithms presented in chapter 4 can also be used to optimize a DOBS (making it optimal w.r.t. the inconsistent sets it is aware of).

5.1.1 A Deductively Open Belief Space (DOBS)

A DOBS is a belief space that is by definition not guaranteed to be deductively closed. It contains (1) the currently asserted set of core beliefs (the belief base B) and (2) the beliefs that *have been* derived from them *so far*. For example, a DOBS might include the base beliefs p and $p \rightarrow q \wedge s$ along with the derived belief s but without the derivable (but not *yet* derived) belief q. There is a marked difference between the concept of a DOBS and that of a belief base [Nebel1989, Hansson1999]. A belief base B is a finite set of beliefs that represents a larger set of beliefs K called its belief space which is its *deductive closure*: $Cn(B) = K$. The DOBS *is* its belief space of *explicit* beliefs only and can grow through additional derivations *without expanding its base beliefs*.

However, the DOBS for some belief base B *does* form a finite set of beliefs whose deductive closure is the same as that of B — thus, technically, both the DOBS and B are belief bases for the same deductively closed belief space. The concept for a belief base that I use, however, refers to some *core* set of beliefs which are asserted with independent standing (as opposed to *any* finite set of beliefs whose closure is a predetermined belief space). See [Hansson1999] or my discussion in Chapter 2 for a more complete discussion of base beliefs.

5.1.2 The Need for a DOBS Formalization

Most belief change approaches use the concept of retaining (or returning) consistency to define their belief revision operations. A belief *base* is considered inconsistent if an inconsistency exists in its *deductive closure*. Whether the inconsistency is found by deductive closure or some procedure, such as resolution refutation, it still requires looking past what is known explicitly into the implicit beliefs to find an inconsistency. This requires time and space as well as a logic and reasoning system that guarantee completeness. Even a system using classical propositional logic could fail if the process requires more memory/time than the system/user has available.

I address the question of how to implement ideal techniques in a non-ideal system. I do this by formalizing theories that take into account the fact that deductive closure cannot be guaranteed in a real-world, need-based, resource-bounded, implemented system. These theories need to define a belief maintenance system that:

1. is not dependent on deductive closure

2. takes time and computational limitations into account

    - recognizing that these limitations might result in revision choices that are poor in hindsight

3. catches and corrects these sub-optimal choices as efficiently as possible.

5.2 Formalizing a DOBS

Because a DOBS cannot guarantee completeness, it cannot guarantee that it is consistent. This prevents a DOBS from being able to satisfy the guidelines described in Section 2.2, which center on the concept of guaranteeing consistency.

Once I present my formalization for a DOBS, I can redefine the terminology used in the basic constraints and postulates for belief change so that the revised versions can be adhered to by a DOBS. This results in guidelines that real world implementations can achieve, and by which real world systems can be compared.

5.2.1 Defining a DOBS

A deductively open belief space consists of a set of base beliefs (*B*) and the beliefs that are known to be derived from them (*D*). These beliefs are derived gradually over time, so *D* can increase monotonically even as *B* remains unchanged. Derived beliefs are typically generated as a result of some query process or inference, such as:

- a query from a user to the system (e.g., “Does Tweety fly?”)
- a query generated by the system during backward inference (e.g., a user asks “Does Tweety fly?”, and the system—knowing that birds fly—asks itself, “Is Tweety a bird?”... if so, then Tweety flies)
- the system performing forward inference (e.g., the system is told “Tweety is a bird” with a request to perform forward inference, so it reasons that “Tweety flies”, because it knows that birds fly).

I am currently defining a DOBS system as one that stores beliefs it has derived rather than continually re-deriving them when needed. This assumes that the cost to re-derive a belief is higher (e.g., takes longer) than the cost to access the stored derivation information from memory. I discuss an alternative approach later in Section 5.4. If the system stores derivations, obviously, it can only store derivations that it has *actually performed* (or inferred)—not *all* possible derivations. For example: given $B = \{p, p \rightarrow q, r, r \rightarrow q\}$ and $D = \{q\}$ (derived from p and $p \rightarrow q$), the system is currently unaware that r and $r \rightarrow q$ also imply q.

5.2.2 The Knowledge State That Determines a DOBS

The information required by a DOBS system is represented by a knowledge state, *KS*. The DOBS *is* the belief space of the knowledge base, $BS(KS)$. Given the language $\mathcal{L}$ as described in Section 1.2, I define a DOBS knowledge state as:

$KS =_{def} \langle B, B^{\cup}, D^{\cup}, H \rangle$,

where $B^{\cup} \subseteq \mathcal{L}, D^{\cup} \subseteq Cn(B^{\cup}), B \subseteq B^{\cup}$, and H contains a record of the derivations performed to derive the propositions in $D^{\cup}$.

Unless otherwise noted, assume that all future examples and definitions of belief change operations are using $KS = \langle B, B^{\cup}, D^{\cup}, H \rangle$ as their starting belief state.

$B^{\cup}$ is identical to the $B^{\cup}$ discussed in Chapter 4. It consists of all the hypotheses (also called assumptions) ever introduced into the knowledge state as self-supporting beliefs. For further discussion of what kinds of beliefs can be self-supporting, see [Hansson1999]. $B^{\cup}$ contains both currently believed hypotheses (the belief base of the DOBS) as well as those that have been retracted from the belief space.

$D^{\cup}$ consists of every proposition p ever derived from some set $A \subseteq B^{\cup} \setminus \{p\}$ using one or more inference steps. Whenever a proposition is derived, it becomes a member of $D^{\cup}$ and a record of its derivation is stored in H: $\langle p, A \rangle \in H$ means that p was derived from the set A. A proposition can exist in both $B^{\cup}$ and $D^{\cup}$.

B consists of all propositions that are currently asserted as being self-supporting beliefs (i.e., the current base beliefs). Therefore, $B^{\cup} \setminus B$ contains propositions that are no longer believed to be self-supporting, although some of these may be in the current belief space as *derived* beliefs *if* they have derivation histories in H showing that they can be derived from B.

H consists of the derivation histories for all elements of $D^{\cup}$. There are multiple ways in which this could be implemented (e.g., ATMS, JTMS, derivation tree), but I will formalize it as $\langle p,A\rangle \in H$ is the implementation independent record of a derivation of p from the set A, where $p \notin A$—although a belief can (through identity) be derived from itself, that information is excluded from H, and I assume that the system does not bother to make that kind of derivation. Every proposition d in $D^{\cup}$ must have at least one derivation history stored in H indicating it is derived from some $B' \subseteq (B^{\cup} \setminus \{d\})$. A proposition can have more than one derivation history.

D is a subset of $D^{\cup}$ that need not be maintained in the DOBS system knowledge state, because it can be determined using B and H. D is the set of beliefs that the system knows can be derived from the base beliefs in B according to the information stored in H.

The key parts of a DOBS knowledge state are illustrated in Figure 5.1 and described below. The fact that $B \cup D$ is the belief space (BS) is proved later in this section. $Cn(B)$ is included for clarity, but it is not an actual part of a knowledge state, and the knowledge state has no information regarding which elements are in $Cn(B)$ outside of $BS = B \cup D$.

L all well-formed formulas (WFFS) in the language

B$^{\cup}$ asserted propositions with independent standing

—both believed and disbelieved

B the currently believed hypotheses

Cn(B) the complete deductive closure of **B**

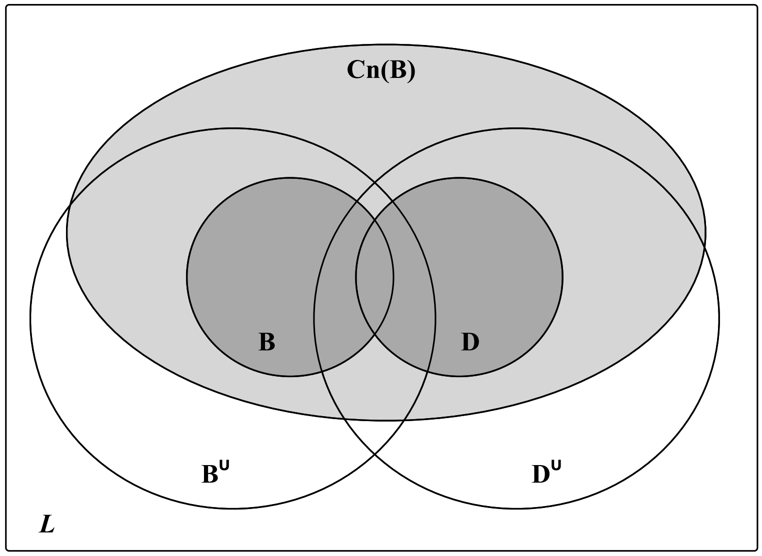

Figure 5.1: This figure shows the different designations for beliefs stored by a system that reasons with a Deductively Open Belief Space (DOBS). The lightly shaded area that forms $Cn(B)$ is included for clarity and comparison purposes—its information is *not* stored by a DOBS system. The actual belief space (BS) is the darkly shaded area formed by $B \cup D$.

—potentially infinite (depending on the logic used)

—included here for conceptualization (not part of *KS*)

D$^\cup$ beliefs known to be derived from **B**$^\cup$

D beliefs known to be derived from **B**

BS the current belief space, THE DOBS: **B∪D** (darkly shaded)

— a finite subset of **Cn(B)**

Note: **B∪D = BS = the DOBS**

DOBS – Coherence Version

I distinguish the propositions in $B^\cup$ from those in $D^\cup$ to allow a foundations approach for belief maintenance. Recall that a foundations approach says that derived beliefs that lose their support are no longer believed. A coherence approach can be implemented, however, by inserting each derived belief into $B^{\cup}$ as a self-supporting belief. B should also include all beliefs known to be derivable from B (determined by the derivation histories in H). In this sense, B *is* the belief space (the DOBS). Cf. Sec 2.2.1 for a discussion of foundations vs. coherence theories.

Inserting derived beliefs into B allows a belief that loses its support to remain in the belief space unless it needs to be removed to eliminate an inconsistency. Note, however, that the derivation histories of any derived beliefs must still be retained in H to aid in belief change operations, because they are the only record of how to retract a derivable belief from a DOBS—not only must the belief be removed from B, but some of its supporting beliefs must be removed as well, or it would be re-derived.

5.2.3 DOBS Terminology

KS-derivability

Since a DOBS can have propositions that are derivable but not, yet, derived, I use the concept of a proposition p being *known to be derivable* from a set of propositions A. This is denoted as $A \vdash_{KS} p$, read as "A KS-derives p", and is defined by the rules below:

1. $\{p\} \vdash_{KS} p$.

2. If there exists some $\langle p, A \rangle \in H$, then $A \vdash_{KS} p$.

3. $A \vdash_{KS} B$ means that $(\forall p) : p \in B \Rightarrow A \vdash_{KS} p$.

4. A superset of a set that KS-derives a proposition also KS-derives that proposition:

   $(A \vdash_{KS} p) \wedge (A \subsetneq B) \Rightarrow B \vdash_{KS} p$.

5. $(A \vdash_{KS} B) \wedge (B \vdash_{KS} p) \Rightarrow A \vdash_{KS} p$.

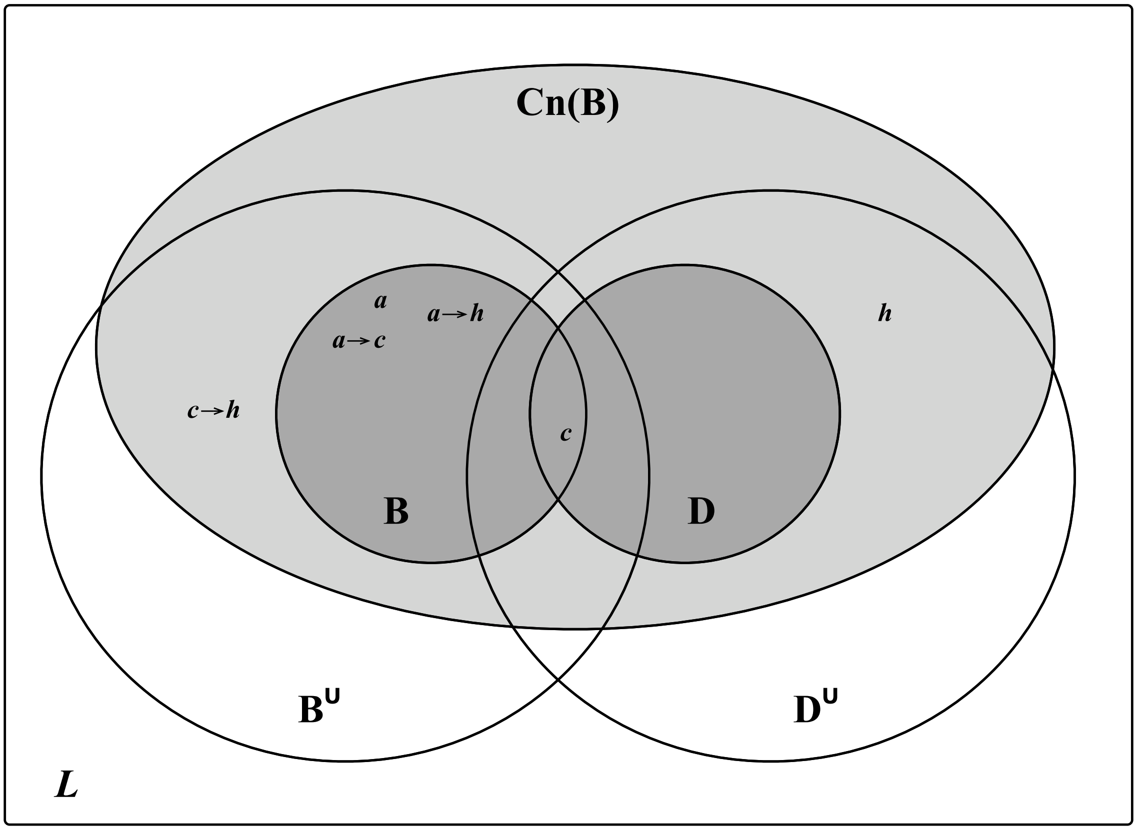

Figure 5.2: Expanding on the diagram in Figure 5.1, this figure shows an example of a DOBS where $B^{\cup} = \{a,\ a \rightarrow h,\ a \rightarrow c,\ c,\ c \rightarrow h\}$, $B = B^{\cup} \setminus \{c \rightarrow h\}$, $D^{\cup} = \{c,\ h\}$, and H contains the following derivations: $\langle c,\ \{a,\ a \rightarrow c\}\rangle$, $\langle h,\ \{c,\ c \rightarrow h\}\rangle$, and $\langle h,\ \{a,\ a \rightarrow c,\ c \rightarrow h\}\rangle$. The system has derived h using $c \rightarrow h$, which is no longer in the base—this places h within $D^{\cup}$. The system is not *yet* aware that h is derivable from the current base using the beliefs a and $a \rightarrow h$—this places h outside D. The belief c is both a base belief and derivable from other beliefs in the current base—this places c within $B \cap D$.

An example of a knowledge state and its DOBS are illustrated in Figure 5.2—an expansion of the diagram shown in Fig 5.1. The proposition h is not currently in BS, because, although it is derivable from B, it is only KS-derivable using the retracted belief $c \rightarrow h$. This is why the system places it in $D^{\cup}$, but *not* in D. Because this is a foundations approach representation (as opposed to coherence), the fact that h is not in $B^{\cup}$ means that it has never been designated as a base belief. Section 5.2.3 contains a more detailed explanation with more examples.

Recall that the elements of $Cn(B)$ are mostly unknown to the system. The system only knows that $(B \cup D) \subsetneq Cn(B)$—no matter what is in $B \cup D$, $Cn(B)$ must be larger, because it contains an infinite number of beliefs. The system knows $c \rightarrow h \in B^{\cup}$ and $h \in D^{\cup}$. The placement of $c \rightarrow h$ and h within $Cn(B)$ is for clarity and is not something that is known from the knowledge state itself.

For any $p \notin B \cup D$, the system has not yet determined whether $p \in Cn(B)$—unless the system somehow determined that $B \nvdash p$. This is typically computationally expensive, and deals with completeness issues; so, for the purpose of this chapter, I will assume that a system cannot *guarantee* that a belief p is *not* derivable from a given base, merely that it has not *yet* been so derived. Many other beliefs could be placed in $Cn(B)$, but none that would be represented in the DOBS tuple.

KS-closure

Because I am removing the omniscient aspect of a DCBS and its consequence operation, I want the DOBS to "remember" as much as possible, including propositions that are no longer believed. Once a base set of hypotheses, B, is chosen, the implementable closure of B is limited by KS (i.e., by its derivation histories in H). I call this *KS-closure*, and its consequence operator is C_{KS}. Its definition is: $C_{KS}(B) =_{def} \{p | B \vdash_{KS} p\}$.

The belief space of a DOBS is defined by its belief state, KS, as the KS-closure of its base beliefs (B):

$BS(KS) =_{def} C_{KS}(B)$ = *the* DOBS

$D(KS)$ is the set of *derived* propositions that are currently believed:

$$D(KS) =_{def} \{p | B \setminus \{p\} \vdash_{KS} p\}$$

In other words, KS represents all the propositions that exist in the system along with a record of how they were derived, and $BS(KS)$ represents only those propositions that are currently believed — therefore, it *is* the DOBS. The DOBS must keep track of the disbelieved propositions and derivations to avoid having to repeat earlier derivations if disbelieved propositions are returned to the belief space (assuming that memory space is a more plentiful resource than computing speed/time).

For shorthand purposes, $BS(KS)$ and $D(KS)$ can be written as BS and D respectively when their KS is clear from the context. The information that $p \in B^{\cup} \cup D^{\cup}$, can be written in a shorthand version as $p \in KS$. This is not to be confused with $p \in BS$, though the latter implies the former.

Observation: $BS = B \cup D$

Proof. By the definitions above, $B \vdash_{KS} A \Rightarrow A \subseteq BS$.

a) Prove $B \cup D \subseteq BS$. $(\forall p) : p \in B \Rightarrow B \vdash_{KS} p$. Therefore, $B \subseteq BS$. $(\forall p) : p \in D \Rightarrow B \backslash \{p\} \vdash_{KS} p$. And, of course $(\forall p) : B \backslash \{p\} \subseteq B$. Therefore, $(\forall p) : p \in D \Rightarrow B \vdash_{KS} p$. Thus, $D \subseteq BS$, and $B \cup D \subseteq BS$.

b) Prove $BS \subseteq B \cup D$. $(\forall p) : p \in BS \Rightarrow (B \vdash_{KS} p)$. $(\forall p) : (p \in B) \vee (p \notin B)$. Therefore, $(\forall p) : p \in BS \Rightarrow (B \vdash_{KS} p) \wedge ((p \in B) \vee (p \notin B))$. Therefore, $(\forall p) : p \in BS \Rightarrow ((B \vdash_{KS} p) \wedge (p \in B)) \vee (B \vdash_{KS} p) \wedge (p \notin B))$. If $p \notin B$, then $B \vdash_{KS} p \Leftrightarrow B \backslash \{p\} \vdash_{KS} p$. Therefore, if $p \notin B$, then $B \vdash_{KS} p \Leftrightarrow p \in D$; and $(\forall p) : p \in BS \Rightarrow ((B \vdash_{KS} p) \wedge (p \in B)) \vee (B \vdash_{KS} p) \wedge (p \in D))$. Therefore, $(\forall p) : p \in BS \Rightarrow (p \in B) \vee (p \in D)$. Thus, $(\forall p) : p \in BS \Rightarrow p \in B \cup D$. $\square$

KS-consistency

Any set is inconsistent if a contradiction has been or *can be derived* from its beliefs. Thus, checking for an inconsistency requires examining implicit beliefs. This is time consuming for a DOBS, which can never guarantee a *complete* exploration of its implicit beliefs.

A DOBS knowledge state (and its base, as well as the DOBS itself) is *KS-inconsistent* if and only if it is *known* to derive an inconsistency.

Definition 5.2.1 KS-inconsistent(KS) $\equiv B \vdash_{KS} \perp$.

If a DOBS is not KS-inconsistent, then it is called *KS-consistent* — i.e., there are no explicit inconsistencies in $C_{KS}(B)$, so it is *not known to be inconsistent*.

Definition 5.2.2 KS-consistent(KS) $\equiv B \nvdash_{KS} \perp$.

This means a DOBS can be both inconsistent and KS-consistent at the same time: For example, $B = \{q, p, p \rightarrow \neg q\}$, but $\neg q$ has not, yet, been derived. Note that you can also refer any set, A, as KS-consistent or KS-inconsistent as long as there is a DOBS knowledge state associated with that set from which you can determine the KS-closure of the set ($C_{KS}(A)$).

KS-consolidation

Whenever an inconsistency must be resolved in a DOBS, some beliefs must be removed. Which beliefs are removed is often determined by examining the entire belief space and using some guidelines (such as integrity constraints 3 and 4 from Section 2.2.2). A DOBS, however, is incomplete, because of the lack of deductive closure. It is possible that newly derived beliefs would add information that might have altered the belief contraction choices made in an earlier operation.

The DOBS version of Hansson's kernel consolidation[2] operation is called *KS-consolidation*, and it returns a belief base that is KS-consistent and follows the following postulates (for a belief base, B that is consolidated to produce $B!_{KS}$):

KS-C1 $B!_{KS}$ is KS-consistent *KS-consistency*

KS-C2 $B!_{KS} \subseteq B$ *inclusion*

KS-C3 If $p \in B \backslash B!_{KS}$, then there is some B' such that $B!_{KS} \subseteq B' \subseteq B$, $C_{KS}(B')$ is KS-consistent, and $C_{KS}(B' + p)$ is KS-inconsistent *Core-retainment*

[2]Cf. Section 2.2.5 and/or [Hansson1997].

In Section 5.5, I discuss how the DDR algorithms can be used to perform reconsideration on a DOBS knowledge state to maintain the most preferred state (per the current information stored in the knowledge state) without having to preform a full consolidation over all of $B^{\cup}$—assuming we have a linear ordering over the base beliefs as described in chapters 3 and 4.

An Example of a DOBS Knowledge State

Figure 5.3 is a further expansion of the DOBS diagrams from Figures 5.1 and 5.2. Its explanations and examples, which follow, should help clarify the many parts of a DOBS and its knowledge state, *KS*. Many sections of the *KS* are described below with examples of the type of propositions that might be in them. For the sake of simplifying this example, assume all propositions inside the circle $\mathbf{B}^{\cup}$ were at one time asserted (also in the base) after which the formulas in $\mathbf{D}^{\cup}$ were derived (using these base beliefs) as recorded in H.

The derivations in H are stored as pairs containing the belief and the base beliefs underlying its derivation. For example, the pair $\langle c, \{a, a \rightarrow c\}\rangle$ expresses that c was derived from the base beliefs a and $a \rightarrow c$. This is an ATMS style of recording justification for a derived belief, though these derivations also match the JTMS style, since they are all one step derivations from base beliefs (cf. Section 2.4 for a TMS discussion).

After these derivations took place, some propositions were retracted from the belief base to produce the current base $\mathbf{B}$. These retracted propositions are now in $\mathbf{B}^{\cup} \setminus \mathbf{B}$, and consist of $p, a \rightarrow f, c \rightarrow d, c \rightarrow h, e, f, y, z, z \rightarrow w$, and $z \rightarrow y$.

Any propositions *requiring* one of these for support must not be in $\mathbf{D}$, but is in $\mathbf{D}^{\cup} \setminus \mathbf{D}$, instead (foundations theory; cf. Section 2.2.1). These propositions that are no longer known to be derivable from $\mathbf{B}$ are f, h, y, and w (plus d which is asserted in $\mathbf{B}$, but not known to be derivable from $\mathbf{B} \setminus \{d\}$).

Beliefs in the base $\mathbf{B}$ are asserted as self-supporting. Two of these propositions, c and d, are also in $\mathbf{D}^{\cup}$, because of the two derivations stored in H: $\langle c, \{a, a \rightarrow c\}\rangle$ and $\langle d, \{c, c \rightarrow d\}\rangle$, respectively. The first is a

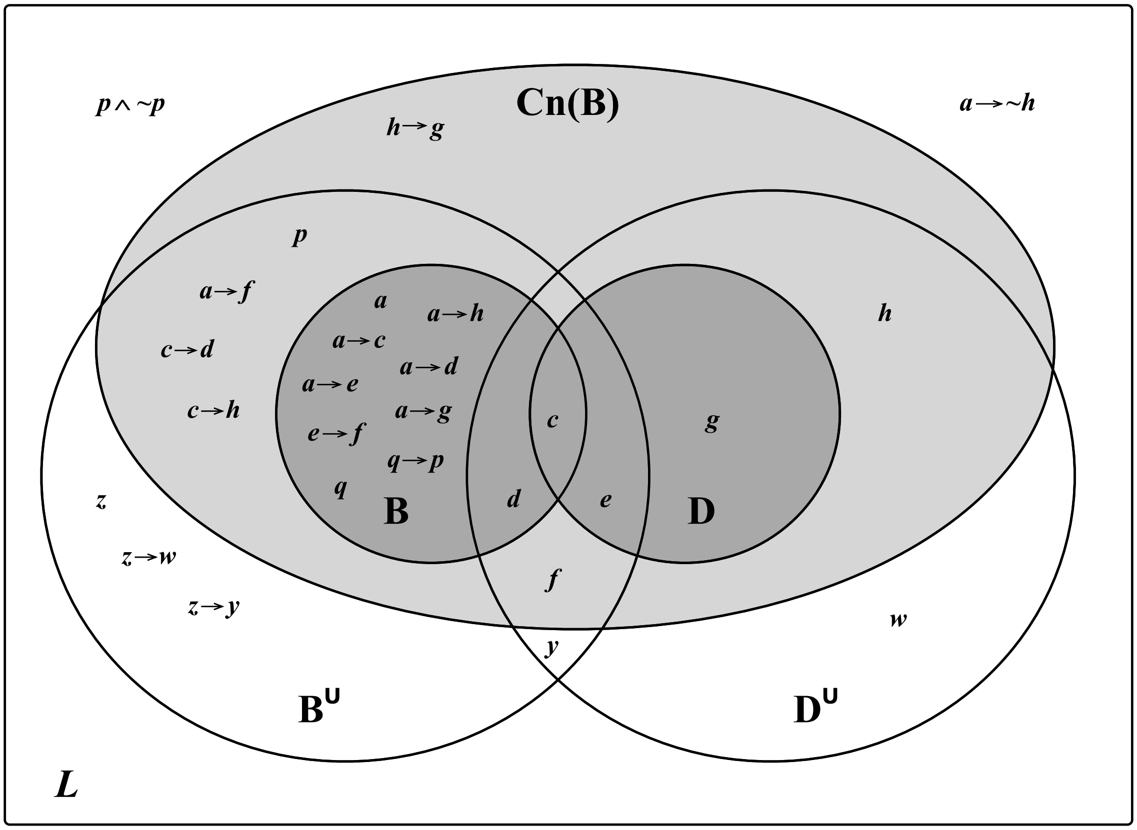

Figure 5.3: This figure is a further expansion of the DOBS diagrams in Figures 5.1 and 5.2. The shaded area represents the DOBS $= BS = B \cup D$. Remember that $Cn(B)$ is included for comparison purposes, only. Information regarding its boundaries and contents (with the exception of BS) are not a part of the knowledge state KS. For this KS, H contains the following derivations: $\langle c, \{a, a \rightarrow c\}\rangle$, $\langle d, \{c, c \rightarrow d\}\rangle$, $\langle e, \{a, a \rightarrow e\}\rangle$, $\langle f, \{a, a \rightarrow f\}\rangle$, $\langle g, \{a, a \rightarrow g\}\rangle$, $\langle h, \{c, c \rightarrow h\}\rangle$, $\langle w, \{z, z \rightarrow w\}\rangle$, $\langle y, \{z, z \rightarrow y\}\rangle$.

currently believed derivation, but the second is disbelieved due to the retraction of the proposition $c \rightarrow d$. This is shown by c being in $\mathbf{B} \cap \mathbf{D}$ whereas d is only in $\mathbf{B} \cap \mathbf{D}^{\cup}$.

The three beliefs in $\mathbf{D}$ — c, e, and g — are in three different sections All are known to be derived from B, but c is also asserted as a hypothesis and e is disbelieved as a hypothesis .

The propositions not mentioned, yet, are $p \wedge \neg p$, $a \rightarrow \neg h$, and $h \rightarrow g$. The first two are located in $\mathcal{L} \setminus (B^{\cup} \cup D^{\cup} \cup Cn(B))$ and the third is located in $Cn(B) \setminus (B^{\cup} \cup D^{\cup})$. These propositions are, actually, not known by KS, but they are included in the diagram as examples of the type of propositions that would be in these areas conceptually. A DOBS knowledge state only has information about the beliefs that exist in the union of its $B^{\cup}$ and $D^{\cup}$. A DOBS *system* that guarantees completeness (which, typically, requires a simple logic and a small base) might be able to calculate/determine some subset of the infinite number of elements outside $B^{\cup} \cup D^{\cup} \cup Cn(B)$.

5.3 DOBS Constraints

The key to understanding DOBS belief change constraints is to remember that they are applied in terms of the DOBS terminology. When removing *known* inconsistencies, they deal only with *known* derivations (stored in H) of *known to be derivable* propositions (stored in $D^{\cup}$). Deductive closure is not an option. For the purposes of this dissertation I assume that KS-closure is possible, though, for an extremely large system, time or space might restrict this.

Now that I have formalized a DOBS, I can assess the key changes necessary to adjust the list of integrity constraints from Section 2.2.2 so that they can be used as belief revision guidelines for a DOBS. Alterations are in **boldface**. Additions or clarifications are in *italics*. The revised constraints are:

1. a knowledge base should be kept **KS-consistent** whenever possible;
2. if a proposition **is known to be derivable** from the beliefs in the knowledge base *using the derivations currently known*, then it should be included in KS-closure of that knowledge base **(KS-closure)**;
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 a proposition should not need to be re-derived from a set of propositions from which it had previously been derived.

Constraint 3 reminds us that the beliefs removed to insure consistency should be directly related to some inconsistency—if unconnected to an inconsistency, a belief should remain in the base. Constraint 4 suggests removing weaker or less credible beliefs rather than those that are more credible (if given the choice).[3]

How to satisfy both Constraints 3 and 4 is an ongoing debate. A typical consideration is when do many weak beliefs overrule a single, more credible (up till now, that is) belief.

5.4 Dealing with Resource-Boundedness

The examples above are merely to illustrate that implemented systems are imperfect. Likewise, even an ideal DOBS will run into resource limitation problems. Each system can alter the DOBS formalization to suit its techniques for handling resource limitations. The following discussion involves implementation concepts, but it addresses them at a general theoretical level without getting into the minute details of implementation. These implementation adjustments to a DOBS must be understood when comparing different implemented systems.

The key to comparing systems is to consider NOT the state at rest, but the state of the DOBS when decisions need to be made.

5.4.1 Two Extremes

When implementing a KR&R system, one of the key questions is how to balance storage vs. re-derivation — i.e., what gets saved in memory vs. what items should be rederived. A system with a large domain, fast

[3]I do not consider the concepts of "entrenchment" as described in [Williams1994a], where the possible results of belief change are ordered specifically to include the fact that some beliefs cannot be removed without removing others.

hardware and efficient processes might choose to only save its base beliefs and rederive other beliefs whenever they are needed. In this case, $D^{\cup}$ and $B^{\cup} \setminus B$ would remain empty, and H would only store information during query or belief change procedures. After the procedure is completed, H could be emptied.

Alternately, a system that has lots of memory with fast look-up strategies but has a slow processor or inefficient inference algorithms would favor retaining all derivations and their histories in $D^{\cup}$ and H, respectively. This way, even beliefs that are retracted and then returned to the belief space will have any previously performed relevant derivations immediately available to them. This is also favorable for a system that alternates between several states (e.g., red light vs. green light; a video game for an experienced agent) rather than always encountering new states (e.g., a video game that is new or unfamiliar to the agent playing it).

Obviously, most systems fall somewhere between these two extremes. Cyc follows closely to the first, and SNePS favors the second. In both cases, however, belief change decisions are made only when H is full and active. During the decision process, these systems are very similar. SNePS focuses on only the relevant information in H, while Cyc fills H with relevant information.

5.4.2 Freeing Memory

The system that stores information in memory might find a need to reduce information stored to free memory space. The information most essential to maintaining the integrity of the knowledge state would be B, which must be retained at all costs, though subcontexts that are rarely used could be stored in some secondary storage location.

If KS-consolidation for review of previously discarded hypotheses is rarely used, $B^{\cup} \setminus B$ could be eliminated. Both $D^{\cup}$ and D can be rebuilt from B and H, so these can go as well – $D^{\cup} \setminus D$ first. Any removal from $D^{\cup} \setminus D$ could also include the removal of the relevant derivation information from H, since it is no longer essential to deriving *BS*.

The choice to drop D before H might be switched if the system is rarely *removing* beliefs from its *BS*. In this case, storing B and D for quick access might be preferable and H could be emptied. This raises two concerns.

1. The DOBS terminology is defined by knowing the derivation information in H. If that information is discarded, then it should be recognized that the system is storing a sub-version of the knowldege state, and that the presence of a belief, p, in D is evidence that $B \vdash_{KS} p$.[4]

2. When a belief p is to be retracted (or a contradiction detected), then, derivations relevant to the retraction (or contradiction) should be stored in H until the retraction process is completed and the new B and D are established. After this point, H could then be emptied.

Summary

These issues illustrate the need to compare implemented systems by more than their logic, size and theories. Systems should be analyzed according to their shortcomings as well. How do they handle these shortcomings – with ignorance, aggressiveness, optimism or caution? The system must suit the needs of its user. When resource limitations do arise, how does the system handle them?

By using a common formalization (like a DOBS), systems can be compared more easily and benchmarks for analyzing DOBS alterations will allow a standard for comparing and measuring the efficiency and accuracy as well as the shortcomings and potential hazards of implemented systems.

[4]This use of $B \cup D$ comes into question depending largely on the logic. If using a relevance logic, as in SNePS, the support set underlying a belief determines how that belief can be used in future derivations, making H a necessity. For a non-monotonic logic, the assertion of a new belief into $B \cup D$ requires questioning all beliefs in D (or at least those marked as defeasible).

5.5 DDR in a DOBS

DDR can be implemented in a DOBS system with minimal changes to the system knowledge base and algorithms. This assumes, however, that the system can calculate inconsistent sets and order beliefs linearly.

Given a DOBS knowledge state *KS*, KS-Complete-I(*KS*) is the set of inconsistent sets in $B^{\cup}$ that are *(believed to be)* minimally inconsistent.

Definition 5.5.1 KS-Complete-I(*KS*) $= \{N \mid N \subseteq B^{\cup}, N \vdash_{KS} \perp \wedge (\forall N' \subset N) : N' \nvdash_{KS} \perp\}$.

5.5.1 Defining a DOBS-DDR Knowledge State

To perform DDR in a DOBS, we need to expand the DOBS knowledge state to include the missing elements from a DDR knowledge state as defined in Chapter 4 (Definition 4.2.1): A DOBS-DDR knowledge state is a DOBS knowledge state with the additional elements (I, $\succeq$ and Q) that also designate it as a DDR knowledge state. Or, in other words, a DOBS-DDR knowledge state is a DDR knowledge state with the additional elements ($D^{\cup}$ and H) that also designate it as a DOBS knowledge state. Note that, although I is defined w.r.t. KS-Complete-I(*KS*), the definition of I satisfies the definition of I for a DDR knowledge state in Chapter 4.

Definition 5.5.2 A DOBS-DDR knowledge state *KS* is defined as $KS = \langle B, B^{\cup}, D^{\cup}, H, I, \succeq, Q \rangle$, where

- $B^{\cup} \subseteq \mathcal{L}$.
- $D^{\cup} \subseteq Cn(B^{\cup})$.
- $B \subseteq B^{\cup}$.
- H contains a record of every derivation that has been performed to derive the propositions in $D^{\cup}$.
- $I =$ KS-Complete-I(*KS*).
- $\succeq$ is a linear preference ordering on the beliefs in $B^{\cup}$

- Q is a priority queue consisting of a sequence, possibly empty, of tagged beliefs $(\langle p_1, \tau_1 \rangle, \ldots, \langle p_n, \tau_n \rangle)$ such that
  - $p_i \in X$
  - $i \neq j \;\Rightarrow\; p_i \neq p_j$
  - $p_i \succ p_{i+1}\,,\;\; 1 \leq i < n$
  - $\tau \in \{justout, in?, both\}$

Assume all DOBS-DDR knowledge states are in this form and subscripts (or prime quotes) can be used to distinguish between multiple knowledge states as discussed in Chapter 4 (Section 4.2.2).

For the purpose of this chapter, a NAND-set of a knowledge base *KS* is any subset of $B^{\cup}$ that is *(believed to be)* minimally inconsistent.

Definition 5.5.3 A set N is a NAND-set IFF $N \in$ KS-Complete-I(*KS*).

This is consistent with the introduction of NAND-sets in Section 4.1 and Definition 4.2.2: A set N is a NAND-set IFF $N \in I$.

A DOBS system cannot guarantee to know *all* minimally inconsistent subsets of $B^{\cup}$. Nor can it guarantee that the elements in I are minimally inconsistent. It can only guarantee that the sets in I are minimally KS-inconsistent. A DOBS system gradually builds I as contradictions are detected. The current state of I represents the hard constraints for the system *as far as is currently known*, which is the best that a DOBS can offer. In other words, I is where the system stores its knowledge regarding inconsistency. Note that KS-Complete-I(*KS*) satisfies the definition of I for a DDR *KS* as defined in Chapter 4 (Definition 4.2.1).

If a system derives both p and $\neg p$, The inconsistent set $\{p, \neg p\}$ stored in I has corresponding information in H. If, however, the system uses other techniques for detecting inconsistent sets that do not translate into a derivation of a belief and its negation (e.g., SATAN's global adjustment strategy identifying NAND-sets; cf.

Section 3.5.5), then there may be information in I that does not have a parallel representation in H[5]. For the purpose of this dissertation, I will assume that the information in I contains a superset of the information in *H regarding the KS-consistency* of that knowledge state.

In other words:

Observation 5.5.4 *I determines KS-consistency:* $(\forall S \in I, \forall B' \subseteq B^{\cup}) : S \not\subseteq B' \equiv B' \nvdash_{KS} \perp$.

And, as in Definition 4.2.16, Safe-per-I(KS) $\equiv (\forall S \in I) : \neg$Active$(S, KS)$.

Because of this, Safe-per-I(KS) equates to KS-Consistent(KS).

Observation 5.5.5 *For a DOBS-DDR knowledge state KS, Safe-per-I(KS) $\equiv$ KS-Consistent(KS).*

5.5.2 Base and Belief Credibility Values and a Linear Ordering Over Bases

The credibility value for any belief in $B^{\cup} = p_1, p_2, \ldots, p_n$ can still be calculated as in chapter 3 (Definition 3.2.1): $\text{Cred}(p_i, B^{\cup}, \succeq) = 2^{n-i}$. Again, as discussed in Chapter 3 (page 71), these credibility values are merely a numerical representation of the linear ordering of the beliefs that simplifies the definitions and proofs of belief base and knowledge state optimality.

The base credibility value of a DOBS-DDR knowledge state is defined the same way as defined in Chapter 4 (Definition 4.2.23): For a DOBS-DDR knowledge state KS that has the elements $B, I, \succeq$, and $B^{\cup} = p_1, p_2, \ldots, p_n$, $\text{Cred}(B, B^{\cup}, I, \succeq) =_{def} \sum_{p_i \in B} 2^{n-i}$ (the bit vector indicating the elements in B) when Safe-per-I(KS). Otherwise, when $\neg$Safe-per-I(KS), $\text{Cred}(B, B^{\cup}, I, \succeq)$ = -1.

The linear ordering over DDR-DOBS bases ($\succeq$) is defined as it was in Def 4.2.24) in Chapter 4: For two DOBS-DDR knowledge states KS and KS' that have identical elements for $B^{\cup}$, I, and $\succeq$: $B \succeq B'$ IFF $\text{Cred}(B, B^{\cup}, I, \succeq) \geq \text{Cred}(B', B^{\cup}, I, \succeq)$. Likewise, $B \succ B'$ IFF $\text{Cred}(B, B^{\cup}, I, \succeq) > \text{Cred}(B', B^{\cup}, I, \succeq)$. As before, $B \succ B'$ implies $B \succeq B'$.

[5] *H can* contain this information if, for every NAND-set N that is detected, $\langle \perp, N \rangle \in H$.

Comparing Bases of States with Different NAND-sets

Note that when ordering the bases of two knowledge states KS_1 and KS_2, not only must $B_1^{\cup} = B_2^{\cup}$ and $\succeq_1 = \succeq_2$ (as in Chapter 3) but $I_1 = I_2$ as well. This is easy to accomplish, because any two knowledge states with identical $B^{\cup}$ and $\succeq$ can merge the information in their elements for I, H, and $D^{\cup}$. Note, however that merging I_1 and I_2 does *not* result in $I' = I_1 \cup I_2$, because I' must then be adjusted for minimality—recall that $(\forall S \in I') : S \vdash_{KS} \perp$, and $(\nexists S' \subset S) : S' \nvdash_{KS} \perp \}$. This merging must be done before the bases of two DOBS-DDR knowledge states can be compared or ordered.

Changing I May Require Updating Q

Unfortunately, changing a knowledge state from KS_1 to KS_2 by altering its I component from I_1 to I_2, can result in an alteration of the status of retracted base beliefs—those previously designated as JustifiedOut or Protected-by-Q may no longer be such, so the priority queue must *also* be adjusted when I is altered or it may lose its Proper-Q status.

Assuming soundness[6], any change taking I_1 to I_2 must either

1. add a new NAND-set (that is not a superset of an existing NAND-set) or
2. replace an existing NAND-set N_1 with a *proper* subset $N_2 \subsetneq N_1$ that is also KS-inconsistent.

Assume a change is made under category (2). For $p =$Culprit$(N_1, \succeq)$, if $p \notin N_2$, then p must be DDR-Q-Inserted into the priority queue. There are two possible reasons for this—either

- N_1 was precarious making JustifiedOut(p, KS_1), and now p might be BadOut and thus need protecting; or
- N_1 was protecting p by satisfying lines $Q3$-$Q8$ for Protected-By-Q; and, since N_2 cannot protect p (it is not an IC-NAND-set for p in KS_2), p must protect itself.

[6]This assumes that the reasoning system is sound: i.e., I is not being altered by either (1) adding a new set to I that is *not* inconsistent or (2) removing some existing set in I that has just been discovered to be consistent.

Performing a full analysis to determine if one of these situation holds and whether the only recourse is for p to protect itself is computationally expensive and not necessary *at this time*. I suggest that p should merely be DDR-Q-Inserted into the queue, because one of its IC-NAND-sets is replaced by a NAND-set that it is not in (something easily determined when making the change from I_1 to I_2). The analysis of whether p can return to the base will happen in due time when DDR processes p. We never need to spend the time to analyze the details of how (or even whether) p is JustifiedOut *until* it is processed by DDR.

If making a change as described for category (1), there is no need to alter the queue. This is because $(\forall p \in X) : \text{BadOut}(p, KS_2) \Rightarrow \text{BadOut}(p, KS_1)$, and the protection in KS_1 is retained in KS_2. The new NAND-set cannot cause JustifiedOut(p, KS_1) to be BadOut(p, KS_2), because whatever precarious IC-NAND-set for p that made it JustifiedOut in KS_1 also exists in KS_2. Alternatively, if the new NAND-set is itself a precarious IC-NAND-set for p, it could convert BadOut(p, KS_1) to JustifiedOut(p, KS_2); but that would not require any Protection-by-Q for p, so there is no need to alter the queue.

5.5.3 DOBS-DDR Knowledge State Belief Change Operations

The operations that can be performed on a DDR knowledge state (defined in Section 4.2.12) can also be performed on a DOBS-DDR knowledge state. The operations affect $B^{\cup}$, $\succeq$, and Q in the same way as for a DDR knowledge state. $D^{\cup}$ and H are unaffected by DOBS-DDR knowledge state belief change operations. The changes to B and I are similar to the changes for a DDR knowledge state with the following adjustments:

- The operation of kernel consolidation that is used in defining some DDR operation is replaced with the KS-consolidation operation that was defined in Section 5.2.3. The DDR knowledge state belief change operations that are affected by this alteration are: consolidation, semi-revision, reconsideration, and optimized-addition.

- Belief change operations on a DOBS-DDR knowledge state are assumed to operate separately from the reasoning operations that might derive a new NAND-set. Therefore, I is unaltered by DOBS-DDR belief change operations. Since I is unaltered by reconsideration, it is also unaltered by DDR.

As in Chapter 4, I assume that contraction is done for consistency maintenance only. Therefore, all operations on a DOBS-DDR knowledge state consist of expansion, consolidation, semi-revision, reconsideration, and optimized addition; and these operations can be defined in terms of base expansion and/or KS-consolidation.

DOBS-DDR Knowledge State Expansion

The expansion of a DOBS-DDR knowledge state KS by the pair $\langle p, \succeq_p \rangle$ is written as $KS + \langle p, \succeq_p \rangle$ and results in the DOBS-DDR knowledge state KS_1 containing the following elements:

- $B_1 = (B + p)$, where $+$ is the operation of belief base expansion.
- $B_1^{\cup} = B^{\cup} + p$, where $+$ is the operation of belief base expansion.
- $I_1 = I$.
- $\succeq_1$ is similar to $\succeq$, but adjusted to include the preference information $\succeq_p$ as defined for DDR knowledge state expansion on page 147.
- Q_1 is similar to Q with alterations as defined for DDR expansion on page 147.

DOBS-DDR Knowledge State KS-Consolidation

$KS!_{KS}$ is KS-consolidation of the DOBS-DDR knowledge state KS which produces KS_1, where $B_1 = B!_{KS}$ (recall that $!_{KS}$ is the operation of DOBS belief base KS-consolidation as defined on page 191), and Q_1 is altered as described for DDR knowledge state consolidation on page 148. All other elements in KS are unaltered by DOBS-DDR knowledge state KS-consolidation.

DOBS-DDR Knowledge State KS-Semi-Revision

The operation of KS-semi-revision on a DOBS-DDR knowledge state KS is the addition of some belief p (along with the preference ordering information regarding that belief, $\succeq_p$) followed by KS-consolidation. It is written as $KS +_{!_{KS}} \langle p, \succeq_p \rangle$ and results in the DOBS-DDR knowledge state KS_1 containing the following elements:

- $B_1 = (B+p)!_{KS}$, where $+$ is the operation of belief base expansion and $!_{KS}$ is KS-consolidation performed using the information in $\succeq_1$.
- $B_1^{\cup} = B^{\cup} + p$, where $+$ is the operation of belief base expansion.
- $I_1 = I$.
- $\succeq_1$ is similar to $\succeq$, but adjusted to include the preference information $\succeq_p$ as defined for DDR knowledge state expansion on page 147.
- Q_1 is similar to Q with alterations as defined for DDR knowledge state kernel semi-revision on page 148.

DOBS-DDR Knowledge State KS-Reconsideration

$KS\circledcirc_{KS}$ is the KS-reconsideration of the DOBS-DDR knowledge state KS which produces KS_1 whose elements are the same as those in KS with the following exceptions: $B_1 = B^{\cup}!_{KS}$ (where $!_{KS}$ is the operation of KS-consolidation) and Q_1 is empty.

DOBS-DDR Knowledge State KS-Optimized-Addition

The operation of KS-optimized-addition on a DOBS-DDR knowledge state KS is the addition of a belief p (along with the preference ordering information regarding that belief, $\succeq_p$) followed by the KS-optimizing operation of KS-reconsideration. It is written as $KS +_{!_{\cup KS}} \langle p, \succeq_p \rangle$ and results in a new DOBS-DDR knowledge state, KS_1 containing the following elements:

- $B_1 = (B^{\cup} + p)!_{KS}$, where $!_{KS}$ is KS-consolidation, and $!_{KS}$ is performed using the information in $\succeq_1$. Note, however, that the new base is a result of KS-semi-revision of $B^{\cup}$ by p (not semi-revision of B by p)—this is the difference between *KS-semi-revision* of a DOBS-DDR knowledge state and *KS-optimized-addition* to a DOBS-DDR knowledge state. Recall: p is not *guaranteed* to be in B_1.
- $B_1^{\cup} = B^{\cup} + p$
- $I_1 = I$
- $\succeq_1$ is similar to $\succeq$, but adjusted to include the preference information $\succeq_p$ as defined for DDR knowledge state expansion on page 147.
- Q_1 is empty.

5.5.4 The QImpact and Preference Ordering for DOBS-DDR Knowledge States

QImpact(KS) for a DOBS-DDR knowledge state is defined as in definition 4.2.43 in Chapter 4—by the credibility value of its first belief:

Given a DDR knowledge state KS with the elements $B, B^{\cup}, I, \succeq$, and Q, if Empty(Q), QImpact(KS) $=_{def}$ 0; otherwise, given First(Q) $= \langle p, \tau \rangle$, then QImpact($KS$) $=_{def}$ Cred$(p, B^{\cup}, \succeq)$.

For the purpose of this dissertation and the discussion of DDR on a DOBS-DDR system, I will define knowledge state preference ordering assuming that $D^{\cup}$ and H have no effect on the preference ordering of DOBS-DDR knowledge states. In reality, more information in H or $D^{\cup}$ is preferred over less information, but this has no bearing on the DDR algorithm. This is because no reasoning is performed during DDR, so $D^{\cup}$ and H remain unchanged during DDR. I leave the full discussion of their effect on preference ordering and optimality for future work.

Two DOBS-DDR knowledge states (KS_1 and KS_2) that share the same $B^{\cup}$, I, and $\succeq$ (and $D^{\cup}$, and H) can be ordered as in chapter 4 (Definition 4.2.45):

Given the knowledge state KS containing elements $B, B^{\cup}, I, \succeq$, and Q, if $\neg$Safe-per-I(KS), then Cred($B, B^{\cup}, I, \succeq$)= -1, and $\nexists KS'$ s.t. $KS \succeq_{KS} KS'$. Otherwise, Safe-per-I(KS), and given the knowledge state KS_1 with the elements $B_1, B^{\cup}, I, \succeq$, and Q_1, and Proper-Q(KS) and Proper-Q(KS_1):

- If $B = B_1$ and QImpact(KS) = QImpact(KS_1), then $KS \succeq_{KS} KS_1$ and $KS_1 \succeq_{KS} KS$. Their queues may differ, as long as the first elements in their queues are the same.
- $KS \succ_{KS} KS_1$ if either
  - $B \succ B_1$ OR
  - $B = B_1$ and QImpact(KS) $<$ QImpact(KS_1).
- If $KS \succ_{KS} KS_1$, then $KS \succeq_{KS} KS_1$.

A DOBS-DDR knowledge state KS is KS-Optimal w.r.t. $B^{\cup}$, I, $\succeq$ (assuming a set $D^{\cup}$, and H), if it is preferred over all other knowledge states with the same $B^{\cup}$, I, $\succeq$ (and $D^{\cup}$, and H). This is identical to the definition for five-tuple DDR knowledge state optimality (Definition 4.2.51).

Definition 5.5.6 Given a DOBS-DDR knowledge state KS containing the elements $B, B^{\cup}, I, \succeq$, and Q, KS-Optimal(KS) $\equiv$ Optimal-per-I(KS) .

If I=Complete-I(KS), then the knowledge state has the complete set of minimally consistent NAND-sets stored in I and KS-Optimality equates to true optimality.

Theorem 5.5.7 *I=Complete-I(KS) $\Rightarrow$ (KS-Optimal(KS) $\equiv$ Optimal-per-$\perp$(KS))*

Proof.

I=Complete-I(KS) (prem.). I = Complete-I(KS) $\Rightarrow$ (Optimal-per-I(KS) $\equiv$ Optimal-per-$\perp$(KS)) (Obs 4.2.53). Therefore, Optimal-per-I(KS) $\equiv$ Optimal-per-$\perp$(KS). KS-Optimal(KS) $\equiv$ Optimal-per-I(KS) (Def 5.5.6). Therefore, KS-Optimal(KS) $\equiv$ Optimal-per-$\perp$(KS). $\square$

For a large, complex real-world knowledge representation and reasoning system, however, it is unlikely that I will ever be complete. If it is complete, though, then DDR run to completion results in the optimal knowledge state—the Optimal-per-$\perp$ knowledge state—as presented in Theorem 4.3.8, which also applies to a DOBS-DDR knowledge state. Typically, however, KS-Optimality is the best that a DOBS-DDR can provide. It states that the base knowledge state is optimal *given the information at hand* (given I, the record of all known inconsistencies that are KS-minimal). The only way for the base to be non-optimal is if it contains an inconsistency that has yet to be detected.

The definitions for Protected-By-Q(p,KS), Proper-Q(KS) and Proper-KS(KS) for a DOBS-DDR knowledge state are the same as those for a DDR knowledge state (except for the knowledge state having additional parts)—cf. Definitions 4.2.31, 4.2.32, and 4.2.33.

Because we also know that Safe-per-I(KS) $\equiv$ KS-Consistent(KS) (Obs 5.5.5), if KS is Proper-KS, it is also KS-Consistent:

Observation 5.5.8 *Given a DOBS-DDR knowledge state* $KS = \langle B, B^{\cup}, D^{\cup}, H, I, \succeq, Q\rangle$, *Proper-KS(KS)* $\equiv$ *KS-Consistent*$(KS) \wedge$ *Proper-Q(KS)).*

The DDR priority queue is maintained as in Chapter 4.

The definitions for the DDR invariant relations Proper-Q and Proper-KS, as well as their helper relation Protected-by-Q, also apply to a DOBS-DDR knowledge state (cf. Section 4.2.10). Their adjustment to the DOBS version is inherent in their dependence on the make-up of I. The theorems and proofs presented in that section also hold for a DOBS-DDR knowledge state.

5.6 DDR for a DOBS-DDR Knowledge State

The DDR algorithm and its helper functions need not be altered for a DOBS-DDR knowledge state. The only difference is that the knowledge state for a DOBS-DDR system is a 7-tuple rather than the 5-tuple assumed in chapter 4. This difference is most obvious in the algorithms during assignments by reference—where a knowledge state's elements are assigned individual variable names for reference during the algorithm or changes to these elements are assigned back into the knowledge base. In these instances, the elements of a DOBS-DDR knowledge base that are neither accessed nor affected by the DDR algorithms ($D^{\cup}$ and H) are not included in the chapter 4 algorithm assignments and are thus to be assumed unaltered when applying the chapter 4 algorithms to the chapter 5 DOBS-DDR knowledge state 7-tuple.

All DDR algorithm theorems and proofs from Chapter 4 also apply to a DOBS-DDR knowledge state.

The result of DDR performed to completion on a DOBS-DDR knowledge state is the KS-Optimal knowledge state.

Theorem 5.6.1 *Given that $KS = KS_{post}$ is the DOBS-DDR knowledge state resulting from the DDR(KS_{pre}) running to completion, Proper-KS(KS_{pre}) $\Rightarrow$ KS-Optimal(KS_{post}).*

Proof.

$KS = KS_{post}$ is the DOBS-DDR knowledge state resulting from the DDR(KS_{pre}) running to completion (given). Proper(KS_{pre}) (premise). Proper-KS(KS_{pre}) $\Rightarrow$ Optimal-per-I(KS_{post}) (Thm 4.3.7). KS-Optimal(KS) $\equiv$ Optimal-per-I(KS) (Def 5.5.6). Therefore, KS-Optimal(KS). □

DDR is a KS-optimizing algorithm for a DOBS-DDR system. All other anytime and additional benefits of DDR that are discussed in Section 4.5 hold equally well for DDR on a DOBS-DDR knowledge state. These benefits hold for para-consistent, monotonic logics like first-order-predicate logic and relevance logic [Anderson & Belnap1975, Anderson, Belnap, & Dunn1992]. This is because KS-Optimality is based on what the *system* has detected and not on complete reasoning or deductive closure. Any system whose detected NAND-sets are unaffected (except by proper subset refinement) by further knowledge acquisition can use the DDR algorithm to become KS-Optimal.

Further work is needed to analyze whether the detected NAND-sets in systems using *default* logic might be usable. The set $\{p, p \rightarrow q, \neg q\}$ might be a NAND-set, yet not if t is added to the base and p is a default belief that is blocked by t. There *are*, however, ways that DDR might assist a default logic system. These are discussed in Section 8.4.1 of Chapter 8.

5.7 In Summary

DDR can be implemented in a DOBS system. Performing DDR on a KS-consistent knowledge state (with a properly maintained priority queue) results in a KS-optimal knowledge state. If I is complete, then the result of running DDR to completion is a knowledge state that is Optimal (a.k.a. Optimal-per-$\perp$).

Chapter 6 presents the extended case study of such an implementation of DDR in an existing TMS system.

Chapter 6

DDR in SNePS: An Extended Case Study

6.1 Introduction

This chapter deals with implementing the theories presented in this dissertation into an existing knowledge representation and reasoning (KR&R) system called SNePS (Semantic Network Processing System) [Shapiro & The SNePS Implementation Group2004]. SNePS is implemented in LISP and the logic for SNePS is a relevance logic adaptation of first-order-predicate logic. Because this logic is monotonic and the system can detect and save NAND-sets, DDR will work equally well as with classical propositional logic. When run to completion, the knowledge state will be KS-Optimal.

The data structures used by DDR are the standard data structures for LISP (as defined by defstruct) which can have multiple fields (or slots) for holding values. These fields can be accessed by the system in order to perform reasoning, belief revision, actions, etc. When an item that is represented by such a data structure is placed on a queue, in a list, in a binary search tree, etc., it is not a copy of the item that is placed; the value of the pointer for that data structure is placed, instead. This saves both the space needed for holding a copy and the time needed to make a copy.

The key implementation issues are:

- implementing the *KS* including $B^{\cup}, B, X, I, \succeq, Q$
  - expanding existing data structures
  - adding new data structures
- working with the current SNePS implementation
  - using existing features: NAND set detection, user-selection of culprits
  - implementing new features:
    * developing a generic binary search tree (BST) data structure for storing ordered lists (sorted by descending credibility of culprit beliefs)
    * maintaining queues of active NAND-sets
    * user-determined linear ordering of culprits
    * automating belief ordering and culprit selection through use of source ordering
    * maintaining minimal (KS-minimal) NAND-sets
- implementing consolidation
- implementing the DDR algorithm and helper functions
- maintaining backward compatibility and making DDR invisible to users not interested in using it.

One major simplification of the original DDR design as defined in Chapters 4 and 5 is that not *all* beliefs are ordered linearly. The current implementation allows the user to select the culprit when a NAND-set is detected by the system. Once selected, it is removed by the system (effectively a consolidation operation on that NAND-set with the user's selection as the decision function) and the user assigns the culprit to a place in a linear ordering of culprits that gets built up *as* culprits are chosen. This is sufficient, because the only beliefs that get ordered in DDR are the retracted beliefs; therefore, we only need retracted beliefs to be linearly ordered.

Most of the implementation details in this chapter are the result of a close collaboration with Michael Kandefer, who performed the actual implementation coding of my data structures and algorithms. Although the basic algorithms, data-structures, and validation and regression testing, were my domain, Michael was solely in charge of dealing with the intricacies of integrating with the SNePS code, implementing a BST data structure in SNePS,[1] and developing new user commands.

6.2 Data Structures Altered and Added to Accommodate DDR

6.2.1 *ddr*

The global variable *ddr* allows the user who is not interested in DDR to be virtually unaffected by these changes made to implement DDR. When *ddr* is set to “t” (by a user wishing to use DDR), the knowledge state for DDR is maintained and DDR can be called by the user. When *ddr* is set to “nil” (the default setting to allow for backward compatibility), the system works much as it did before DDR implementation, though it still maintains a set of active NAND-sets (as described in Section 6.2.4) to support the addition of the consolidation operation to the SNePS system (cf. Section 6.2.6 for discussion of consolidation).

6.2.2 Expanding the Belief Node Structure to Accommodate DDR

In the SNePS network, beliefs are stored as *nodes* (cf. [Shapiro & The SNePS Implementation Group2004] for SNePS implementation details). Each SNePS node is made up of several fields that store information including the actual belief that the node represents. These fields also include the origin sets for that belief (ATMS-style), the name of the node, etc. To perform DDR in an efficient way, new fields are needed in the SNePS node and will be utilized by the ddr-algorithm code.

The fields added to the SNePS node are:

[1]The BST code that was integrated with SNePS came from a website belonging to Stuart Russell at University of California: `http://www.cs.berkeley.edu/ russell/code/utilities/binary-tree.lisp`. We are converting it to a red-black tree to insure computational efficiency.

- *ddr-q-tag* - a tag used to mark the state of this node on the ddr queue, nil if not in the queue ($ddr\text{-}q\text{-}tag \in \{justout, in?, both, nil\}$)
- *culprit-node* - a pointer to the culprit node in the *culprits* BST (Sec 6.2.4) that contains this node, nil if not a culprit node
- *ddr-q-node* - a pointer to the node in the *ddr-q* BST (Sec 6.2.4) that contains this node (saves time when finding a node in the queue), nil if not in the queue
- *ic-NAND-sets* - the list of NAND-sets in *I* that this node *is* a culprit of
- *nc-NAND-sets* - the list of NAND-sets in *I* that this node is in, but *is not* a culprit of

There is also a new SNePS data structure, called an *nv-pair*, that contains the node for a culprit belief along with its value:

- *node* - the ddr node in this structure (a SNePS node, with the added DDR fields)
- *value* - the value for this node; range is (0,1); the lower the value the weaker the credibility of the belief.

The BSTs for culprit beliefs (Sec 6.2.4) store nv-pairs, not just the beliefs. The *value* is a real number assigned *by the system* to insure that the ordering over the culprits remains linear.[2]

Comparing *value* and Credibility Value

Although the sum of these real number values of the culprits in no way represents the value of a base (unlike the way credibility values were used in Section 5.5.2), a non-increasing ordering of the culprit beliefs by these real number values matches the linear ordering of the culprit beliefs by $\text{Cred}(p_i, B^{\cup}, \succeq)$: for culprit beliefs p_i and p_j, $p_i \succ p_j$ IFF $\text{Cred}(p_i, B^{\cup}, \succeq) > \text{Cred}(p_j, B^{\cup}, \succeq)$ (from Sec 5.5.2) and $p_i \succ p_j$ IFF value(p_i) $>$ value(p_j).

[2]Cf. Sec 6.2.7 for details on *how* and *why* this linear ordering is maintained in this way.

6.2.3 The Data Structure for a NAND-set

The key DDR-related fields in the data structure for a NAND-set are:

- *name* - name of this NAND-set (includes a unique identifying number; e.g., N23)
- *nodes* - the set of nodes (beliefs) in this NAND-set (its inconsistent set; a SNePS nodeset)
- *size* - cardinality of *nodes*
- *b-hyp-count* - cardinality of ($nodes \cap B$)
- *test-num* - (default = 0) assigned *NAND-set-test-num* for any procedure accessing this NAND-set
- *temp-count* - counter utilized by ddr functions; zeroed whenever *test-num* is re-assigned
- *culprit* - the culprit node for this NAND-set (its weakest element, currently selected by the user)
- *culprit-strength* - strength (culprit value) of this NAND-set's culprit.

The fields for *size* and *b-hyp-count* are used to check whether a NAND-set is Active or Precarious. If $b\text{-}hyp\text{-}count = size$, the NAND-set is Active. If $b\text{-}hyp\text{-}count = size - 1$, then the NAND-set is Precarious.

test-num*, *temp-count* and *NAND-set-test-num

Several procedures access NAND-sets numerous times and keep track of how many times each NAND-set is accessed. However, it is possible that only a small subset of all the NAND-sets will be accessed during this procedure. Each NAND-set has a field for a temporary counter (*temp-count*) for keeping track of the number of times it is accessed by the procedure; *temp-count* is incremented each time the NAND-set is accessed.

To avoid having to zero *all* the NAND-sets' temporary counters (computationally expensive if only a small subset will be accessed), I make use of the global variable *NAND-set-test-num*, which is incremented each time one of these procedures is called.[3] Each NAND-set has another field (*test-num*) that is

[3] These procedures never overlap or call each other, so *NAND-set-test-num* is never incremented until the previous procedure is completed (or stopped).

assigned the value of *NAND-set-test-num* when that NAND-set is *first* accessed by the procedure. Only when *test-num* gets a re-assignment is the value of *temp-count* reset to 0. This way, only the NAND-sets that are *actually* accessed get their temporary counters zero-ed. The procedures using this technique are Update-with-new-NAND-set (as shown in its algorithm on page 218) and Safe-Return (as mentioned in the discussion following its algorithm on page 225).

This system of using the global *NAND-set-test-num* and two NAND-set fields permanently allocates memory space for maintaining these counts. This could also be implemented using a hash table that uses space only when the procedure is active. It can hash on the unique number in the *name* of the NAND-set and can grow dynamically as more NAND-sets get accessed. My implementation sacrifices memory for a guarantee of a lookup time (the time to increment NAND-set's counter) that is worst case $O(1)$, whereas the look-up time for a hash table is *average* time $O(1)$—but, potentially worse. The choice between reducing memory load vs. run-time is best handled by the system implementers. I describe the details here, because they are essential to the complexity analysis that follows in Section 6.2.5.

6.2.4 Implementing the Knowledge State *KS*

The DDR knowledge state *KS* is defined as the five-tuple $\langle B, B^{\cup}, I, \succeq, Q\rangle$.

SNePS already maintains a current context ($= B$) and all base beliefs ($= B^{\cup}$). The *KS* implementation adds three BSTs for ordering culprit beliefs and three more BSTs for ordering NAND-sets. In all cases, the elements in the BSTs are ordered by the *value* of the culprit (in non-increasing order). Culprit BSTs store nv-pairs. NAND-set BSTs store NAND-sets ordered by the *value* (= *culprit-strength*) of their culprits. This takes advantage of the *linear ordering* of the beliefs in the nv-pairs, and results in a *total pre-order*[4] over NAND-sets (total pre-order, because every NAND-set has a culprit, but two different NAND-sets can have

[4]Cf. Sec 2.5 for a review of different orderings.

the same culprit—giving up anti-symmetry).

The three BSTs that store culprit belief nv-pairs for a knowledge state *KS* are:

- *culprits* - BST of *all* culprit nodes ($= \succeq$) and
- *removed-beliefs* - BST of the beliefs removed from B ($= X$)
- *ddr-q* - BST of the culprits on the priority queue ($= Q$)[5]

When SNePS detects a NAND-set, the user is given the option of (1) resolving the inconsistency by selecting one (or more) of the beliefs in the NAND-set and removing it from the base[6] or (2) ignoring the inconsistency (letting the NAND-set remain a subset of the current base). For the purpose of this dissertation, I assume that the user chooses a *single* culprit for retraction to restore consistency. I leave the case of the user selecting multiple culprits for a single NAND-set to future research.

In the second case, the knowledge state is KS-inconsistent, and no culprit has been selected. If such a NAND-set becomes non-Active, the weakest of its retracted beliefs is designated as its culprit.

The three BSTs that store NAND-sets in a knowledge state *KS* are:

- *all-NAND-sets* - BST of *all* NAND-sets in the system ($\equiv I$)
- *active-NAND-sets* - BST of active NAND-sets for this KS (used to implement consolidation)
- *active-NAND-sets-nc* - non-ordered list of NAND-sets with no culprits (where the user will need to select culprits for retraction to eliminate the inconsistencies).[7]

6.2.5 Implementing I

The need for *KS* to contain three BSTs of NAND-sets foreshadows the complexity of maintaining I and the subsets of I that are needed to implement DDR.

[5]The nv-pair is different from the pair assigned to the queue in the text (cf. Chapter 4). In the text, the pair includes a tag $\tau \in \{in?, both, justout\}$. In the implementation, the tag for a belief is stored in the belief's *ddr-q-tag* field. This is helpful, again, for debugging and code development as well as insuring a one-step lookup to determine the tag for any given belief. Recall that the tag is *nil* if the belief is *not* on the queue.

[6]This is equivalent to performing kernel consolidation on that NAND-set.

[7]The use of this data structure was developed by Michael Kandefer.

Storing and Updating Elements of I

I is a set of minimal (as far as is currently known; KS-minimal) NAND sets. To maintain this minimality, any additions to I must be tested for minimality. If the addition is a superset of an existing NAND set, then it is not added. If the addition is a proper subset of any existing NAND sets, it gets added and the supersets get removed.

procedure Update-with-new-NAND-set(N, KS) ;;; N is the new NAND-set

  NAND-set-test-num$\leftarrow$ (1+ *NAND-set-test-num*)

  loop for each p in N

    for each N' in (*nc-NAND-sets*(p) $\cup$ *ic-NAND-sets*(p)) do

      when *test-num*(N') $\neq$ *NAND-set-test-num* **do** ;;; is this first access of N' for this update

        test-num(N') $\leftarrow$ *NAND-set-test-num* ;;; sets to the number for *this* update

        temp-count(N') $\leftarrow 0$;;; zeros *temp-count*

      temp-count(N')$\leftarrow$ (1+ *temp-count*(N')) ;;; increments *temp-count*

      when *temp-count*(N') = *size*(N') **do** ;;; ($N' \subseteq N$)

        STOP - Exit Update-with-new-NAND-set — do NOT add new NAND-set to I

      when *temp-count*(N') = *size*(N) **do** ;;; $N \subsetneq N'$

        if *culprit*(N') $\in N$ **then**

          culprit(N)$\leftarrow$ *culprit*(N')

          Replace(N',N,KS)

        else

          for each $n' \in N'$ **do**

            Remove N' from *nc-NAND-sets*(n') and *ic-NAND-sets*(n')

Remove N' from : *all-NAND-sets*, *active-NAND-sets*, and *active-NAND-sets-nc*

Add N to *active-NAND-sets-nc*

DDR-Q-Insert(*culprit*(N'),Q, $\succeq$) ;;; insures Proper-Q; cf. Section 5.5.2

for each $n \in N$ **do**

Add N to *nc-NAND-sets*(n)

endif

procedure Replace(old-N, new-N, KS)

;;; replaces an existing old-N with its proper subset new-N

;;; only called when when *culprit*(old-N) $\in$ new-N

for each node n in new-N, **do** replace old-N with new-N in n's NAND-set lists;

for each node n in old-N but NOT in new-N, **do** remove old-N from n's NAND-set list

replace old-N with new-N on all NAND-set lists (*all-NAND-sets* and *active-NAND-sets*)

This algorithm used for maintaining minimality in I offers an improvement over the computational complexity of traditional subset/superset testing. During this testing to insure that a new NAND-set A is *neither* a subset *nor* a superset of any set N_i in the current $I = \{N_1, N_2, \ldots, N_n\}$:

- no element that is *not* in A is ever accessed, and
- the complexity of processing some $p \in A$ is $O(j)$, where $j = \mid \{N_i \mid N_i \in I \wedge p \in N_i\} \mid$.

The only additional work occurs if the new NAND-set is a proper subset of an existing NAND-set.

An altered version of this algorithm might also be useful for maintaining the origin sets of derived beliefs, which also require minimality.[8]

[8]The number and size of origin sets for any given belief may be so small it, it would not be worth the overhead. This would best be determined by benchmarking with large, intricate and interconnected knowledge bases.

6.2.6 Implementing Consolidation

The algorithm for consolidation of a knowledge state *KS* is similar to calling KS-Add-Remove and Safe-Return with the following changes: no belief is being added to the base, and the set of beliefs being removed (R) is determined by Safe-Return processing all *active* NAND-sets rather than the NC-NAND-sets of p.

Because there are no known culprits for the NAND-sets in *active-NAND-sets-nc*, these are processed using an interactive user interface, where the user selects the culprits (as with standard culprit selection in SNePS). NAND-sets in *active-NAND-sets* are processed in decreasing order of the credibility of their culprits, and the culprit of an Active NAND-set is retracted.

As culprits are retracted during consolidation, any NAND-sets (in either *active-NAND-sets* or *active-NAND-sets-nc*) that become non-Active are removed from their respective lists. The knowledge state is KS-consistent IFF *active-NAND-sets* $\cup$ *active-NAND-sets-nc*$= \emptyset$.

The current implementation performs consolidation only when the user *explicitly* calls for consolidation to be performed. Similarly, the current implementation performs DDR only when the user *explicitly* calls for DDR to be performed. Since DDR can only be run if the knowledge state is KS-consistent, and it is reasonable that the user would prefer to save the effort of manually calling consolidation prior to calling DDR, consolidation is the first step of the implemented DDR algorithm. This is a deviation from the algorithm in Chapter 4, but it is practical given that SNePS allows the system to operate in a KS-inconsistent state.

6.2.7 Selecting Culprits and Ordering Beliefs

The Selection and Linear Ordering of Culprit Beliefs

The ordering in the ideal system introduced in Chapter 4 is a linear ordering over *all* beliefs. This was to simplify the proofs using the credibility value of a base. Michael Kandefer gets full credit for the algorithm and implementation that places a new culprit belief into the linear culprit ordering.

In this implementation,

- any belief that is removed from the base is considered a culprit belief, and
- whenever a new culprit belief is selected by the user, the user is asked to place that belief in a linear ordering of the existing culprit beliefs.

Since all BSTs are ordered based on culprit values, the lack of ordering among non-culprit beliefs will not affect these data structures. We can redefine the credibility of the base B of a knowledge state KS as follows:

Definition 6.2.1 Given KS with the elements B, $B^{\cup}$, and *culprits*(KS)= $c_1, c_2, \ldots, c_n$,

$(\exists B' \subseteq B^{\cup}) : B' = B^{\cup} \backslash culprits(KS)$ and $(\forall KS_1 = \langle B_1, B^{\cup}, all\text{-}NAND\text{-}sets, culprits(KS), Q_1 \rangle) : B' \subseteq B_1$.

- If Safe-per-I(KS), then Cred($B, B^{\cup}$, *all-NAND-sets*, *culprits*(KS)) $=_{def} \sum_{c_i \in (B \cap culprits(KS))} 2^{n-i}$.
- If $\neg$Safe-per-I(KS), then Cred($B, B^{\cup}$, *all-NAND-sets*, *culprits*(KS)) $=_{def} -1$.

Given a set $B^{\cup}, I$, and $\succeq$, every possible base differs only in its culprit beliefs (all non-culprit beliefs are present in all bases), the credibility value described above is sufficient to compare bases as in previous chapters, but based on the linear ordering of culprit beliefs alone.

The interface for assisting the user in placing a new culprit into the pre-existing chain of culprits prints out a linear representation of the pre-existing culprits with numbers for the spots between each culprit and at either end of the ordering:

```
You have selected wff9: ¬q(a) as the culprit and an ordering
needs to be provided.  The current culprit nodes are:

wff8: ¬r(a)

wff6: m(a)

From the list below, choose a numbered slot to insert wff9: ¬q(a)
node into.  The beliefs are ordered from most preferred to least
preferred:  [1] wff8: ¬r(a) [2] wff6: m(a) [3]
```

```
Please, enter a number between 1 and 3.  The culprit will be
inserted in the slot with the number specified:
```

The user then enters the number indicating the location in the chain where the new culprit should be placed.[9] Based on the user's selection, the culprit node is given a value equal to the average of the values of its two surrounding nodes. If assigned the leftmost position ([1]), the node value is the average of 1 and the value of the node on its right. If assigned the rightmost position ([3] in the example above), the node value is half the value of the node on its left (= averaging with 0). Real numbers are preferred to positive integers, because insertion within the chain of culprits does not require shifting up the numerical values of the beliefs that are more credible than the newly inserted node. Again, I emphasize that the value of a culprit node is merely used to assist with placement within the various BSTs, and the linear ordering of culprits is to be considered a *qualitative ordering*, not a quantitative one.

Once a new culprit node has a value, various BSTs are updated to reflect this new information:

1. *culprits* gets the new nv-pair (the culprit node and its value) inserted into it
2. the same nv-pair is also inserted into *removed-beliefs*
3. the same nv-pair is inserted into *ddr-q* and the *ddr-q-tag* for the belief is set to *justout*
4. the NAND-set whose inconsistency was just resolved is updated with its new culprit in *all-NAND-sets*; along with any other NAND-sets containing the just selected culprit node that were on *active-NAND-sets-nc*
5. for all NAND-sets containing the culprit node:
   - if on an active list, they are removed—removed from *active-NAND-sets* and *active-NAND-sets-nc*;
   - their *b-hyp-count* is decremented.

[9]Although every culprit node has a value associated with it (ranging from 0 to 1), these values are used by the system to maintain the chain only. They are visible in some output of the current implementation, but this is to allow error checking by the knowledge engineers. In the final implementation, the user should never see the value of a culprit node.

Automating Belief Ordering by Ordering Sources

In earlier work [Johnson & Shapiro1999b, Johnson & Shapiro1999a, Johnson & Shapiro2000b], belief ordering was achieved by ascribing beliefs to specific sources and/or categories and ordering those sources or categories. The beliefs were then ordered as their source/category was ordered.

On the positive side, *all* beliefs were entered into the ordering, which allowed for an automatic belief revision algorithm to be applied to resolve NAND-set inconsistencies—provided the ordering resulted in a least element in each NAND-set that was derived.

On the negative side:

1. this required a lot of time preparing input, because every base belief needed a second base belief assigning it to a specific source or category;
2. the ordering was a total pre-order, which *could* have resulted in a NAND-set having no least element and thus requiring the user to select a culprit, anyway, albeit from the weakest subset in each NAND-set;
3. this same pre-order also makes the definition for optimal and preferred knowledge states less clear, a problem reserved for future research.

If the source or category information about base beliefs could be determined at input time (i.e., not requiring additional input from the user), this technique would be *very* beneficial.

For the initial implementation of DDR, I elected to streamline the ordering process. The user need only consider the ordering for culprits. I leave category ordering for use by DDR algorithms for future work.

6.2.8 Implementing Q

Q is implemented as the *ddr-q* of a knowledge state, which is a BST storing nv-pairs. Each node in an nv-pair has a field that contains its priority queue tag (*ddr-q-tag*).

6.3 Implementing DDR

The implementation of DDR is straightforward—following the algorithms presented in Chapter 4. When computing the complexity of the algorithms, all assignments take $O(1)$ time, because they are simple value assignments or pointer assignments (for data structures). The algorithms shown in the subsection below are the algorithms presented in Chapter 4. The discussion following each algorithms explain any intricacies regarding *how* they are implemented, and discusses the complexity of the implementation.

6.3.1 Safe-Return

function Safe-Return(p, KS) ;;; returns a set of beliefs

  $\langle B, B^{\cup}, I, \succeq,\ Q\,\rangle \leftarrow_{ref} KS$

  $R \leftarrow_{ref} \{\ \}$;;; initializing R

  $NQ \leftarrow_{ref}$ *NAND-set-Q*(NC-NAND-sets(p, KS) , $\succeq$) ;;; p's NC-NAND-sets in a queue

  loop until Empty(NQ)

    $N \leftarrow_{ref}$ Pop(NQ) ;;; First element in NQ removed into N

    if $(N \setminus \{p\}\ \subseteq\ B$ AND $N \cap R = \emptyset)$ **then**

      $R \leftarrow_{ref} R \cup \{$Culprit($N, \succeq$)$\}$;;; culprit for N inserted into R

    endif

  end loop

  return R

The time to make the BST NQ is $O(n \lg n)$, where $n = |$NC-NAND-sets(p, KS)$|$. Incorporated into that cost is the $O(n)$ cost conversion of the BST to a list where Pop has a cost of $O(1)$.[10] Note that this cost

[10] $O(n \lg n + n) = O(n(\lg n + 1)) = O(n \lg n)$.

is incurred only if the belief p is being returned to the base, thus increasing the credibility of the base and moving the knowledge state closer to its KS-optimal state.

The loop is a process that requires incrementing the global variable *NAND-set-test-num*. For each N popped off NQ, if *test-num*$\neq$*NAND-set-test-num*, then

test-num$\leftarrow$*NAND-set-test-num* and *temp-count*$\leftarrow 0$.

The test for $N \setminus \{p\} \subseteq B$ takes $O(1)$ time by checking if *b-hyp-count*=*size*-1. The test for $N \cap R = \emptyset$ also takes $O(1)$ time, because it is merely a test for *test-num*$= 0$ (as explained below).

When a belief r is added to R, that addition is done in $O(1)$ time, because r is merely pushed onto the list R (no ordering is necessary, because all elements in R are removed from the base during the procedure Add-Remove, regardless of their order).

There is a more complex step to the addition of the belief r to R, however. Each NAND-set containing r must have *temp-count* incremented—indicating that it has a non-empty intersection with R. The cost of this update for any given r added to R is $O(m)$, where $m = |$*ic-NAND-sets*$(r) \cup$ *nc-NAND-sets*$(r)|$. This cost, however, is *only* incurred when the belief r is determined to conflict with p's return to the base—i.e., p's return to the base changes r into a belief designated as MustOut.

If the testing for $N \cap R = \emptyset$ was performed using a subset-test that compared the elements of the two sets, the expense of that comparison would be incurred for every test, as opposed to an expense incurred only when a conflicting belief is found. This would be more expensive, computationally, assuming that the number of beliefs in a NAND-set can be very large compared to the number of NAND-sets for any *culprit* belief *that is currently in the base*.

6.3.2 Add-Remove

procedure KS-Add-Remove(KS, p, R),

$\langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle \leftarrow_{ref}\ KS$

$Q_1 \leftarrow_{ref}$ New-Q( )

for each r in R **do**

$Q_1 \leftarrow_{ref}$ Insert($\langle r, justout\rangle$, Q_1 , $\succeq$) $\rangle$;;; $\leftarrow_{ref}$ for clarity, Insert is destructive.

endfor

$KS\ \leftarrow_{ref}\ \langle\ B \leftarrow_{ref} (B \cup \{p\}) \setminus R,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,$ Popped(Merge-Qs$(Q, Q_1, \succeq))\rangle$

The time to perform Add-Remove can be broken into several parts; and, for this analysis, let us assume the implementation uses a BST for X, as well as for B and priority queues.

1. Building Q_1 takes $O(n_R \lg n_R)$ time, where $n_R = \mid R \mid$.
2. Merging Q and Q_1 takes $O(n_R \lg(n_R + n_Q))$ time, where $n_Q = \mid Q \mid$.
3. Popping the queue takes $O(\lg(n_R + n_Q))$ time.
4. Removing R from the base takes $O(n_R(\lg n_B + \lg n_X))$, where $n_B = \mid B \mid$ and $n_X = \mid X \mid$.
5. Inserting p into the base takes $O(\lg n_B + \lg n_X)$ time.

As in with Safe-Return, however, these expenses are only incurred if a belief is being inserted into the base improving the credibility of that base. Also note that as R approaches $\emptyset$, the total computation expense approaches $O(\lg n_B + \lg n_X)$, because expenses 1, 2, and 4 all approach $O(0)$ and $\mid X \mid \geq \mid Q \mid$ which makes the expense of 3 included in the expression of the expense of 5.

6.3.3 Process-Justout

procedure Process-Justout (KS, p)

$\langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ Q\ \rangle \leftarrow_{ref} KS$

for each $N \in$ NC-NAND-sets(p, KS)

$q \leftarrow_{ref}$ Culprit(N)

if $q \notin B$ **do** ;;; Note: $p \succ q$

$Q \leftarrow_{ref}$ DDR-Q-Insert$(q, Q, \succeq)$;;; $\leftarrow_{ref}$ for clarity, DDR-Q-Insert is destructive.

endif

endfor

$KS \leftarrow_{ref} \langle B, B^{\cup}, I, \succeq,$ Popped(Q) $\rangle$

The for-loop in Process-Justout takes $O(n \lg(q+n))$, where $n = |$NC-NAND-sets$(p, KS)|$ and $q = |Q|$. This is because the loop processes n times, DDR-Q-Insert has a worst case $O(\lg q)$ cost, and worst case has every DDR-Q-Insert actually inserting a belief onto (rather that changing its tag or just leaving it alone). If all the retracted culprits in NC-NAND-sets(p, KS) were already on the queue, the entire for-loop would take $O(n)$ time, because the Change-Tag procedure is $O(1)$.

Popping the queue takes $O(\lg(q+n))$ time. If all the retracted culprits in NC-NAND-sets(p, KS) were already on the queue, it would reduce to $O(\lg q)$.

6.3.4 Update-KS

procedure Update-KS $(KS, update, p, R)$

$\langle B, B^{\cup}, I, \succeq, Q \rangle \leftarrow_{ref} KS$

Case *update* =

PopQ:

$KS \leftarrow_{ref} \langle B, B^{\cup}, I, \succeq,$ Popped$(Q) \rangle$

ProcJustout:

Process-Justout(KS, p)

AddRem:

KS-Add-Remove(KS, p, R)

Popping the queue (when *update* =*PopQ*) takes $O(\lg n)$ time, where $n = \mid Q \mid$.

The other costs are discussed above.

6.3.5 DDR Algorithm

procedure DDR(*KS*)

loop until Empty(Q_{KS})
    $\langle p, \tau \rangle \leftarrow_{ref}$ First(Q_{KS})
    if$_1$ ($\tau = in?$ or $\tau = both$) , **then**
        can-return $\leftarrow$ BadOut(p, KS)
        if$_2$ *can-return* , **then** (note: BadOut(p, KS))
            $R \leftarrow_{ref}$ Safe-Return(p, KS)
            update $\leftarrow$ *AddRem*
        else$_2$ (note: JustifiedOut($p.KS$))
            if$_3$ $\tau = both$, **then**
                update $\leftarrow$ *ProcJustout*
            else$_3$ (note: $\tau = in?$)
                update $\leftarrow$ *PopQ*
            endif$_3$
        endif$_2$
    else$_1$ (note: $\tau = justout$)
        update $\leftarrow$ *ProcJustout*

```
17        endif_1
18        Update-KS(KS, update, p, R)          ;;; DESTRUCTIVE—alters the DDR knowledge state KS
19    end loop
```

Testing the queue to see if it is Empty (line 1) takes $O(1)$ time.

Getting the first element off the queue (line 2) should take $O(1)$ time assuming that the data-structure for the queue includes a pointer to the first element. Resetting such a pointer during the Popped procedure does not increase the cost of the procedure.

Determining the tag for the belief p (line 3) takes $O(1)$ time, because it is in *ddr-q-tag*(p).

Determining *can-return* requires checking all NAND-sets in *ic-NAND-sets*(p) to see if any are precarious (*count*$=$*size*-1). This takes $O(n_1)$ time, where $n_1 = |$*ic-NAND-sets*$(p)|$.

All other costs are discussed above.

In summary, the computational cost for one pass through the DDR loop varies widely, but it is relatively small unless the belief being processed is being returned to the base. The jump in cost is a reasonable price to pay to improve the credibility of the knowledge state base. Additionally, the largest part of the cost of returning p comes from (1) determining the set R which must be removed to maintain consistency, (2) removing R from the base into the exbase, and (3) putting all elements of R onto the queue.

6.4 DDR Implementation Example Showing Anytime Features

The DDR example in this section is based on the graph of beliefs and NAND-sets shown in Figure 6.1, which is a slightly altered subgraph of the graph shown in Figure 4.2 used in DDR Example 4.4 on page 164.

Due to system constraints, these beliefs have to be represented in predicate form, and implication arrows are double-arrows: e.g., $p \rightarrow \neg q$ in the graph (and in earlier chapters) becomes $p(a) \Rightarrow \neg q(a)$ in the system. The selections of output shown in the example below highlight the following aspects of the DDR implementation (elements that were not newly added to the SNePS system for DDR are labeled as "basic SNePS"):

- entering beliefs into the SNePS system (basic SNePS)
- querying the system to see what it knows (basic SNePS)
- choosing the culprit ordering after culprit selection produces a new culprit
- calling DDR
- stopping DDR before it is finished—highlighting anytime aspects of DDR
- adding more beliefs to the base, and deriving a new NAND-set (basic SNePS)
- removing another culprit, and ordering it as more credible that the top element in *ddr-q*
- calling DDR, running it to completion, and showing DDR is interleavable and optimizing.

All indented, non-bulleted `typewriter` font indicated sections of a single run in SNePS using the SNePSLOG interface (cf. [Shapiro & The SNePS Implementation Group2004] for detailed information regarding SNePS and SNePSLOG). The SNePS run is very lengthy, so it has been edited in the following ways:

- portions that do not illustrate the implementation of DDR are deleted;
- redundant sections or sections shown earlier are also deleted;
- sections have been interspersed with commentary in normal font with only beliefs in `typewriter` font;
- the symbol for negation in the SNePS system ($\sim$) is replaced with $\neg$ in the run below;
- input to SNePS is in **`bold typewriter`** font;
- but, other than these edits, the output shown below has not been altered.

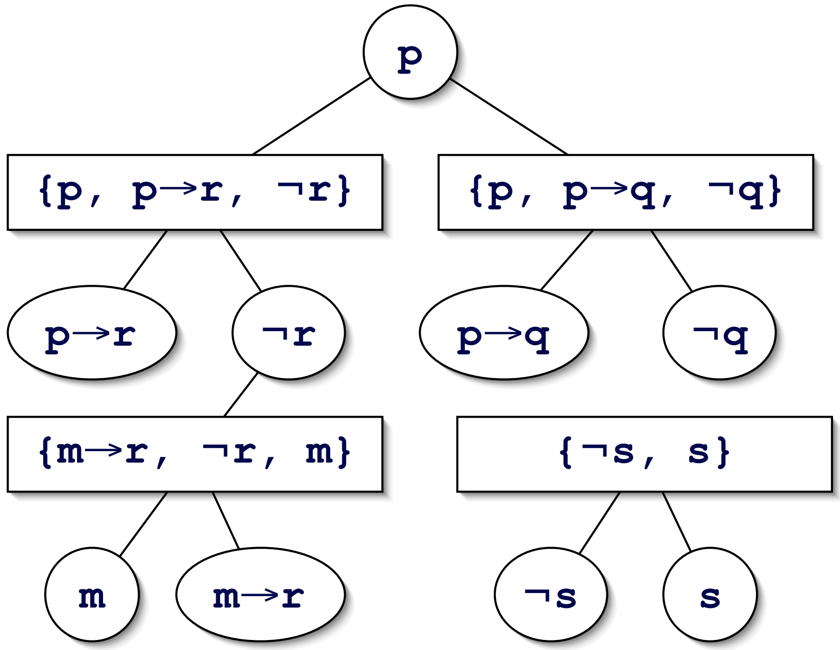

Figure 6.1: This graph shows the beliefs (circles/ovals) and NAND-sets (rectangles) that are used in the implemented DDR example presented in Section 6.4. Due to system constraints, these beliefs have to be represented in predicate form, and implication arrows are double-arrows: e.g., $p \rightarrow \neg q$ in the graph becomes $p(a) \Rightarrow \neg q(a)$ in the system. The culprit in each NAND-set is the right-most belief in the box. The linearly ordered set of culprits is $C = \neg q, \neg r, s, m$. The optimal base has all culprit beliefs retracted except for m.

The following beliefs are entered into the SNePS sytem using the SNePSLOG interface, which uses a colon (:) as its prompt [Shapiro & The SNePS Implementation Group2004]:

```
:  p(a) ⇒ q(a)

:  p(a) ⇒ r(a)

:  m(a)⇒ r(a)

:  ¬r(a)

:  ¬q(a)

:  m(a)
```

Then the system is asked if r(a) is derivable.

: **r(a)?**

When the NAND-set { `m(a) ⇒ r(a), ¬r(a), m(a)` } is detected, the user chooses to have `m(a)` removed from the base. No ordering assistance from the user is needed, because `m(a)` is the only culprit at this time. The interaction just described was deleted from this example, because it does not illustrate DDR, but it can be seen in the Appendix in Section A.4.[11]

The system is then told to add `p(a)` with forward inference.

: **p(a)!**

The forward inference detects two NAND-sets (actual NAND-set resolution output not shown here):

- { `p(a), p(a) ⇒ r(a), ¬r(a)` }
- { `p(a), p(a) ⇒ q(a), ¬q(a)` }

The user chooses to remove `¬r(a)`, placing it as more credible than `m(a)`. The user then removes `¬q(a)` and places it above `¬r(a)` in the ordering—this last interface was seen in the culprit ordering interface example shown in Section 6.2.7 on page 221.

A knowledge engineer can examine the elements of the knowledge state, and these elements currently show the value of the culprit beliefs. The final implementation will include a simpler set of commands for the user that will hide the belief values The next command shows the removed beliefs ($= X$).

: **(sneps::knowledge-state-removed-beliefs sneps::*default-ks*)**

`(wff9: ¬q(a)   wff8: ¬r(a)   wff6: m(a))`

When DDR is called, the system first prints out the starting elements in *ddr-q*. It then prints out feedback for each pass through the DDR loop—the belief that is being processed and its tag followed by *ddr-q*

[11] For a complete example of the SNePS NAND-set detection and culprit selection user interface, see [Shapiro2000].

after processing. Note how the processing of `¬r(a)` resulted in the tag for `m(a)` being changed from *justout* to *both*. In this example, the option to pause after each loop completion is exercised. DDR is continued following the processing of `¬q(a)`, but it is stopped by the user after the processing of `¬r(a)` (and before processing `m(a)`). This interaction is shown below.

```
:  (sneps::ddr sneps::*default-ks* t)
Starting ddr...  ddr-q (
wff9:  ¬q(a):  0.875 (tag:  justout)
wff8:  ¬r(a):  0.75 (tag:  justout)
wff6:  m(a):  0.5 (tag:  justout))


wff9:  ¬q(a)
sneps::justout End of ddr loop...  ddr-q (
wff8:  ¬r(a):  0.75 (tag:  justout)
wff6:  m(a):  0.5 (tag:  justout))


Continue running ddr?  (y or n) y


wff8:  ¬r(a)
sneps::justout End of ddr loop...  ddr-q (
wff6:  m(a):  0.5 (tag:  both))


Continue running ddr?  (y or n) n nil
```

At this point, DDR has been stopped. The knowledge state is improved over that when DDR was first called, even though the base is unchanged. This is because the top element on *ddr-q* is weaker than the element that topped it when DDR was first called. This is an example of the knowledge state being available at any time and the promise that it will always improve with each pass through the DDR loop. The fact that the beliefs already processed are stronger than the belief on the queue is an example of diminishing returns.

Even though *ddr-q* is not empty, the system can receive input and perform reasoning and contradiction handling. To show this, the user enters the following two beliefs:

```
: s(a)
: ¬s(a)
```

Following the detection of the NAND-set {`¬s(a), s(a)`}, the user selects `s(a)` as the culprit (so it is removed from the base), and places it between `¬r(a)` and `m(a)` in the ordering (not shown).

When DDR is re-called, the more credible `s(a)` is processed before `m(a)`. When `m(a)` is processed, though, it is returned to the base. DDR stops when *ddr-q* is empty.

```
: (sneps::ddr sneps::*default-ks* t)
Starting ddr... ddr-q (
wff11: s(a): 0.625 (tag: justout)
wff6: m(a): 0.5 (tag: both))


wff11: s(a)
sneps::justout End of ddr loop... ddr-q (
wff6: m(a): 0.5 (tag: both))
```

```
Continue running ddr?  (y or n)y


wff6:  m(a)

sneps::both

"After add remove" End of ddr loop...  ddr-q ()


Continue running ddr?  (y or n) y

Finished ddr...  ddr-q ()
```

By looking at the removed beliefs (the beliefs in the current exbase X), we can see that `m(a)` has been returned to the base, even though it remains a culprit belief. This is shown below.

```
:  (sneps::knowledge-state-removed-beliefs sneps::*default-ks*)

(wff11:  s(a)   wff9:  ¬q(a)   wff8:  ¬r(a))


:  (sneps::knowledge-state-culprits sneps::*default-ks*)

culprits (from strongest to weakest):  (

wff9:  ¬q(a):  0.875

wff8:  ¬r(a):  0.75

wff11:  s(a):  0.625

wff6:  m(a):  0.5)
```

All removed beliefs are JustifiedOut—their return to the base would make one of their IC-NAND-sets active. Therefore, the base is KS-optimal.

Note that the list of removed beliefs is *not* ordered by $\succeq$ *at this time*. This implementation decision was based on the assumption that most retracted beliefs *stay* retracted. If a large amount of movement is expected in and out of the base, however, a BST would be a better choice.

In the Appendix (Section A.5, page 289), there is an implemented DDR run that shows the SNePS computer system performing DDR (non-interactively) and duplicating DDR Example 4.4.1 from Chapter 4 (cf. Section 4.4 on page 164 and Figure 4.2).

Chapter 7

Conclusions

7.1 Reconsideration

Reconsideration is a belief base optimizing operation that eliminates operation order effects. Reconsideration improves the knowledge state by optimizing the belief base with the help of hindsight. The result is the return of previously retracted beliefs reminiscent of the Recovery postulate.

The optimal base is the base whose credibility bitmap is maximal among all *consistent* bases. Defining optimality by the maximal bitmap for a consistent base is a technique supported in belief revision research [Williams & Sims2000, Chopra, Georgatos, & Parikh2001, Booth *et al.*2005].

The optimizing results are similar in some ways to the results of belief liberation [Booth *et al.*2005] and mirror the effects of belief change operations on belief sequences [Chopra, Georgatos, & Parikh2001] as well as those experienced by performing reconsideration on a a linear ordering of beliefs in SATEN [Williams & Sims2000].

Reconsideration can be used to reformulate the Recovery postulate by first recognizing that Recovery is an axiom about a *pair* of belief change operations (contraction and expansion), as opposed to just a contraction axiom. By replacing the expansion operation in Recovery with the operation of optimized-addition (i.e., expansion followed by reconsideration), the new axiom of Optimized-Recovery is formed. This axiom not only can be expressed in terms of a belief space ($K \subseteq (K \sim p) +_{\cup!} p$), but also in terms of a belief base ($B \subseteq (B \sim p) +_{\cup!} p$). This is a a Recovery-like axiom that even belief *bases* can adhere to—as long as adherence is not dependent on the inconsistency caused by $B \cup \{p\} \vdash \perp$.

An axiom that is always satisfied is: $\neg a \notin B \Rightarrow B \subseteq (B +_{\cup!} \neg a) +_{\cup!} a$. This mirrors the Recovery-like iterated revision postulate in [Chopra, Ghose, & Meyer2002].

Using a SATEN demonstration that offers six separate adjustment strategies for belief revision of a base with a non-linear ordering (a total-preorder), reconsideration improved the base in five of the six cases. The improvement was optimal in three of those cases—three cases that also showed adherence to Optimized-Recovery. This indicates that reconsideration can be a useful tool even for systems that cannot fully meet the belief ordering restrictions recommended.

7.2 DDR

DDR is an efficient, anytime, TMS-friendly algorithm that performs reconsideration by improving the belief base in increments. It offers all the features of reconsideration with added benefits for implemented systems.

Implementing DDR in a TMS-style system allows optimization of the base in a computationally friendly way.

- DDR optimizes the most important parts of the base first and makes incremental improvements towards optimality.
- Each change made to the knowledge state by DDR truly is an *improvement*
- Each improvement is measurable.
- Each improvement is immediately available to the user for reasoning or belief change.

- DDR can yield to urgent reasoning demands—it can be stopped and restarted without loss of optimality upon completion.

- DDR can be performed in multiple stages that can interleave with the addition of new information.

There no longer needs to be a trade-off between having a consistent base ready for reasoning and having an optimal base (by delaying consistency maintenance until all information is gathered).

DDR provides an efficient method for re-optimizing a base if its beliefs are re-ordered. The DDR priority queue is a simple indicator of when optimization may be needed: when the queue is not empty. The DDR QImpact value provides a measure of the confidence in the current base and in derivations made from that base:

- Non-culprit beliefs are never retracted.

- Any culprit belief whose credibility is strictly stronger than the current QImpact value is in its optimal assignment regarding the base and exbase: i.e., it is in the base if and only if it will also be in the optimal base.

- Given a consistent base, reasoning from a subset of that base that contains no culprit belief whose credibility value is weaker than the QImpact value produces derivations that are entailed by the optimal base—they will survive optimality. A user can have full confidence in these beliefs.

- Reasoning using a culprit base belief with a credibility value that *is* weaker than the QImpact value will produce derivations that *might* be invalid in the optimal knowledge state, because their support *might* be retracted during DDR. These derivations should be flagged as suspect.

By using the DDR queue, a belief space (both base and derived beliefs) can be partitioned into high and low confidence designations that can be adjusted as the QImpact value changes.

7.3 DOBS

The formalization of a deductively-open belief space (DOBS) provides terminology that allows a clearer discussion of the shades of gray that implemented systems have regarding consistency—specifically the differences between the following levels of consistency:

1. being consistent (KS-consistency and consistency),[1]
2. being inconsistent, but not knowing it (KS-consistency and inconsistency), and
3. being inconsistent and knowing it (KS-inconsistency and inconsistency).

DDR performed to completion in an ideal system guarantees optimality and consistency. This is optimality at the first (top) level.

DDR performed in a DOBS can only guarantee KS-optimality—optimality at the second level—which recognizes that there may be currently unknown contradictions that make the current state non-optimal (because it is inconsistent). After a newly detected inconsistency negates KS-optimality, DDR can be used to re-optimize (still at level two, but a "wiser" level two).

DDR can be implemented in any TMS-style system, and the system can run as it normally does until the time to re-optimize is determined. It merely needs to maintain its priority queue and detected NAND-sets.

7.4 Implementation of DDR

7.4.1 Complexity and Anytime Issues

Pointers and counters added to data structures (for beliefs and NAND-sets) reduce the complexity of the algorithms used by the DDR process by eliminating searches within larger data structures and comparisons

[1] I always assume a sound reasoner, so the system can never be KS-inconsistent and consistent—though flaws in implementation or inelegant halting could possibly result in such a state.

between sets of beliefs. A global test-number counter reduces complexity and improves the recovery of a system when procedures are halted part way through. By assigning the global counter number to each element as it is dealt with, there is a way to determine if the contact with an element is a first contact or a repeated contact. This eliminates the need for group counter zeroing at the beginning of procedures, which reduces complexity when a procedure might access a small percentage of the elements in a group—zeroing all counters is prohibitively expensive. It also insures that updating information in a large group, if interrupted, has a record of which elements have been updated and which have not.

My algorithm for detecting subset/superset relationships between a belief set A and a collection of belief sets $\{B_1, B_2, \ldots, B_n\}$ improves the complexity of that operation. Traditional subset/superset testing involves handling many beliefs that are *not* shared between sets. My algorithm reduces the complexity to (worst case) the sum of the cardinality of the intersections: $\sum \mid A \cap B_i \mid$, for $1 \leq i \leq n$. Only intersecting beliefs are handled and no belief in a set is handled more than once.

7.4.2 Ordering Beliefs

Ordering beliefs can be done in several ways. Requiring the user to order the beliefs at the time they are entered into the base is the most cumbersome and invasive technique.

Ordering sources and categories of beliefs and associating each belief with a source and/or category allows the user to develop a pre-ordering on the beliefs without having to examine, compare and order individual beliefs. This works for sources of information (e.g., UAV, troops, INTEL,...) and for belief categories (e.g., rule, fact, sensor-reading, ...). If every NAND-set has a least element (i.e., each NAND-set spans categories but never has two beliefs in its weakest category), DDR can run without assistance. If two beliefs in the same category are being considered as culprits, further heuristics are needed or, as with our implementation, the user can be asked to select the culprit(s).

If the association to a belief credibility category or source is automated (e.g., the system recognizes the difference between a sensor reading and a rule or between sources providing information to the system), then much of the work involved in ordering the beliefs is eliminated. Categorizing each belief by hand is time-consuming and should be avoided.

Asking the user to order beliefs that are actually selected as culprits (selected by the user or some other agency) is a reasonable solution when categorizing is not an option, because it avoids ordering or categorizing *non-culprit* beliefs. When a new culprit is chosen (again, this must be by the user), it is placed within the ordering by the user.

When/How to Run DDR

DDR can be run after every belief change operation that involves base belief removal (optimized-additon; favoring optimality), during downtime of the system (when input/query volume is low; favoring usability), and/or upon user demand.

Because DDR alters the knowledge state that the user is familiar with, it is important to notify the user of changes to the base. As with any DOBS system, there is a tradeoff between optimality and usability. DDR changes/improvements can be implemented as found (favoring optimality), upon notification and acceptance by the user (user-friendly, but invasive), or at some predetermined time (typically downtime; e.g., update every night at 2:00 AM). Notifying the user of changes to the set of *derived* beliefs that are lost/regained due to DDR-based changes might also be helpful, but potentially extensive. Filtering for relevance to current projects may be helpful.

7.5 In Summary

Reconsideration can assist KR&R systems with hindsight base credibility improvement.

DDR provides many improvements to a KR&R system that detects NAND-sets by

- eliminating operation order side-effects with an efficient, anytime, KS-optimizing process;
- offering a measurement of credibility for a KS-inconsistent base and its deductively open belief space; and
- performing optimized adjustment to the continual influx of new information—whether it comes as as new input or newly derived beliefs and NAND-sets.

Chapter 8

Future Work

This chapter mentions areas and topics of further research that would build on the research presented in this dissertation.

8.1 Non-Linear Orderings

The research in this dissertation assumed that base beliefs could be ordered linearly. Future research should explore reconsideration and DDR for orderings that are non-linear. Specific issues should include

1. base beliefs in a partial order;
2. base beliefs in a total pre-order;
3. base beliefs in a pre-order that is neither antisymmetric nor comparable;
4. dealing with NAND-sets that do not have a unique weakest element.

It is possible to have (1), (2), or (3), without triggering (4). How best to define an optimal base and handle culprit selection in situation (4) remains an open question.

The situation mentioned in (4) was discussed in Sections 3.4.2 and 3.4.3 along with the concepts of credulous vs. skeptical reasoning. When presented with a non-singleton set *A* of equally viable culprits to choose from, the skeptical choice is to remove *all* beliefs in the set ($B' = B \setminus A$), and the credulous choice is to non-deterministically select only one belief in the set for removal ($B' = B \setminus \{p\}$, where $p \in A$). Future work should explore and define the benefits reconsideration (and DDR) can provide to a system that performs either skeptical or credulous reasoning. Let us assume that future knowledge acquisition and/or reasoning identifies a set $A' \subseteq A$ as no longer containing viable culprits:

- In a skeptical system, reconsideration would return the beliefs in A' to the base, provided there was no other reason preventing their return. This allows a system to play it safe by removing all possible culprits yet still recover those that are later deemed to be safe (in hindsight).
- In a credulous system, reconsideration (along with a new credulous selection) would swap the originally removed belief p with one of the less preferred beliefs in $A \setminus A'$—returning p to the base and removing some $q \in A \setminus A'$.

8.2 Multiple Contexts

The research in this dissertation assumes one $B^{\cup}$ and one base B that is desired to be as credible as possible. Future work should explore the possibility of maintaining a global $B^{\cup}$ (and, for DDR, a global I), but allowing multiple sub-contexts to be defined using subsets of $B^{\cup}$ (and, for DDR, I) along with possibly differing orderings. Reflective of multi-context reasoning in an ATMS, a NAND-set S that is discovered in one context would be added to the global I and made available to other contexts if and when S is a subset of their $B^{\cup}$ elements. Each context would maintain its own knowledge state tuple.

8.3 Contraction and Disbelief in a System Using Reconsideration

It would be interesting to explore the role that user-initiated contraction (i.e., contraction that is not done for consistency maintenance, only) can have in a system that performs reconsideration. Recall that traditional contraction of a belief from a base B is undone by reconsideration.

Contraction could be implemented as a removal from $B^{\cup}$. In this case, there is the issue of whether (for a DOBS-DDR system) to remove elements in I that contain the contracted belief. If not removed, elements in I are no longer subsets of $B^{\cup}$ as originally defined. If removed, then the resulting I is less informed that it was before, and the removed information would not be available if the contracted belief is reasserted in the future.

If maintaining subsets of $B^{\cup}$ for each context, user-initiated contraction could be performed on the context specific $B^{\cup}$ with corresponding changes to the context-specific I. If the contracted belief is later returned, the sets that were removed from the context-specific I would still be present in the global I for re-acquisition.

There could be different contraction operations for contracting a base belief (e.g., input error) and contracting a belief from the belief space (as with AGM or base contraction). Contractions might also be considered permanent—e.g., removing a belief and disallowing its return. Note that permanent contraction from a belief *space* should be compared to *disbelief*, and reasoning from such a contraction may be possible: e.g., if q is forbidden to return to the belief space and $p \rightarrow q$, then $\neg p$ may be derivable.

8.4 DDR-Specific Research Topics

8.4.1 DDR for Default Logic Systems

In a default logic system, the removal of a belief might allow a blocked default belief to return. A DDR-like algorithm might be useful for processing the order in which various blocked default beliefs are considered for possible return by focusing on the more important beliefs first as opposed to processing them in the order they became (or, more accurately, *may* have become) un-blocked. If a default logic system has a section of its base that contains monotonic beliefs, DDR could be helpful in maximizing the credibility of that section.

8.4.2 Postulates Constraining DDR Belief Change Operations

Operations on a DDR knowledge state were defined using kernel consolidation. Future research should develop postulates constraining the belief change operations performed on a DDR knowledge state. In addition to changes to the current base and the priority queue, these should also include operations that change other elements in the tuple.

8.4.3 Further Research on Knowledge State Preference Orderings

In this dissertation, the development and discussion of the DDR and DOBS-DDR knowledge states were focused on elements that affect the DDR algorithm and its anytime features. Future work should explore and define other aspects of these knowledge states. One key issue is how changes to I, $D^{\cup}$, and H should affect knowledge state preference ordering ($\succeq_{KS}$). Recall that the definitions in this dissertation held those elements constant.

Although $B^{\cup}$ was also held constant, it is impractical to compare two knowledge states with differing values for $B^{\cup}$. Comparing knowledge states with different orderings ($\succeq$) might also be possible provided the orderings are not inconsistent with each other.

8.5 DOBS-Specific Research Topics

Research is needed to explore the changes to $D^{\cup}$ and H in a DOBS that results from reasoning. This work should formalize the query process and forward inferencing of a DOBS that produce new derivations. This would be an extension of the work presented in [Johnson & Shapiro2000a].

As with the DDR knowledge state, future work should include DOBS-adjusted versions of the standard belief change postulates for belief sets (recalling that a DOBS can be implemented in a coherence version) and belief bases. This would be an extension of the work presented in [Johnson & Shapiro2001].

A currently unexplored area of DOBS research involves using the DOBS formalization to develop guidelines and benchmark tests for evaluating and comparing implemented systems. These descriptions can better identify the strengths and weaknesses of a system so that potential users can choose the strengths *and weaknesses* that best fit their needs.

8.6 Implementation-Specific Research Topics

The following list contains action items that are specific to implementing DDR and automated belief revision in SNePS:

- Implementing DDR in multiple contexts.
- Improving NAND-set detection. Explore other tests that can be added to improve inconsistency detection. Note: be aware of cost/benefit trade-offs.
- Providing the user with an interface (text or GUI) for altering a belief ordering.
- Providing a GUI interface to assist with culprit selection.
- Implementing sufficient heuristics to assist (or eliminate) the user during culprit selection.
  - Implementing DDR and automated belief change for reasoning and *acting* agents (robots, VR agents, etc).
  - Implementing an auto-ordering that develops a pre-order based on the user's culprit selections. This would replace or reduce the user having to manually insert a new culprit into an existing order. This requires meta-checking to insure there are no cycles in the ordering. It also requires redeveloping the DDR algorithms to work with a pre-order rather than a linear order—this affects the ordering of the priority queues and NAND-set queues.

– Assisting the user during culprit selection of a NAND-set by providing existing culprit ordering of *pertinent* beliefs. This would include any pre-order information about the beliefs in the NAND-set being processed.

Appendix A

Appendix: Proofs for Chapter 4

A.1 Theorem Proofs

A.1.1 Theorem 4.2.36

Given the DDR knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q \rangle$, Proper-Q($KS$), $\langle p, justout \rangle \in Q$, and $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1 \leftarrow_{ref}$ Change-Tag$(p, Q, both, \succeq)$ $\rangle$: Proper-Q(KS_1).

Proof.

1. $KS = \langle B, B^{\cup}, I, \succeq, Q \rangle$ (Premise)
2. Proper-Q(KS) (Premise)
3. $\langle p, justout \rangle \in Q$ (Premise)
4. $Q_1 =$ Change-Tag$(p, Q, both, \succeq)$ (Premise)
5. $\langle p, both \rangle \in Q_1$ (4,Def 4.2.30)
6. $(\forall q \neq p, \forall \tau) : \langle q, \tau \rangle \in Q \Leftrightarrow \langle q, \tau \rangle \in Q_1$ (4,Def 4.2.30)
7. Q_1 is a DDR priority queue (5,6,Def 4.2.1)
8. $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1 \rangle$ (Premise)

9. $B = B_1 \;\wedge\; B^{\cup} = B_1^{\cup} \;\wedge\; I = I_1 \;\wedge\; \succeq = \succeq_1 \;\wedge\; Q \neq Q_1 \quad \Rightarrow \quad X = X_1 \;\wedge$

$(\forall p)$: $\text{MustOut}(p, KS) \equiv \text{MustOut}(p, KS_1) \;\wedge\; \text{BadOut}(p, KS) \equiv \text{BadOut}(p, KS_1)$

$\wedge\;$ $\text{JustifiedOut}(p, KS) \equiv \text{JustifiedOut}(p, KS_1)$. (1,8,Obs 4.2.22)

10. Given a DDR knowledge state KS', Proper-Q(KS') $\equiv$

$\forall b:\;$ $\text{BadOut}(b, KS') \;\Rightarrow\; \text{Protected-by-Q}(b, KS')$. (Def 4.2.32)

11. $\forall b:\;$ $\text{BadOut}(b, KS) \;\Rightarrow\; \text{Protected-by-Q}(b, KS)$. (2,10)

12. Given a DDR knowledge state KS', Protected-by-Q(b, KS') $\equiv$

Q1 (**or** $\langle b, in? \rangle \in Q'$

Q2 $\langle b, both \rangle \in Q'$

Q3 $\exists S$ s.t (**and** $S \in I'$

Q4 $\text{Weakest}(S, \succeq) = b$

Q5 $(S \setminus \{b\}) \cap X' \neq \emptyset$

Q6 $\forall s \in ((S \setminus \{b\}) \cap X')$:

Q7 (**or** $\langle s, justout \rangle \in Q'$

Q8 $\langle s, both \rangle \in Q'$)))) (Def 4.2.31)

13. Let b be an arbitrary belief s.t. BadOut(b, KS_1).

14. BadOut(b, KS) (1,8,9,13)

15. Protected-by-Q(b, KS) (11,14)

16. $(b = p) \vee (b \neq p)$ (tautology)

17. If $b = p$, then $\langle b, both \rangle \in Q_1$ (5,12)

18. If $b = p$, then Protected-by-Q(b, KS_1) (12,17)

19. $\langle b, in? \rangle \in Q \quad \vee \quad \langle b, both \rangle \in Q \quad \vee \quad (\exists S):$

$(S \in I \quad \wedge \quad \text{Weakest}(S, \succeq) = b \quad \wedge \quad (S \setminus \{b\}) \cap X \neq \emptyset \quad \wedge$

$(\forall s \in ((S \setminus \{b\}) \cap X) : \quad (\langle s, justout \rangle \in Q \quad \vee \quad \langle s, both \rangle \in Q \quad))).$ (12,15)

20. If $b \neq p$, then $\langle b, in? \rangle \in Q \Leftrightarrow \langle b, in? \rangle \in Q_1$ (6)

21. If $b \neq p$, then $\langle b, both \rangle \in Q \Leftrightarrow \langle b, both \rangle \in Q_1$ (6)

22. $(\forall s \neq p) : (\langle s, justout \rangle \in Q \;\vee\; \langle s, both \rangle \in Q) \Rightarrow (\langle s, justout \rangle \in Q_1 \;\vee\; \langle s, both \rangle \in Q_1)$ (6)

23. $(\forall s = p) : (\langle s, justout \rangle \in Q \;\vee\; \langle s, both \rangle \in Q) \Rightarrow (\langle s, justout \rangle \in Q_1 \;\vee\; \langle s, both \rangle \in Q_1)$ (3,5)

24. $(\forall s) : (\langle s, justout \rangle \in Q \;\vee\; \langle s, both \rangle \in Q) \Rightarrow (\langle s, justout \rangle \in Q_1 \;\vee\; \langle s, both \rangle \in Q_1)$ (22,23)

25. If $b \neq p$, then $\langle b, in? \rangle \in Q_1 \quad \vee \quad \langle b, both \rangle \in Q_1 \quad \vee \quad (\exists S) :$

$(S \in I \quad \wedge \quad \text{Weakest}(S, \succeq) = b \quad \wedge \quad (S \setminus \{b\}) \cap X \neq \emptyset \quad \wedge$

$(\forall s \in ((S \setminus \{b\}) \cap X) : \quad (\langle s, justout \rangle \in Q_1 \quad \vee \quad \langle s, both \rangle \in Q_1 \quad))).$ (19,20,21,24)

26. If $b \neq p$, then Protected-by-Q(b, KS_1) (1,8,12,25)

27. Protected-by-Q(b, KS_1) (16,18,26)

28. $(\forall b) : BadOut(b, KS_1) \Rightarrow$ Protected-by-Q(b, KS_1) (13,27)

29. Proper-Q(KS_1) (10 ,28) □

A.1.2 Theorem 4.2.37

Given the DDR knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$, Proper-Q($KS$), $p \in X$, $p \notin Q$, and $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1 \leftarrow_{ref} \text{Insert}(\langle p, \tau\rangle, Q, \succeq)\rangle$, where $\tau \in \{justout, in?, both\}$: Proper-Q($KS_1$).

Proof.

1. $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ (Premise)
2. Proper-Q(KS) (Premise)
3. $p \in X$ (Premise)
4. $p \notin Q$ (Premise)
5. $Q_1 = \text{Insert}(\langle p, \tau\rangle, Q, \succeq)\rangle$, where $\tau \in \{justout, in?, both\}$ (Premise)
6. Q_1 is a DDR priority queue (3,5,Def 4.2.1,Def 4.2.29)
7. $p \in Q_1$ (5,Def of Insert: Def 4.2.29)
8. $(\forall q, \forall \tau) : \langle q, \tau\rangle \in Q \Rightarrow \langle q, \tau\rangle \in Q_1$ (5,Def of Insert: Def 4.2.29)
9. $KS_1 = \langle B, B^{\cup}, I, \succeq, Q_1\rangle$ (Premise)
10. $B = B_1 \;\wedge\; B^{\cup} = B_1^{\cup} \;\wedge\; I = I_1 \;\wedge\; \succeq = \succeq_1 \;\wedge\; Q \neq Q_1 \quad \Rightarrow \quad X = X_1 \;\wedge$

    $(\forall p)$: MustOut(p, KS) $\equiv$ MustOut(p, KS_1) $\wedge$ BadOut(p, KS) $\equiv$ BadOut(p, KS_1)

    $\wedge$ JustifiedOut(p, KS) $\equiv$ JustifiedOut(p, KS_1) . (1,9,Obs 4.2.22)
11. Given a DDR knowledge state KS', Proper-Q(KS') $\equiv$

    $\forall b$: BadOut(b, KS') $\Rightarrow$ Protected-by-Q(b, KS'). (Def 4.2.32)
12. $\forall b$: BadOut(b, KS) $\Rightarrow$ Protected-by-Q(b, KS). (2,11)
13. Given a DDR knowledge state KS', Protected-by-Q(b, KS') $\equiv$

    Q1 (**or** $\langle b, in?\rangle \in Q'$

    Q2 $\langle b, both\rangle \in Q'$

    Q3 $\exists S$ s.t (**and** $S \in I'$

Q4 $\quad$ Weakest$(S, \succeq) = b$

Q5 $\quad (S \setminus \{b\}) \cap X' \neq \emptyset$

Q6 $\quad \forall s \in ((S \setminus \{b\}) \cap X')$:

Q7 $\quad$ (**or** $\langle s, justout \rangle \in Q'$

Q8 $\quad \langle s, both \rangle \in Q'$)))) $\quad$ (Def 4.2.31)

14. Let b be an arbitrary belief s.t. BadOut(b, KS_1).

15. $\quad$ BadOut(b, KS) $\quad$ (1,9,10,14)

16. $\quad$ Protected-by-Q(b, KS) $\quad$ (12,15)

17. $\quad \langle b, in? \rangle \in Q \quad \vee \quad \langle b, both \rangle \in Q \quad \vee \quad (\exists S):$

$\quad$ ($S \in I \quad \wedge \quad$ Weakest$(S, \succeq) = b \quad \wedge \quad (S \setminus \{b\}) \cap X \neq \emptyset \quad \wedge$

$\quad (\forall s \in ((S \setminus \{b\}) \cap X): \quad (\langle s, justout \rangle \in Q \quad \vee \quad \langle s, both \rangle \in Q$))). $\quad$ (13,16)

18. $\quad \langle b, in? \rangle \in Q \Rightarrow \langle b, in? \rangle \in Q_1$ $\quad$ (8)

19. $\quad \langle b, both \rangle \in Q \Rightarrow \langle b, both \rangle \in Q_1$ $\quad$ (8)

20. $\quad (\forall s): (\langle s, justout \rangle \in Q \;\vee\; \langle s, both \rangle \in Q) \Rightarrow (\langle s, justout \rangle \in Q_1 \;\vee\; \langle s, both \rangle \in Q_1)$ $\quad$ (8)

21. $\quad \langle b, in? \rangle \in Q_1 \quad \vee \quad \langle b, both \rangle \in Q_1 \quad \vee \quad (\exists S):$

$\quad$ ($S \in I \quad \wedge \quad$ Weakest$(S, \succeq) = b \quad \wedge \quad (S \setminus \{b\}) \cap X \neq \emptyset \quad \wedge$

$\quad (\forall s \in ((S \setminus \{b\}) \cap X): \quad (\langle s, justout \rangle \in Q_1 \quad \vee \quad \langle s, both \rangle \in Q_1$))). $\quad$ (17,18,19,20)

22. $\quad$ Protected-by-Q(b, KS_1) $\quad$ (1,9,13,21)

23. $(\forall b): BadOut(b, KS_1) \;\Rightarrow\;$ Protected-by-Q(b, KS_1) $\quad$ (14,22)

24. Proper-Q(KS_1) $\quad$ (11 ,23) □

A.1.3 Theorem 4.2.40

Given the DDR knowledge states $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$ and $KS_1 = \langle B_1, B^{\cup}, I, \succeq, Q_1\rangle$, where Proper-Q($KS$), $p \in X,\ p \in B_1,\ p \notin Q_1$ and $(\forall q \neq p, \forall \tau) : (q \in B \equiv q \in B_1) \wedge (\langle q, \tau\rangle \in Q \equiv \langle q, \tau\rangle \in Q_1)$; then following statement is true: Proper-Q(KS_1).

Proof.

1. Proper-Q(KS) (premise)
2. $p \in X$ (premise)
3. $p \in B_1$ (premise)
4. $p \notin Q_1$ (premise)
5. $(\forall q \neq p) : (q \in B \equiv q \in B_1)$ (premise)
6. $(\forall q \neq p, \forall \tau) : (\langle q, \tau\rangle \in Q \equiv \langle q, \tau\rangle \in Q_1)$ (premise)
7. Given any knowledge state $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$,

   Proper-Q(KS) $\equiv (\forall b)$: BadOut(b, KS) $\Rightarrow$ Protected-by-Q(b, KS). (Def 4.2.32)
8. $(\forall b)$: BadOut(b, KS) $\Rightarrow$ Protected-by-Q(b, KS). (1,7)
9. Protected-by-Q(b, KS') $\equiv$

   Q1 (**or** $\langle b, in?\rangle \in Q'$

   Q2 $\langle b, both\rangle \in Q'$

   Q3 $\exists S$ s.t (**and** $S \in I'$

   Q4 Weakest($S, \succeq$) $= b$

   Q5 $(S \setminus \{b\}) \cap X' \neq \emptyset$

   Q6 $\forall s \in ((S \setminus \{b\}) \cap X')$:

   Q7 (**or** $\langle s, justout\rangle \in Q'$

   Q8 $\langle s, both\rangle \in Q'$)))) (Def 4.2.31)

10. Let b be an arbitrary belief s.t. BadOut$(b, KS) \wedge b \neq p$.

11. Protected-by-Q(b, KS). (8,10)

12. $X_1 =_{def} B^{\cup} \setminus B_1$ (premise,Def 4.2.1)

13. Given any KS, BadOut$(b, KS) \equiv ((b \in X) \wedge (\neg \text{MustOut}(b, KS)))$. (Def 4.2.19)

14. $\neg$BadOut(p, KS_1) (3,13)

15. $b \in X_1$ (5,10)

16. $\langle b, in?\rangle \in Q \equiv \langle b, in?\rangle \in Q_1$ (6,10)

17. $\langle b, both\rangle \in Q \equiv \langle b, both\rangle \in Q_1$ (6,10)

18. $((\langle b, in?\rangle \notin Q \wedge \langle b, both\rangle \notin Q) \Rightarrow$

Q3 $(\exists S$ s.t (**and** $S \in I$

Q4 Weakest$(S, \succeq) = b$

Q5 $(S \setminus \{b\}) \cap X \neq \emptyset$

Q6 $\forall s \in ((S \setminus \{b\}) \cap X)$:

Q7 (**or** $\langle s, justout\rangle \in Q$

Q8 $\langle s, both\rangle \in Q$))). Take such an S. (9,11)

19. $p \notin S \Rightarrow (\forall s \in ((S \setminus \{b\}) \cap X_1) \neq \emptyset : \langle s, justout\rangle \in Q_1 \vee \langle s, both\rangle \in Q_1)$ (5,6,18)

20. $(S \setminus \{b, p\}) \cap X \neq \emptyset \Rightarrow (\forall s \in ((S \setminus \{b\}) \cap X_1) \neq \emptyset :$

$(\forall s \in ((S \setminus \{b\}) \cap X_1) \neq \emptyset : \langle s, justout\rangle \in Q_1 \vee \langle s, both\rangle \in Q_1)$ (3,6,18)

21. $(S \setminus \{b, p\}) \cap X = \emptyset \Rightarrow (S \setminus \{b\}) \subseteq B_1 = \emptyset$ (3)

22. If $S \in I$, then Culprit$(S, \succeq) =_{def}$ Weakest$(S, \succeq)$. (Def 4.2.6)

23. $b =$ Culprit$(S, \succeq)$. (18,22)

24. Given any $KS = \langle B, B^{\cup}, I, \succeq, Q\rangle$,

MustOut$(b, KS) \equiv (\exists S \in I) : S \setminus \{b\} \subseteq B$ and $b =$ Culprit$(S, \succeq)$. (Def 4.2.18)

25. $(S\setminus\{b,p\})\cap X \;=\; \emptyset \;\Rightarrow \text{MustOut}(b,KS_1)$ (21,23,24)

26. $(S\setminus\{b,p\})\cap X \;=\; \emptyset \;\Rightarrow \neg\text{BadOut}(b,KS_1)$ (13,25)

27. $\neg\text{BadOut}(b,KS_1) \vee$

$(\forall s\in\; ((S\setminus\{b\})\cap X_1)\neq\emptyset):\; \langle s,\mathit{justout}\rangle\in Q_1 \vee \langle s,\mathit{both}\rangle\in Q_1$ (19,20,26)

28. $(\langle b,\mathit{in?}\rangle\notin Q \wedge \langle b,\mathit{both}\rangle\notin Q) \;\Rightarrow\; (\neg\text{BadOut}(b,KS_1)\vee$

Q3 $(\exists S$ s.t (**and** $S\in I$

Q4 $\text{Weakest}(S,\succeq) = \; b$

Q5 $(S\setminus\{b\})\cap X_1 \;\neq\; \emptyset$

Q6 $\forall s\in\; ((S\setminus\{b\})\cap X_1)$:

Q7 (**or** $\langle s,\mathit{justout}\rangle\in Q_1$

Q8 $\langle s,\mathit{both}\rangle\in Q_1$)))). (18,27)

29. $\langle b,\mathit{in?}\rangle\in Q_1 \vee \langle b,\mathit{both}\rangle\in Q_1 \;\vee\; \neg\text{BadOut}(b,KS_1)\vee$

Q3 $(\exists S$ s.t (**and** $S\in I$

Q4 $\text{Weakest}(S,\succeq) = \; b$

Q5 $(S\setminus\{b\})\cap X_1 \;\neq\; \emptyset$

Q6 $\forall s\in\; ((S\setminus\{b\})\cap X_1)$:

Q7 (**or** $\langle s,\mathit{justout}\rangle\in Q_1$

Q8 $\langle s,\mathit{both}\rangle\in Q_1$)))). (16,17,28)

30. $\neg\text{BadOut}(b,KS_1)\vee \text{Protected-by-Q}(b,KS_1)$ (9,29)

31. $(\forall b)$: $(\text{BadOut}(b,KS)\wedge b\neq p)\Rightarrow(\neg\text{BadOut}(b,KS_1)\vee\text{Protected-by-Q}(b,KS_1))$ (10,30)

32. $(\forall b)$: $(\neg\text{BadOut}(b,KS))\vee(b=p)\vee(\neg\text{BadOut}(b,KS_1))\vee(\text{Protected-by-Q}(b,KS_1))$ (31)

33. $b=p\Rightarrow\; \neg\text{BadOut}(b,KS_1)$ (14)

34. $(\forall b)$: $\neg\text{BadOut}(b,KS)\vee\neg\text{BadOut}(b,KS_1)\vee\text{Protected-by-Q}(b,KS_1)$ (32,33)

35. Given any KS, $(\forall b): \neg\mathrm{BadOut}(b, KS) \equiv (b \in B) \vee (\mathrm{MustOut}(b, KS))$. (13)

36. $(\forall b): b \in B \Rightarrow b \in B_1$ (2,5)

37. $(\forall b): b \in B \Rightarrow \neg\mathrm{BadOut}(b, KS_1)$ (35,36)

38. $(\forall b): (S \setminus \{b\} \subseteq B) \Rightarrow (S \setminus \{b\} \subseteq B_1)$ (3,5)

39. $(\forall b): \mathrm{MustOut}(b, KS) \Rightarrow \mathrm{MustOut}(b, KS_1)$ (24,38)

40. $(\forall b): \mathrm{MustOut}(b, KS) \Rightarrow \neg\mathrm{BadOut}(b, KS_1)$ (35,39)

41. $(\forall b): \neg\mathrm{BadOut}(b, KS) \Rightarrow \neg\mathrm{BadOut}(b, KS_1)$ (35,37,40)

42. $(\forall b): \neg\mathrm{BadOut}(b, KS_1) \vee$ Protected-by-Q(b, KS_1) (34,41)

43. $(\forall b): \mathrm{BadOut}(b, KS_1) \Rightarrow$ Protected-by-Q(b, KS_1) (42)

44. Proper-Q(KS_1) (7,43) □

A.2 Helper-Functions: Theorems and Proofs

A.2.1 Theorem and Proof for Safe-Return (p, KS): Theorem 4.3.1

Theorem 4.3.1 Given KS and p and the following:

- Preconditions for Safe-Return(p, KS): $KS = KS_{pre}$ is a DDR knowledge state containing the elements $B, B^{\cup}, I, \succeq$, and Q; Proper-KS(KS_{pre}); BadOut(p, KS_{pre}); $\langle p, \tau \rangle =$ First(Q_{pre}), where $\tau \in \{in?, both\}$.
- $R =$ Safe-Return(p, KS).

then the following postconditions hold:

- Postconditions for Safe-Return(p, KS): Returns a set of beliefs R ; $KS = KS_{post} = KS_{pre}$; Proper-KS($KS$); BadOut($p, KS$); $R \subseteq B$; $(\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$; $\sum_{p_i \in ((B+p) \setminus R)}$Cred$(p_i, B^{\cup}, \succeq) >$ $\sum_{p_j \in B}$Cred$(p_j, B^{\cup}, \succeq)$; $(\forall r \in R,) :\ p \succ r$; $(\forall r \in R, \exists S \in I) : r =$ Culprit$(S, \succeq) \ \wedge \ S \subseteq (((B+p) \setminus R) + r)$.

Proof.

function Safe-Return(p, KS) ;;; returns a set of beliefs R

;;; Proper-KS(KS) KS is never altered; Proper-KS(KS) holds throughout; Call this **axiom1**.

$\langle B, B^{\cup}, I, \succeq,\ Q \rangle \leftarrow_{ref} KS$

;;; $p \in Q$; $\therefore\ p \in X$ (Preconditions; Def 4.2.1)

;;; Safe-per-I(KS) (Def 4.2.33)

;;;; $(\forall S \in I) : \neg$Active(S, KS) (Def 4.2.16)

;;; $\therefore (\forall S \in I) : p \in S \ \vee \ S \not\subseteq (B+p)$ (Def 4.2.12); Call this **axiom2**

$R \leftarrow \{\ \}$;;; initializing R

;;; $R \subseteq B; R \neq \emptyset \Rightarrow ((\forall r \in R) : p \succ r \ \wedge \ (((B+p) \setminus R) + r) \vdash \perp)$ holds vacuously

$NQ \leftarrow_{ref} NAND\text{-}set\text{-}Q$(NC-NAND-sets$(p, KS)$, $\succeq$) ;;; NQ= p's NC-NAND-sets in a queue

;;; $\{S \mid S \in NQ\} = \{S \mid (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))\}$ (Def 4.2.1, Def 4.2.8,Def 4.2.11)

;;; $\therefore (\forall S \in NQ) : (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))$

;;; $(\forall S \in NQ) : p \succ \text{Culprit}(S, \succeq)$ (Def 4.2.4, Def 4.2.6)

;;; $NQ_{start} \leftarrow NQ$

;;; $\{S \mid S \in NQ_{start}\} = \{S \mid (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))\}$ call this **axiom3**

;;; $(\forall S_j, S_{j+1} \in NQ) : \text{Culprit}(S_j, \succeq) \succeq \text{Culprit}(S_{j+1}, \succeq)$ for $1 \leq j < n$. (Def 4.2.8)

;;; $R \subseteq B; R \neq \emptyset \Rightarrow ((\forall r \in R) : p \succ r \wedge (((B + p) \setminus R) + r) \vdash \perp)$ still holds

;;; $UNS \leftarrow \{\ \}$;;; for reference in this proof; UNS for Unsafe NAND-sets

;;; $(\forall S \in UNS) : S \in I$ (holds vacuously)

loop until Empty(NQ)

;;; $\neg$Empty(NQ); $NQ_{top} \leftarrow NQ$; $R_{top} \leftarrow R$ (inside **loop**)

;;; $R \subseteq B; R \neq \emptyset \Rightarrow ((\forall r \in R) : p \succ r \wedge (((B + p) \setminus R) + r) \vdash \perp)$ (still holds)

;;; $(\forall S \in NQ) : (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))$ (still holds)

;;; $\therefore (\forall S \in NQ_{top}) : (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))$

;;; $(\forall S \in UNS) : S \in I$ (still holds)

$N \leftarrow_{ref}$ Pop(NQ) ;;; First element in NQ removed into N

;;; NQ = Rest(NQ_{top}); N =First(NQ_{top}); $N \in NQ_{top}$; $N \notin NQ$ (Def 4.2.29)

;;; $(N \in I) \wedge (p \in N) \wedge (p \succ \text{Culprit}(N, \succeq))$ ($N \in NQ_{top}$; Def 4.2.29)

;;; $R = R_{top}$; $R \subseteq B$; $R \neq \emptyset \Rightarrow ((\forall r \in R) : p \succ r \wedge (((B + p) \setminus R) + r) \vdash \perp)$ (still holds)

;;; $(\forall r \in R) : (r \succeq \text{Culprit}(N, \succeq))$ (ordering of NQ_{top})

;;; $(\forall S \in NQ) : (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))$ (NQ = Rest(NQ_{top}))

;;; $(\forall S \in UNS) : S \in I$ (still holds)

if $(N \setminus \{p\} \subseteq B$ AND $N \cap R = \emptyset)$ **then**

;;; $(N \in I) \wedge (p \in N) \wedge (p \succ \text{Culprit}(N, \succeq))$ (still holds)

;;; $\therefore p \neq \text{Culprit}(N, \succeq) \wedge \text{Culprit}(N, \succeq) \in B$ (preceding two lines)

;;; $N \setminus \{p\} \subseteq B$; $N \cap R = \emptyset$; (inside **if**)

;;; $\therefore N \subseteq ((B + p) \setminus R)$

;;; MustOut(Culprit$(N, \succeq)$, $\langle (B \cup \{p\}) \setminus R$, $B^{\cup}$, I, $\succeq$, $Q \rangle$) (Def 4.2.18)

;;; $R = R_{top}$; $R \subseteq B$; $R \neq \emptyset \Rightarrow ((\forall r \in R) : p \succ r \wedge (((B + p) \setminus R) + r) \vdash \perp)$ (still holds)

;;; $(\forall r \in R) : r \succ \text{Culprit}(N, \succeq)$ (now, $r \succ \text{Culprit}(N, \succeq)$, because $N \cap R = \emptyset$)

;;; $(\forall S \in NQ) : (S \in I) \wedge (p \in S) \wedge (p \neq \text{Culprit}(S, \succeq))$ (still holds)

;;; $(\forall S \in UNS) : S \in I$ (still holds)

;;; $(\forall S \in UNS) : \text{Culprit}(N, \succeq) \notin S$ (ordering of NQ_{top})

$R \leftarrow_{ref} R \cup \{\text{Culprit}(N, \succeq)\}$;;; culprit for N inserted into R

;;; $R \neq \emptyset$; $R \neq R_{top}$; $R_{top} \subsetneq R$; $R = (R_{top} + \text{Culprit}(N, \succeq))$;

;;; $\therefore N \cap R = \{\text{ Culprit}(N, \succeq)\}$ ($N \cap R_{top} = \emptyset$; previous comment)

;;; $R \subseteq B$; $(\forall r \in R) : p \succ r$ (Culprit$(N, \succeq) \in B$; $p \succ \text{Culprit}(N, \succeq)$)

;;; $N \cap R \neq \emptyset$; $N \not\subseteq ((B \cup \{p\}) \setminus R)$

;;; $UNS \leftarrow_{ref} UNS \cup N$

;;; $(\forall S \in UNS) : S \in I$ (still holds, because $N \in I$)

;;; $(\forall S \in UNS) : S \cap R = \{\text{Culprit}(S, \succeq)\}$ (ordering of NQ_{top})

;;; $(\forall S \in UNS) : S \setminus ((B + p) \setminus R) = \text{Culprit}(S, \succeq)$ ($S \setminus \{p\} \subseteq B$, from **if** condition)

;;; $(\forall r \in R, \exists S \in UNS) : (S \in I) \wedge r = \text{Culprit}(S, \succeq) \ldots$ (Preceding comments)

;;; $\ldots \wedge S \subseteq (((B + p) \setminus R) + r)$ (Preceding comments)

;;; $(\forall r \in R, \exists S \in I) : r = \text{Culprit}(S, \succeq) \wedge S \subseteq (((B + p) \setminus R) + r)$ (Preceding comment)

;;; $\therefore (\forall r \in R) : (((B + p) \setminus R) + r) \vdash \perp$ (Obs 4.2.13)

;;; $(\forall S \in NQ) : (S \in I) \ \wedge \ (p \in S) \ \wedge \ (p \neq \text{Culprit}(S, \succeq))$ (still holds)

endif

;;; $N \in NQ_{start}$; $N \notin NQ$ (still holds)

;;; $N \not\subseteq ((B \cup \{p\}) \setminus R)$ (either failed **if** condition, or Culprit$(N, \succeq) \in R$)

;;; $(\forall S \in \{S \mid S \in NQ_{start}\} \setminus \{S \mid S \in NQ\}) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (Preceding comments)

;;; $R \subseteq B$; $(\forall r \in R) : p \succ r$ (still holds)

;;; $(\forall r \in R) : (((B + p) \setminus R) + r) \vdash \perp$ (still holds)

;;; $(\forall S \in NQ) : (S \in I) \ \wedge \ (p \in S) \ \wedge \ (p \neq \text{Culprit}(S, \succeq))$ (still holds)

;;; $(\forall S \in UNS) : S \in I$ (still holds)

;;; $(\forall r \in R, \exists S \in I) : r = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (((B + p) \setminus R) + r)$ (still holds)

end loop

;;; Proper-KS(KS) **axiom1**

;;; KS is unchanged since the beginning

;;; BadOut(p, KS); (Precondition: still holds)

;;; $\therefore (\nexists S \in I) : p = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (B + p)$ (Def 4.2.18 ;Def 4.2.19;)

;;; $\therefore (\nexists S \in I) : p = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (B + p) \setminus R$ (set theory); call this **safe1**

;;; $\therefore (\forall S \in I) : p \in S \ \vee \ S \not\subseteq (B + p)$ **axiom2**

;;; $\therefore (\nexists S \in I) : p \notin S \ \wedge \ S \subseteq (B + p) \setminus R$ (set theory); call this **safe2**

;;; Empty(NQ) (out of **loop**)

;;; $(\forall S \in \{S \mid S \in NQ_{start}\} \setminus \{S \mid S \in NQ\}) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (still holds)

;;; $\{S \mid S \in NQ_{start}\} \setminus \{S \mid S \in NQ\} = \{S \mid S \in NQ_{start}\}$ (Empty(NQ))

;;; $\{S \mid S \in NQ_{start}\} = \{S \mid (S \in I) \ \wedge \ (p \in S) \ \wedge \ (p \neq \text{Culprit}(S, \succeq))\}$ **axiom3**

;;; $\therefore (\nexists S \in I) : (p \in S) \ \wedge \ (p \neq \text{Culprit}(S, \succeq)) \ \wedge \ S \subseteq ((B \cup \{p\}) \setminus R)$ call this **safe3**

;;; $\therefore$ $(\nexists S \in I) : S \subseteq ((B \cup \{p\}) \setminus R)$ (**safe1**;**safe1**;**safe3**)

;;; $R \subseteq B$; $(\forall r \in R) : p \succ r \ \wedge \ (((B+p) \setminus R) + r) \vdash \perp$ (still holds)

;;; $\therefore$ $\sum_{p_i \in ((B+p)\setminus R)} \mathrm{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \mathrm{Cred}(p_j, B^{\cup}, \succeq)$

;;; $(\forall r \in R, \exists S \in I) : r = \mathrm{Culprit}(S, \succeq) \ \wedge \ S \subseteq (((B+p) \setminus R) + r)$ (still holds)

return R $\square$

A.2.2 Theorem and Proof for KS-Add-Remove(KS, p, R): Theorem 4.3.2

Theorem 4.3.2 Given KS, p, R and the following

- Preconditions for KS-Add-Remove(KS, p, R): $KS = KS_{pre}$ is a DDR knowledge state;

  Proper-KS(KS_{pre}); BadOut(p, KS_{pre}); $\sum_{p_i \in ((B+p)\setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$;

  $R \subseteq B_{pre}$; $\langle p, \tau \rangle =$ First(Q_{pre}), where $\tau \in \{in?, both\}$; $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$; and

  $(\forall r \in R) : p \succ r$, and $(\exists S \in I_{pre}) : r = \text{Culprit}(S, \succeq_{pre}) \wedge S \subseteq (((B_{pre} + p) \setminus R) + r)$.

- KS_{post} is the knowledge state resulting from KS-Add-Remove(KS, p, R)

then the following postconditions hold:

- Postconditions for KS-Add-Remove(KS, p, R): KS_{post} is a DDR knowledge state; Proper-KS(KS_{post});

  $B_{post} = (B_{pre} \cup \{p\}) \setminus R$; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; $(\forall r \in R)$: JustifiedOut(r, KS_{post});

  QImpact(KS_{pre})$>$ QImpact(KS_{post}) ; $B_{post} \succ B_{pre}$; $KS_{post} \succ_{KS} KS_{pre}$; $p \notin Q_{post}$; $(\forall r \in R) : p \succ r \wedge \langle r, justout \rangle \in Q_{post}$; $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau \rangle \in Q_{pre}) \equiv (\langle b, \tau \rangle \in Q_{post}))$.

Proof.

procedure KS-Add-Remove(KS, p, R),

$\langle\, B\,,\, B^{\cup}\,,\, I\,,\, \succeq\,,\, Q \,\rangle \leftarrow_{ref} KS$

;;; $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau \rangle \in Q_{pre}) \equiv (\langle b, \tau \rangle \in Q))$ (no changes, yet)

$Q_1 \leftarrow_{ref}$ New-Q( ) ;;; A new priority queue unrelated to KS

for each r in R **do**

;;; $r \notin Q_1$

$Q_1 \leftarrow_{ref}$ Insert($\langle r, justout \rangle$, Q_1 , $\succeq$) $\rangle$

;;; $\langle r, justout \rangle \in Q_1$ (Def of Insert: Def 4.2.29)

;;; $(\forall q \neq r) : \langle r, \tau \rangle \in Q_1 \Leftrightarrow \langle r, \tau \rangle \in Q_1$ (Def 4.2.29)

;;; $(\forall q, \tau) : \langle q, \tau \rangle \in Q_1 \Rightarrow (q \in R) \wedge \tau = justout$

endfor

;;; $(\forall q, \tau) : \langle q, \tau \rangle \in Q_1 \Leftrightarrow (q \in R) \land \tau = justout$ (still holds; for loop results)

;;; $\therefore (\forall r \in R) : \langle r, justout \rangle \in Q_1$

;;; $KS = KS_{pre}$ (no changes, yet)

;;; $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau \rangle \in Q_{pre}) \equiv (\langle b, \tau \rangle \in Q))$ (no changes, yet)

;;; Proper-KS(KS) (Precondition; still holds)

;;; BadOut(p, KS) (Precondition; still holds)

;;; $R \subseteq B$ (Precondition; still holds)

;;; $\therefore (\forall r \in R) : r \notin Q$ (Def 4.2.1)

;;; $\therefore (\forall r \in R) : \langle r, justout \rangle \in$Merge-Qs$(Q_{pre}, Q_1, \succeq)$ (Def 4.2.30)

;;; $\langle p, \tau \rangle =$ First(Q_{pre}), where $\tau \in \{in?, both\}$ (Precondition; still holds)

;;; $(\forall r \in R) : \; p \succ r$ (Precondition; still holds)

;;; $\therefore \langle p, \tau \rangle =$ First(Merge-Qs$(Q_{pre}, Q_1, \succeq)$), where $\tau \in \{in?, both\}$ (Def 4.2.29,Def 4.2.30)

;;; $\therefore (\forall r \in R) : \langle r, justout \rangle \in$Popped(Merge-Qs$(Q_{pre}, Q_1, \succeq)$) (Def 4.2.29)

$KS \leftarrow_{ref} \langle \, B \leftarrow_{ref} (B \cup \{p\}) \setminus R, \; B^{\cup} \, , \; I \, , \; \succeq \; , \; Q \leftarrow_{ref}$ Popped(Merge-Qs$(Q, Q_1, \succeq)$)$\rangle$

;;; $p \notin Q$; $(\forall q \in Q) : p \succ q$ (Def of Popped: Def 4.2.29)

;;; QImpact$(KS_{pre}) >$ QImpact(KS)) (Def 4.2.43)

;;; $(\forall b \notin (R \cup \{p\})) : ((\langle b, \tau \rangle \in Q_{pre}) \equiv (\langle b, \tau \rangle \in Q))$ (Def of Popped: Def 4.2.29)

;;; $B^{\cup} = B^{\cup}_{pre}$; $I = I_{pre}$; $\succeq = \succeq_{pre}$

;;; $B = (B_{pre} \cup \{p\}) \setminus R$

;;; $\therefore p \in B$ ($p \notin Q$ adheres to Sec 4.2.12 on semi-revision without re-ordering)

;;; $(\forall r \in R) : \; p \succ r$ (still holds)

;;; $\therefore \sum_{p_i \in ((B+p) \setminus R)}$Cred$(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B}$Cred$(p_j, B^{\cup}, \succeq)$

;;; $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$ (Precondition; still holds)

;;; $\therefore (\forall S \in I) : S \not\subseteq B$

;;; $\therefore$ Safe-per-I(KS) (Def 4.2.12,Def 4.2.16)

;;; $Q =$ Popped(Merge-Qs$(Q_{pre}, Q_1, \succeq)$)

;;; $(\forall r \in R, \exists S \in I_{pre}) : r =$ Culprit$(S, \succeq_{pre}) \ \wedge\ S \subseteq (((B_{pre} + p) \setminus R) + r)$ (Precondition; still holds)

;;; $\therefore (\forall r \in R, \exists S \in I) : r =$ Culprit$(S, \succeq) \ \wedge\ S \subseteq (B + r)$

;;; $\therefore (\forall r \in R) : B + r \vdash \perp$ (Obs 4.2.13))

;;; $\therefore (\forall r \in R)$: MustOut$(r, KS)$ (Def 4.2.18)

;;; $(\forall r \in R) : r \notin B$ ($B = (B_{pre} \cup \{p\}) \setminus R$; adheres to $(KS_{pre} +_! p)$ Def 4.2.12)

;;; $(\forall r \in R) : \langle r, justout \rangle \in$ Popped(Merge-Qs$(Q_{pre}, Q_1, \succeq)$) (still holds)

;;; $\therefore (\forall r \in R) : \langle r, justout \rangle \in Q$ (follows queue maintenance for $(KS_{pre} +_! p)$ Def 4.2.12)

;;; Proper-Q(KS) (Thm 4.2.40,Thm 4.2.41,Thm 4.2.42)

;;; and $KS \succ_{KS} KS_{pre}$ (Def 4.2.23; Def 4.2.24); Def 4.2.45)

;;; Proper-KS(KS) (Def 4.2.33) $\square$

A.2.3 Theorem and Proof for Process-Justout(KS,p): Theorem 4.3.3

Theorem 4.3.3 Given KS and p and the following

- Preconditions for Process-Justout (KS,p): $KS = KS_{pre}$ is a DDR knowledge state; Proper-KS(KS_{pre}); JustifiedOut(p,KS_{pre}); and $\langle p,\tau\rangle =$ First(Q_{pre}), where $\tau \in \{justout, both\}$;
- KS_{post} is the knowledge state resulting from Process-Justout (KS,p);

then the following postconditions hold:

- Postconditions for Process-Justout (KS,p): KS_{post} is a DDR knowledge state; Proper-KS(KS_{post}) ; $B_{post} = B_{pre}$; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; QImpact(KS_{pre})> QImpact(KS_{post}) ; $KS_{post} \succ_{KS} KS_{pre}$; Q_{post} resembles Q_{pre} with the following changes:
  - $p \notin Q_{post}$, and
  - ($\forall N \in$NC-NAND-sets$(p,KS), \exists q =$Culprit$(N,\succeq)$): $q \in X \Rightarrow \langle q, in?\rangle \in Q_{post} \vee \langle q, both\rangle \in Q_{post}$

  Note: $(\forall N \in$ NC-NAND-sets$(p,KS), \forall b \in (X \setminus \{p\}), \forall\tau)$: $b \neq$Culprit$(N,\succeq) \Rightarrow (((\langle b,\tau\rangle \in Q_{pre}) \equiv (\langle b,\tau\rangle \in Q_{post}))$.

Proof.

procedure Process-Justout (KS,p)

$\langle B, B^{\cup}, I, \succeq, Q\rangle \leftarrow_{ref} KS$

;;; $(\forall x \in Q) : x \neq p \Rightarrow p \succ x$ (p is in first pair on Q)

;;; JustifiedOut(p,KS) (given)

;;; Proper-KS(KS) (given)

;;; Therefore Safe-per-I(KS) and Proper-Q(KS) (Def 4.2.33)

for each $N \in$ NC-NAND-sets(p,KS)

$q \leftarrow_{ref}$ Culprit(N)

;;; $p \succ q$ and $N \in I$ (Def 4.2.11)

;;; JustifiedOut(p, KS) (still holds)

if $q \in X$ **do**

;;; Proper-Q(KS) (still holds)

;;; JustifiedOut(p, KS) (still holds)

;;; $(\forall x \in Q) : x \neq p \Rightarrow p \succ x$ (still holds)

;;; $p \succ q$ (still holds)

$Q \leftarrow_{ref}$ DDR-Q-Insert(q, KS) ;;; Destructively alters Q (and KS)

;;; $(\langle q, in? \rangle \in Q \ \vee \ \langle q, both \rangle \in Q)$ (Defn of DDR-Q-Insert: Def 4.2.30)

;;; $p \succ q$ (still holds)

;;; $(\forall x \in Q) : x \neq p \Rightarrow p \succ x$ (still holds, because $p \succ q$)

;;; Proper-Q(KS) (Thm 4.2.38)

;;; JustifiedOut(p, KS) (still holds)

endif

;;; JustifiedOut(p, KS) (still holds)

;;; $q \in X \Rightarrow \langle q, in? \rangle \in Q \vee \langle q, both \rangle \in Q.$

endfor

;;; $(\forall x \in Q) : x \neq p \Rightarrow p \succ x$ (still holds)

;;; Therefore, $\langle p, \tau \rangle =$ First(Q) still holds

;;; Proper-Q(KS) (still holds)

;;; JustifiedOut(p, KS) (still holds)

;;; $(\forall N \in$NC-NAND-sets$(p, KS), \exists q =$Culprit$(N, \succeq)$:

$q \in X \Rightarrow \langle q, in? \rangle \in Q \vee \langle q, both \rangle \in Q.$ (still holds)

;;; $(\forall N \in$ NC-NAND-sets$(p, KS), \forall b \in X, \forall \tau)$: $b \neq$Culprit$(N, \succeq) \Rightarrow$

$$((\langle b,\tau\rangle \in Q) \equiv (\langle b,\tau\rangle \in Q)) \quad \text{(only culprits of NC-NAND-sets}(p,KS)\text{ altered in }Q)$$

$KS \leftarrow_{ref} \langle\ B\ ,\ B^{\cup}\ ,\ I\ ,\ \succeq\ ,\ \text{Popped(Q)}\ \rangle$

;;; $p \notin Q$; $(\forall q \in Q) : p \succ q$ (Def of Popped: Def 4.2.29)

;;; $\text{QImpact}(KS_{pre}) > \text{QImpact}(KS))$ (Def 4.2.43)

;;; Proper-Q(KS) (Thm 4.2.39)

;;; Safe-per-I(KS) (B and I unchanged, Def 4.2.16)

;;; Proper-KS(KS) (Def 4.2.33)

;;; and $KS \succ_{KS} KS_{pre}$ (Def 4.2.23; Def 4.2.45)

;;; $(\forall N \in$NC-NAND-sets$(p,KS), \exists q =$Culprit$(N,\succeq)$:

$$q \in X \Rightarrow \langle q, in?\rangle \in Q \vee \langle q, both\rangle \in Q. \quad \text{(because } p \neq q\text{, still holds)}$$

;;; $(\forall N \in$ NC-NAND-sets$(p,KS), \forall b \in X \setminus \{p\}, \forall\tau) :\ b \neq$Culprit$(N,\succeq) \Rightarrow$

$$((\langle b,\tau\rangle \in Q_{pre}) \equiv (\langle b,\tau\rangle \in Q_{post})) \quad \text{(definition of Popped: Def 4.2.29)} \quad \square$$

A.2.4 Theorem and Proof for Update-KS$(KS, update, p, R)$: Theorem 4.3.4

Theorem 4.3.4 Given KS, $update, p$, and R and the following

- Preconditions for Update-KS $(KS, update, p, R)$: $KS = KS_{pre}$ is a DDR knowledge state;

  Proper-KS(KS_{pre}); $\langle p, \tau \rangle =$ First(Q_{pre}); $update \in \{PopQ, ProcJustout, AddRem\}$, and

  - If $update = PopQ$: JustifiedOut(p, KS_{pre}); and $\tau = in?$.
  - If $update = ProcJustout$: JustifiedOut(p, KS_{pre}); and $\tau \in \{justout, both\}$.
  - If $update = AddRem$: BadOut(p, KS_{pre}); $R \subseteq B_{pre}$; $\tau \in \{in?, both\}$;

    $\sum_{p_i \in ((B+p) \setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$;

    $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$; and

    $(\forall r \in R) : p \succ r$, and $(\exists S \in I) : r = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (((B_{pre} + p) \setminus R) + r)$;

- KS_{post} is the knowledge state resulting from Update-KS $(KS, update, p, R)$;

then the following postconditions hold:

- Postconditions for Update-KS $(KS, update, p, R)$: KS_{post} is a DDR knowledge state; $KS_{post} \succ_{KS} KS_{pre}$;

  $B^{\cup}_{post} = B^{\cup}_{pre}; I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; QImpact$(KS_{pre}) >$QImpact$(KS_{post})$; Proper-KS$(KS_{post})$.

  - If $update = PopQ$: $B_{post} = B_{pre}$ and $Q_{post} =$Popped(Q_{pre}) , therefore, $p \notin Q_{post}$.
  - If $update = ProcJustout$: $B_{post} = B_{pre}$ and Q_{post} resembles Q_{pre} with the following changes:

    (1) $p \notin Q_{post}$ and (2) $(\forall N \in$NC-NAND-sets$(p, KS), \exists q =$Culprit$(N, \succeq))$: $q \in X \Rightarrow (\langle q, in? \rangle \in Q_{post} \vee \langle q, both \rangle \in Q_{post})$. Regarding unaltered elements of the queue:

    $(\forall N \in$ NC-NAND-sets$(p, KS), \forall b \in X \setminus \{p\}, \forall \tau)$:

    $$b \neq \text{Culprit}(N, \succeq) \Rightarrow \ ((\langle b, \tau \rangle \in Q_{pre}) \equiv (\langle b, \tau \rangle \in Q_{post})).$$

  - If $update = AddRem$: $B_{post} = (B_{pre} \cup \{p\}) \setminus R$; $B_{post} \succ B_{pre}$;

    $(\forall r \in R) : p \succ r \wedge$ JustifiedOut(r, KS_{post}); and Q_{post} resembles Q_{pre} with the following changes:

    * $p \notin Q_{post}$;
    * $(\forall r \in R)$: $\langle r, justout \rangle \in Q_{post}$.

Proof.

procedure Update-KS $(KS, update, p, R)$

;;; $KS = KS_{pre}$ and contain the elements B_{pre} , $B^{\cup}$, I , $\succeq$, and Q_{pre} (precondition)

;;; Proper-KS(KS_{pre}) (precondition)

;;; Proper-Q(KS_{pre}) (Def 4.2.33)

;;; $\langle p, \tau \rangle =$ First(Q_{pre}) (precondition)

Case *update* =

PopQ:

;;; Proper-KS(KS_{pre}) (still holds)

;;; Therefore, Safe-per-I(KS_{pre}) (Def 4.2.33)

;;; JustifiedOut(p, KS_{pre}) (precondition)

;;; $\tau = in$? (precondition)

$KS = KS_{post} \leftarrow_{ref} \langle$ B , $B^{\cup}$, I , $\succeq$, Popped(Q) $\rangle$

;;; Proper-Q(KS_{post}) (Thm 4.2.39)

;;; QImpact(KS_{pre})$>$ QImpact(KS_{post}); (Thm 4.2.44)

;;; Safe-per-I(KS_{post}) (B, $B^{\cup}$, and I unchanged; Def 4.2.16)

;;; Proper-KS(KS_{post}) (Def 4.2.33)

;;; $KS_{post} \succ_{KS} KS_{pre}$ ($B_{post} = B_{pre}$;Def 4.2.45)

ProcJustout:

;;; Proper-KS(KS_{pre}) (still holds)

;;; JustifiedOut(p, KS_{pre} (precondition)

;;; $\tau \in \{justout, both\}$ (precondition)

Process-Justout(KS, p)

;;; Proper-KS(KS_{post}) (Thm 4.3.3)

;;; QImpact(KS_{pre})> QImpact(KS_{post}); (Thm 4.3.3)

;;; $B_{post} = B_{pre}$; ${B^{\cup}}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; (Thm 4.3.3)

;;; $p \notin Q_{post}$ (Thm 4.3.3)

;;; $(\forall N \in$NC-NAND-sets$(p, KS), \exists q =$Culprit$(N, \succeq))$: $q \in X \Rightarrow$

$$\langle q, in?\rangle \in Q_{post} \vee \langle q, both\rangle \in Q_{post} \qquad \text{(Thm 4.3.3)}$$

;;; $(\forall N \in$ NC-NAND-sets$(p, KS), \forall b \in X \setminus \{p\}, \forall \tau)$: $b \neq$Culprit$(N, \succeq) \Rightarrow$

$$((\langle b, \tau\rangle \in Q_{pre}) \equiv (\langle b, \tau\rangle \in Q_{post})) \qquad \text{(Thm 4.3.3)}$$

;;; $KS_{post} \succ_{KS} KS_{pre}$ (Thm 4.3.3)

AddRem:

;;; Proper-KS(KS_{pre}) (still holds)

;;; BadOut(p, KS_{pre}); $R \subseteq B_{pre}$; $\tau \in \{in?, both\}$; (preconditions)

;;; $\sum_{p_i \in ((B+p) \setminus R)}$Cred$(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B}$Cred$(p_j, B^{\cup}, \succeq)$; (precondition)

;;; $(\forall S \in I) : S \not\subseteq ((B_{pre} + p) \setminus R)$; (precondition)

;;; $(\forall r \in R) : p \succ r$, and $(\exists S \in I) : r =$ Culprit$(S, \succeq)\ \wedge$

$$S \subseteq (((B_{pre} + p) \setminus R) + r); \qquad \text{(precondition)}$$

KS-Add-Remove(KS, p, R)

;;; Proper-KS(KS_{post}) (Thm 4.3.2)

;;; QImpact(KS_{pre})> QImpact(KS_{post}); (Thm 4.3.2)

;;; $B_{post} = (B_{pre} \cup \{p\}) \setminus R$; (Thm 4.3.2)

;;; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; (Thm 4.3.2)

;;; $p \notin Q_{post}$ (Thm 4.3.2)

;;; $(\forall r \in R)$: $\langle r, justout\rangle \in Q_{post}$ (Thm 4.3.2)

;;; $(\forall r \in R) : p \succ r \;\wedge\;$ JustifiedOut(r, KS_{post}) (Thm 4.3.2)

;;; $KS_{post} \succ_{KS} KS_{pre}$ (Thm 4.3.2) $\square$

A.3 Theorem and Proof for the DDR Algorithm: Theorem 4.3.6

Theorem 4.3.6 Given the DDR knowledge state *KS* and the following:

- Preconditions for DDR(*KS*): $KS = KS_{pre}$ is a DDR knowledge state; Proper-KS(KS_{pre}).
- KS_{post} is the knowledge state resulting from calling DDR(*KS*) and running it to completion.

then the following conditions hold:

- Postconditions for DDR(*KS*): KS_{post} is a DDR knowledge state; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post}=\succeq_{pre}$; $I =$ Complete-I(KS_{pre}) $\equiv$ $I =$ Complete-I(KS_{post}); Proper-KS(KS_{post}); Empty(Q_{post}); $KS_{post} \succeq KS_{pre}$.
- Loop conditions for DDR(*KS*): Let KS_{top} be the DDR knowledge state at the top of the DDR loop (just after line 1). And let KS_{bot} be the DDR knowledge state that results from KS_{top} being processed by the DDR loop (just after line 18). For each pass through the DDR loop: Proper-KS(KS_{top}) and Proper-KS(KS_{bot}); QImpact(KS_{top})> QImpact(KS_{bot}); $KS_{bot} \succ_{KS} KS_{top}$. Additionally, if $\langle p, \tau \rangle$ = First(Q_{top}) and $B_{top} \neq B_{bot}$, then
  - $(B' \setminus B) = \{p\}$;
  - $(\forall r \in (B \setminus B'), \exists N \in I) : r =$Culprit$(N, \succeq) \wedge p \succ r$.

Proof.

procedure DDR(*KS*)

;;; Proper-KS(*KS*). (precondition)

;;; Proper-Q(*KS*). (Def 4.2.33)

;;; Safe-per-I(*KS*) (Def 4.2.33)

;;; $KS = KS_{pre}$ (no changes yet)

1 **loop until** Empty(Q_{KS})

;;; $\neg$Empty(Q_{KS}) (satisfies loop condition)

;;; Proper-KS(KS). (still holds)

;;; $KS = KS_{top}$ (KSinside top of DDR loop)

$\langle p, \tau \rangle \leftarrow_{ref}$ First(Q_{KS})

;;; $\langle p, \tau \rangle =$ First(Q_{KS}) (from assignment)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (Def 4.2.1,Def 4.2.29)

;;; $p \in X$ (Def 4.2.1)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

if$_1$ ($\tau = in?$ or $\tau = both$) , **then**

;;; $\tau \in \{in?, both\}$ (satisfies condition)

;;; $\langle p, \tau \rangle =$ First(Q_{KS}) (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

can-return $\leftarrow$ Bad-Out(p, KS)

;;; *can-return* $\equiv$ Bad-Out(p, KS) (from assignment)

;;; $\langle p, \tau \rangle =$ First(Q_{KS}), where $\tau \in \{in?, both\}$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

if$_2$ *can-return* , **then**

;;; BadOut(p, KS) (satisfies condition, *can-return* $\equiv$ Bad-Out(p, KS))

;;; $\langle p, \tau \rangle = \text{First}(Q_{KS})$, where $\tau \in \{in?, both\}$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau \in \{in?, both\} \ \wedge$ BadOut(p, KS)

$R \ \leftarrow_{ref} \ \text{Safe-Return}(p, KS)$

;;; Proper-KS(KS); BadOut(p, KS); $R \subseteq B$; (Thm 4.3.1)

;;; $KS = KS_{top}$ (no change, yet: Thm 4.3.1)

;;; $(\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (Thm 4.3.1)

;;; $(\forall r \in R,) : \ p \succ r$ (Thm 4.3.1)

;;; $(\forall r \in R, \exists S \in I) : r = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (((B + p) \setminus R) + r)$ (Thm 4.3.1)

;;; $\sum_{p_i \in ((B+p) \setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$ (Thm 4.3.1)

;;; $\langle p, \tau \rangle = \text{First}(Q_{KS})$, where $\tau \in \{in?, both\}$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; Preceding comments constrained by: $\tau \in \{in?, both\} \ \wedge$ BadOut(p, KS)

$update \leftarrow AddRem$

;;; $update = AddRem$ (from assignment)

;;; $KS = KS_{pre}$; Proper-KS(KS); BadOut(p, KS); $R \subseteq B$; (still hold)

;;; $(\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (still hold)

;;; $(\forall r \in R,) : \ p \succ r$ (still hold)

;;; $(\forall r \in R, \exists S \in I) : r = \text{Culprit}(S, \succeq) \;\wedge\; S \subseteq (((B+p) \setminus R) + r)$ (still hold)

;;; $\sum_{p_i \in ((B+p) \setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$ (still holds)

;;; $\langle p, \tau \rangle = \text{First}(Q_{KS})$, where $\tau \in \{in?, both\}$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Preceding comments constrained by: $\tau \in \{in?, both\} \;\wedge\;$ BadOut(p, KS)

8 **else**$_2$

;;; JustifiedOut$(p.KS)$ (failed **if**$_2$ condition at line 5; Obs 4.2.21)

;;; $\langle p, \tau \rangle = \text{First}(Q_{KS})$, where $\tau \in \{in?, both\}$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau \in \{in?, both\} \;\wedge\;$ JustifiedOut(p, KS)

9 **if**$_3$ $\tau = both$, **then**

;;; $\tau = both$ (satisfies condition)

;;; JustifiedOut$(p.KS)$ (still holds)

;;; $\langle p, both \rangle = \text{First}(Q_{KS})$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau = both \ \wedge$ JustifiedOut(p, KS)

10 $update \leftarrow ProcJustout$

;;; $update = ProcJustout$ (from assignment)

;;; JustifiedOut$(p.KS)$ (still holds)

;;; $\langle p, both \rangle =$ First(Q_{KS}) (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau = both \ \wedge$ JustifiedOut(p, KS)

11 **else**$_3$

;;; $\tau = in?$ (fails **if**$_3$ condition at line 9)

;;; JustifiedOut$(p.KS)$ (still holds)

;;; $\langle p, in? \rangle =$ First(Q_{KS}) (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau = in? \ \wedge$ JustifiedOut(p, KS)

12 $update \leftarrow PopQ$

;;; $update = PopQ$ (from assignment)

;;; JustifiedOut$(p.KS)$ (still holds)

;;; $\langle p, in? \rangle =$ First(Q_{KS}) (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Proper-KS(KS). (still holds)

;;; Preceding comments constrained by: $\tau = in? \wedge$ JustifiedOut(p, KS)

endif$_3$

;;; $\tau = in? \Rightarrow \langle p, in? \rangle = \text{First}(Q_{KS})$ (from **if/else**$_3$)

;;; $\tau = in? \Rightarrow update = PopQ$ (from **if/else**$_3$)

;;; $\tau = both \Rightarrow update = ProcJustout$ (from **if/else**$_3$)

;;; $\tau = both \Rightarrow \langle p, both \rangle = \text{First}(Q_{KS})$ (from **if/else**$_3$)

;;; JustifiedOut($p.KS$) (holds for $\tau = in? \vee \tau = both$)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (holds for $\tau = in? \vee \tau = both$)

;;; $p \in X$ (holds for $\tau = in? \vee \tau = both$)

;;; $KS = KS_{top}$ (holds for all conditions, so far)

;;; Proper-KS(KS). (holds for $\tau = in? \vee \tau = both$)

;;; Preceding comments constrained by: $\tau \in \{in?, both\} \wedge$ JustifiedOut(p, KS)

endif$_2$

;;; JustifiedOut($p.KS$) $\wedge \tau = both \Rightarrow update = ProcJustout$ (from **if/else**$_2$)

;;; JustifiedOut($p.KS$) $\wedge \tau = in? \Rightarrow update = PopQ$ (from **if/else**$_2$)

;;; BadOut(p, KS) $\Rightarrow update = AddRem$ (from **if/else**$_2$)

;;; BadOut(p, KS) $\Rightarrow R \subseteq B$; (from **if/else**$_2$)

;;; BadOut(p, KS) $\Rightarrow (\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (from **if/else**$_2$)

;;; BadOut(p, KS) $\Rightarrow (\forall r \in R,) : \ p \succ r$ (from **if/else**$_2$)

;;; BadOut$(p, KS) \Rightarrow (\forall r \in R, \exists S \in I)$:

;;; $r = \text{Culprit}(S, \succeq) \wedge S \subseteq (((B+p) \setminus R) + r)$ (from **if/else**$_2$)

;;; BadOut$(p, KS) \Rightarrow$

;;; $\sum_{p_i \in ((B+p) \setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$ (due to $(\forall r \in R) : p \succ r$)

;;; $\tau = in? \Rightarrow \langle p, in? \rangle = \text{First}(Q_{KS})$ (JustifiedOut$(p.KS) \vee$ BadOut$(p.KS)$)

;;; $\tau = both \Rightarrow \langle p, both \rangle = \text{First}(Q_{KS})$ (JustifiedOut$(p.KS) \vee$ BadOut$(p.KS)$)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (holds for all conditions, so far)

;;; $p \in X$ (holds for all conditions, so far)

;;; $KS = KS_{top}$ (holds for all conditions, so far)

;;; Proper-KS(KS). (holds for all conditions, so far)

;;; Preceding comments constrained by: $\tau \in \{in?, both\}$

15 **else**$_1$;;; note: $\tau = justout$

;;; $\tau = justout$ (failed **if**$_1$ condition on line 3; Def 4.2.1)

;;; $\langle p, \tau \rangle = \text{First}(Q_{KS})$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; Therefore, $\neg$Protected-by-Q(p, KS) (Def 4.2.31; not protected in any way)

;;; $p \in X$ (still holds)

;;; Proper-KS(KS). (still holds)

;;; Therefore, Proper-Q(KS) (Def 4.2.33)

;;; Proper-Q$(KS) \equiv (\forall p)$: BadOut$(p, KS) \Rightarrow$ Protected-by-Q(p, KS). (Def 4.2.32)

;;; Therefore, JustifiedOut(p, KS) (Obs 4.2.21)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Preceding comments constrained by: $\tau = justout$

$update \leftarrow ProcJustout$

;;; $update = ProcJustout$ (from assignment)

;;; $\langle p, justout \rangle = \text{First}(Q_{KS})$ (still holds)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (still holds)

;;; $p \in X$ (still holds)

;;; Proper-KS(KS). (still holds)

;;; JustifiedOut(p, KS) (still holds)

;;; $KS = KS_{top}$ (no change during loop, yet)

;;; Preceding comments constrained by: $\tau = justout$

endif$_1$

;;; $\tau = justout \Rightarrow update = ProcJustout$ (from **if/else**$_1$)

;;; JustifiedOut($p.KS$) $\wedge\ \tau = both \Rightarrow update = ProcJustout$ (from **if/else**$_1$)

;;; JustifiedOut($p.KS$) $\wedge\ \tau = in? \Rightarrow update = PopQ$ (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow update = AddRem$ (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow R \subseteq B$; (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow (\forall S \in I) : S \not\subseteq ((B \cup \{p\}) \setminus R)$ (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow (\forall r \in R,) : \ p \succ r$ (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow (\forall r \in R, \exists S \in I) :$

;;; $r = \text{Culprit}(S, \succeq) \ \wedge \ S \subseteq (((B + p) \setminus R) + r)$ (from **if/else**$_1$)

;;; BadOut(p, KS) $\Rightarrow \sum_{p_i \in ((B+p)\setminus R)} \text{Cred}(p_i, B^{\cup}, \succeq) > \sum_{p_j \in B} \text{Cred}(p_j, B^{\cup}, \succeq)$ (frm **if/else**$_1$)

;;; $\tau = in? \Rightarrow \langle p, in? \rangle = \text{First}(Q_{KS})$ (from **if/else**$_1$)

;;; $\tau = both \Rightarrow \langle p, both \rangle = \text{First}(Q_{KS})$ (from **if/else**$_1$)

;;; $\tau = justout \Rightarrow \langle p, justout \rangle = \text{First}(Q_{KS})$ (from **if/else**$_1$)

;;; $\tau = justout \Rightarrow$ JustifiedOut(p, KS) (from **if/else**$_1$)

;;; $(\forall q \in Q) : q \neq p \Rightarrow p \succ q$ (holds for all conditions)

;;; $p \in X$ (holds for all conditions)

;;; $KS = KS_{top}$ (holds for all conditions)

;;; Proper-KS(KS). (holds for all conditions)

;;; Preceding comments satisfy the preconditions for Update-KS$(KS, update, p, R)$

18 Update-KS$(KS, update, p, R)$;;; DESTRUCTIVE—alters the DDR knowledge state KS

;;; $p \notin Q_{post}$ (Thm 4.3.4)

;;; QImpact$(KS_{top})>$ QImpact(KS_{bot}) (Thm 4.3.4)

;;; $KS_{bot} \succ_{KS} KS_{top}$ (Thm 4.3.4)

;;; $B^{\cup}_{bot} = B^{\cup}_{top}$; $I_{bot} = I_{top}$; $\succeq_{bot} = \succeq_{top}$; (Thm 4.3.4)

;;; Proper-KS(KS). (Thm 4.3.4)

;;; If $B_{top} \neq B_{bot}$, then

;;; $(B' \setminus B) = \{p\}$;

;;; $(\forall r \in (B \setminus B'), \exists N \in I) : r =$Culprit$(N, \succeq) \wedge p \succ r$. (Cor 4.3.5)

19 **end loop**

;;; Empty(Q) (loop condition in line 1 is no longer satisfied)

;;; $B^{\cup}_{post} = B^{\cup}_{pre}$; $I_{post} = I_{pre}$; $\succeq_{post} = \succeq_{pre}$; (never change during loop)

;;; $I =$ Complete-I$(KS_{pre}) \equiv I =$ Complete-I(KS_{post}) (I and $B^{\cup}$ unchanged; Def 4.2.3)

;;; For chapter 5: $I =$ KS-Complete-I$(KS_{pre}) \equiv I =$ KS-Complete-I(KS_{post}) (Def 5.5.1)

;;; $\neg$Empty$(Q_{pre}) \Rightarrow KS_{post} \succ KS_{pre}$ one loop pass guarantees improvement

;;; $KS_{post} \succeq KS_{pre}$ $KS_{post} = KS_{pre}$ if Empty(Q_{pre})

;;; Proper-KS(KS). (still holds) $\square$

A.4 SNePS Belief Revision Interaction

The following code is an actual excerpt from the small DDR example demonstration run that was presented in Section 6.4. This section covers the full interaction form the time the user asks the system if it can derive `r(a)` until the system completes its exploration regarding the derivability of `r(a)`. For a complete example of the SNePS NAND-set detection and culprit selection user interface, see [Shapiro2000].

Input is in **`bold`** text type. Output is non-bold `text` type.

```
:  r(a)?


I wonder if r(a)
holds within the BS defined by context default-defaultct


I know it is not the case that r(a)


I know m(a)


Since m(a) => r(a)
and m(a)
I infer r(a)


A contradiction was detected within context default-defaultct.
```

The contradiction involves the newly derived proposition:

r(a)

and the previously existing proposition:

¬r(a)

You have the following options:

1. [C]ontinue anyway, knowing that a contradiction is derivable;

2. [R]e-start the exact same run in a different context which is not inconsistent;

3. [D]rop the run altogether.

(please type c, r or d) =><= **r**

In order to make the context consistent you must delete at least one hypothesis from each of the following sets of hypotheses:

(wff8 wff7 wff6)

In order to make the context consistent you must delete at least one hypothesis from the set listed below.

An inconsistent set of hypotheses:

1 : ¬r(a)

```
(1 supported proposition:  (wff8) )

2 :  m(a) => r(a)

(2 supported propositions:  (wff7 wff4) )

3 :  m(a)

(2 supported propositions:  (wff6 wff4) )


Enter the list number of a hypothesis to examine or

 [d] to discard some hypothesis from this list,

 [a] to see ALL the hypotheses in the full context,

 [r] to see what you have already removed,

 [q] to quit revising this set, or

 [i] for instructions

(please type a number OR d, a, r, q or i) =><= 3


m(a)


WFFS that depend on wff6:

m(a)

r(a)


What do you want to do with hypothesis wff6?

[d]iscard from the context, [k]eep in the context,

[u]ndecided, [q]uit revising this set, [i]nstructions
```

```
(please type d, k, u, q or i) =><= d

The consistent set of hypotheses:

1 :  ¬r(a)
(1 supported proposition:  (wff8) )
2 :  m(a) => r(a)
(2 supported propositions:  (wff7 wff4) )

Enter the list number of a hypothesis to examine or
 [d] to discard some hypothesis from this list,
 [a] to see ALL the hypotheses in the full context,
 [r] to see what you have already removed,
 [q] to quit revising this set, or
 [i] for instructions

(please type a number OR d, a, r, q or i) =><= q

The hyps before removal are:
(wff8:  ¬r(a) wff7:  m(a) => r(a) wff6:  m(a))
The hyps after making them consistent are:
(wff8:  ¬r(a) wff7:  m(a) => r(a))
```

The following (not known to be inconsistent) set of

hypotheses was also part of the context where the

contradiction was derived:

(wff9 wff5 wff3)

Do you want to inspect or discard some of them? =><= **n**

Do you want to add a new hypothesis? =><= **n**

I wonder if r(a)

holds within the BS defined by context default-defaultct

I know it is not the case that r(a)

I wonder if m(a)

holds within the BS defined by context default-defaultct

I wonder if p(a)

holds within the BS defined by context default-defaultct

I know it is not the case that m(a)

¬r(a)

:

A.5 Large DDR Example 4.4.1 Implemented in SNePS

The following run shows the SNePS computer system performing DDR (non-interactively) and duplicating DDR Example 4.4.1 from Chapter 4 (cf. Section 4.4 on page 164 and Figure 4.2).

The starting state of the system is equivalent to KS_1 except that only the culprit beliefs are linearly ordered. The steps that the system goes through when performing DDR are identical to those in the example.

Input is in **`bold`** text type. Output is non-bold `text` type.

The printout below has three parts: (1) showing that the system is in the state described as KS_1 in Example 4.4.1; (2) adding `¬p` and performing belief revision to eliminate the contradiction; and (3) running DDR (non-interactively) to re-optimize the belief state.

Showing the system is at a state equivalent to KS_1

```
:  show-nand-sets
all-ns:
nand-set7:  (wff24:  ¬k(a) wff15:  w(a) => k(a)
wff11:  w(a)), (culprit:  wff24:  ¬k(a)) (size:  3) (count:  2)
nand-set5:  (wff23:  ¬t(a) wff10:  s(a) => t(a)
wff8:  s(a)), (culprit:  wff23:  ¬t(a)) (size:  3) (count:  2)
nand-set10:  (wff22:  ¬v(a) wff18:  z(a) => v(a)
wff17:  z(a)), (culprit:  wff17:  z(a)) (size:  3) (count:  2)
nand-set3:  (wff21:  ¬r(a) wff7:  m(a) => r(a)
wff6:  m(a)), (culprit:  wff6:  m(a)) (size:  3) (count:  2)
nand-set6:  (wff22:  ¬v(a) wff13:  w(a) => v(a)
wff11:  w(a)), (culprit:  wff22:  ¬v(a)) (size:  3) (count:  2)
```

```
nand-set8:  (wff22:  ¬v(a) wff16:  p(a) => v(a)

wff1:  p(a)), (culprit:  wff22:  ¬v(a)) (size:  3) (count:  2)

nand-set2:  (wff21:  ¬r(a) wff5:  p(a) => r(a)

wff1:  p(a)), (culprit:  wff21:  ¬r(a)) (size:  3) (count:  2)

nand-set1:  (wff20:  ¬q(a) wff3:  p(a) => q(a)

wff1:  p(a)), (culprit:  wff20:  ¬q(a)) (size:  3) (count:  2)
```

: **show-removed-beliefs**

```
¬k(a)

¬t(a)

¬v(a)

¬r(a)

¬q(a)
```

: **show-ddr-q**

```
ddr-q ()
```

: **show-active-nand-sets**

```
active-ns:

nil

active-ns-nc:

nil
```

Semi-revision of KS_1 by ¬`p(a))`

```
:  ¬p(a).


"Updating DDR with asserted node information..."


A contradiction was detected within context default-defaultct.
The contradiction involves the proposition you want to assert:
¬p(a)
and the previously existing proposition:
p(a)


You have the following options:
1.  [c] to continue anyway, knowing that a contradiction is derivable;
2.  [r] to revise the inconsistent part of the context
3.  [d] to discard this contradictory new assertion from the context
(please type c, r or d) =><= r


In order to make the context consistent you must delete at least
one hypothesis from each of the following sets of hypotheses:
(wff25 wff1)


In order to make the context consistent you must delete
at least one hypothesis from the set listed below.
```

An inconsistent set of hypotheses:

1 : ¬p(a)

(1 supported proposition: (wff25))

2 : p(a)

(4 supported propositions: (wff12 wff4 wff2 wff1))

Enter the list number of a hypothesis to examine or

[d] to discard some hypothesis from this list,

[a] to see ALL the hypotheses in the full context,

[r] to see what you have already removed,

[q] to quit revising this set, or

[i] for instructions

(please type a number OR d, a, r, q or i) =><= **2**

p(a)

WFFS that depend on wff1:

v(a)

r(a)

q(a)

p(a)

```
What do you want to do with hypothesis wff1?
[d]iscard from the context, [k]eep in the context,
[u]ndecided, [q]uit revising this set, [i]nstructions
(please type d, k, u, q or i) =><= d


The consistent set of hypotheses:
1 :  ¬p(a)
(1 supported proposition:  (wff25) )


Enter the list number of a hypothesis to examine or
 [d] to discard some hypothesis from this list,
 [a] to see ALL the hypotheses in the full context,
 [r] to see what you have already removed,
 [q] to quit revising this set, or
 [i] for instructions

(please type a number OR d, a, r, q or i) =><= q
The hyps before removal are:  (wff25:  ¬p(a) wff1:  p(a))
The hyps after making them consistent are:  (wff25:  ¬p(a))


You have selected wff1:  p(a) as a culprit and an ordering needs
to be provided.  The current culprit nodes are:
wff20:  ¬q(a)
wff21:  ¬r(a)
```

wff22: ¬v(a)

wff6: m(a)

wff17: z(a)

wff23: ¬t(a)

wff24: ¬k(a)

From the list below, choose a numbered slot to insert wff1: p(a) into.

The beliefs are ordered from most preferred to least preferred:

[1] wff20: ¬q(a) [2] wff21: ¬r(a) [3] wff22: ¬v(a) [4]

wff6: m(a) [5] wff17: z(a) [6] wff23: ¬t(a) [7] wff24: ¬k(a) [8]

Please, enter a number between 1 and 8, the culprit will be inserted

in the slot with the number specified: **1**

You have select slot 1

The following (not known to be inconsistent) set of

hypotheses was also part of the context where the

contradiction was derived:

(wff19 wff18 wff17 wff16 wff15 wff13 wff11

wff10 wff8 wff7 wff6 wff5 wff3)

Do you want to inspect or discard some of them? =><= **n**

Do you want to add a new hypothesis? =><= **n**

```
¬p(a)
```

```
:  show-removed-beliefs
¬k(a)
¬t(a)
¬v(a)
¬r(a)
¬q(a)
p(a)
```

```
:  show-culprits
culprits (from strongest to weakest):  (
[wff1:  p(a):  0.99609375]
[wff20:  ¬q(a):  0.9921875]
[wff21:  ¬r(a):  0.984375]
[wff22:  ¬v(a):  0.96875]
[wff6:  m(a):  0.9375]
[wff17:  z(a):  0.875]
[wff23:  ¬t(a):  0.75]
[wff24:  ¬k(a):  0.5])
```

SNePS System Performing DDR on KS_2

```
: ddr
Starting ddr...
Current ddr-q (
[wff1: p(a): 0.99609375] (tag: justout))
```
Q_2

```
Processing belief wff1: p(a)
...with tag: justout
Insert these beliefs into ddr-q for possible return to the context:
(wff20: ¬q(a) wff21: ¬r(a) wff22: ¬v(a))

End of ddr loop...
Current ddr-q (
[wff20: ¬q(a): 0.9921875] (tag: in?)
[wff21: ¬r(a): 0.984375] (tag: in?)
[wff22: ¬v(a): 0.96875] (tag: in?))

Processing belief wff20: ¬q(a)
...with tag: in?
wff20: ¬q(a) can return to the current context.
Returned wff20: ¬q(a) to the current context.

End of ddr loop...
```

```
Current ddr-q (
[wff21:  ¬r(a):  0.984375] (tag:  in?)
[wff22:  ¬v(a):  0.96875] (tag:  in?))

Processing belief wff21:  ¬r(a)
...with tag:  in?
wff21:  ¬r(a) can return to the current context.

Removing wff6:  m(a) from the current context.

Returned wff21:  ¬r(a) to the current context...
...  and removed the following weaker conflicting beliefs:
(wff6:  m(a))

End of ddr loop...
Current ddr-q (
[wff22:  ¬v(a):  0.96875] (tag:  in?)
[wff6:  m(a):  0.9375] (tag:  justout))

Processing belief wff22:  ¬v(a)
...with tag:  in?
wff22:  ¬v(a) cannot return to the current context.
```

```
End of ddr loop...

Current ddr-q (

[wff6:  m(a):  0.9375] (tag:  justout))


Processing belief wff6:  m(a)

...with tag:  justout


End of ddr loop...

Current ddr-q ()


Finished ddr...  ddr-q ()
```

The queue is empty

```
   :  show-removed-beliefs

¬k(a)

¬t(a)

¬v(a)

m(a)

p(a)


   :  show-nand-sets

all-ns:

nand-set7:  (wff24:  ¬k(a) wff15:  w(a) => k(a)

wff11:  w(a)), (culprit:  wff24:  ¬k(a)) (size:  3) (count:  2)
```

```
nand-set5:  (wff23:  ¬t(a) wff10:  s(a) => t(a)

wff8:  s(a)), (culprit:  wff23:  ¬t(a)) (size:  3) (count:  2)

nand-set10:  (wff22:  ¬v(a) wff18:  z(a) => v(a)

wff17:  z(a)), (culprit:  wff17:  z(a)) (size:  3) (count:  2)

nand-set3:  (wff21:  ¬r(a) wff7:  m(a) => r(a)

wff6:  m(a)), (culprit:  wff6:  m(a)) (size:  3) (count:  2)

nand-set6:  (wff22:  ¬v(a) wff13:  w(a) => v(a)

wff11:  w(a)), (culprit:  wff22:  ¬v(a)) (size:  3) (count:  2)

nand-set8:  (wff22:  ¬v(a) wff16:  p(a) => v(a)

wff1:  p(a)), (culprit:  wff22:  ¬v(a)) (size:  3) (count:  1)

nand-set2:  (wff21:  ¬r(a) wff5:  p(a) => r(a)

wff1:  p(a)), (culprit:  wff21:  ¬r(a)) (size:  3) (count:  2)

nand-set1:  (wff20:  ¬q(a) wff3:  p(a) => q(a)

wff1:  p(a)), (culprit:  wff20:  ¬q(a)) (size:  3) (count:  2)

nand-set13:  (wff25:  ¬p(a)

wff1:  p(a)), (culprit:  wff1:  p(a)) (size:  2) (count:  1)
```

`: `**`list-asserted-wffs`** *Currently believed base and derived beliefs: B∪D of a DOBS*

```
¬p(a)

¬r(a)

¬q(a)

n(a)

z(a) => v(a)
```

```
z(a)

p(a) => v(a)

w(a) => k(a)

k(a)

w(a) => v(a)

v(a)

w(a)

s(a) => t(a)

t(a)

s(a)

m(a) => r(a)

p(a) => r(a)

p(a) => q(a)
```

:

Bibliography

[Alchourrón & Makinson1985] Alchourrón, C. E., and Makinson, D. 1985. On the logic of theory change: Safe contraction. *Studia Logica* 44:405–422.

[Alchourrón, Gärdenfors, & Makinson1985] Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50(2):510–530.

[Anderson & Belnap1975] Anderson, A. R., and Belnap, Jr., N. D. 1975. *Entailment*, volume I. Princeton: Princeton University Press.

[Anderson, Belnap, & Dunn1992] Anderson, A. R.; Belnap, Jr., N. D.; and Dunn, M. 1992. *Entailment*, volume II. Princeton: Princeton University Press.

[Beckstein, Fuhge, & Kraetzschmar1993] Beckstein, C.; Fuhge, R.; and Kraetzschmar, G. 1993. Supporting assumption-based reasoning in a distributed environment. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 3–17.

[Benferhat *et al.*2004] Benferhat, S.; Kaci, S.; Berre, D. L.; and Williams, M.-A. 2004. Weakening conflicting information for iterated revision and knowledge integration. *Artif. Intell.* 153(1-2):339–371.

[Booth *et al.*2005] Booth, R.; Chopra, S.; Ghose, A.; and Meyer, T. 2005. Belief liberation (and retraction). *Studia Logica* 79 (1):47–72.

[Castro & Zurita1996] Castro, J., and Zurita, J. 1996. A generic ATMS. *International Journal of Approximate Reasoning* 14:259–280.

[Chopra, Georgatos, & Parikh2001] Chopra, S.; Georgatos, K.; and Parikh, R. 2001. Relevance sensitive non-monotonic inference on belief sequences. *Journal of Applied Non-Classical Logics* 11(1-2):131–150.

[Chopra, Ghose, & Meyer2002] Chopra, S.; Ghose, A.; and Meyer, T. 2002. Iterated revision and recovery: a unified treatment via epistemic states. In van Harmelen, F., ed., *ECAI 2002: 15th European Conference on Artificial Intelligence*, number 77 in Frontiers in Artificial Intelligence and Applications, 541–545. Amsterdam, The Netherlands: IOS Press.

[Cycorp2001a] Cycorp. 2001a. *Cycorp, Creators of the Cyc Knowledge Base*. http://cyc.com/.

[Cycorp2001b] Cycorp. 2001b. *Features of CycL*. http://cyc.com/cycl.html.

[Darwiche & Pearl1997] Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial Intelligence* (89):1–29.

[de Kleer1986] de Kleer, J. 1986. An assumption-based truth maintenance system. *Artificial Intelligence* 28(2):127–162.

[Dean & Boddy1988] Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proc. of AAAI-88*, 49–54.

[Doyle1979] Doyle, J. 1979. A truth maintenance system. *Artificial Intelligence* 12(3):231–272.

[Forbus & de Kleer1993] Forbus, K. D., and de Kleer, J. 1993. *Building Problem Solvers*. Cambridge, MA: MIT Press.

[Gärdenfors & Makinson1988] Gärdenfors, P., and Makinson, D. 1988. Revisions of knowledge systems using epistemic entrenchment. In *TARK*, 83–95.

[Gärdenfors & Rott1995] Gärdenfors, P., and Rott, H. 1995. Belief revision. In Gabbay, D. M.; Hogger, C. J.; and Robinson, J. A., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Oxford: Clarendon Press. 35–132.

[Gärdenfors1988] Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press.

[Gärdenfors1992] Gärdenfors, P. 1992. *Belief Revision*. Cambridge Computer Tracts. Cambridge: Cambridge University Press.

[Hansson1991a] Hansson, S. O. 1991a. *Belief Base Dynamics*. Ph.D. Dissertation, Uppsala University.

[Hansson1991b] Hansson, S. O. 1991b. Belief contraction without recovery. *Studia Logica* 50:251–260.

[Hansson1993] Hansson, S. O. 1993. Reversing the Levi identity. *Journal of Philosophical Logic* 22:637–669.

[Hansson1994] Hansson, S. O. 1994. Kernel contraction. *J. Symb. Logic* 59(3):845–859.

[Hansson1997] Hansson, S. O. 1997. Semi-revision. *Journal of Applied Non-Classical Logic* 7:151–175.

[Hansson1999] Hansson, S. O. 1999. *A Textbook of Belief Dynamics*, volume 11 of *Applied Logic*. Dordrecht, The Netherlands: Kluwer.

[Johnson & Shapiro1999a] Johnson, F. L., and Shapiro, S. C. 1999a. Finding and resolving contradictions in a battle scenario. Technical Report 99-09, Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY.

[Johnson & Shapiro1999b] Johnson, F. L., and Shapiro, S. C. 1999b. Says who? - incorporating source credibility issues into belief revision. Technical Report 99-08, Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY.

[Johnson & Shapiro2000a] Johnson, F. L., and Shapiro, S. C. 2000a. Formalizing a deductively open belief space. Technical Report 2000-02, Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY.

[Johnson & Shapiro2000b] Johnson, F. L., and Shapiro, S. C. 2000b. Implementing integrity constraints in an existing belief revision system. In *Papers from the Belief Change Workshop at NMR2000*. in press.

[Johnson & Shapiro2001] Johnson, F. L., and Shapiro, S. C. 2001. Redefining belief change terminology for implemented systems. In Bertossi, L., and Chomicki, J., eds., *Working Notes for the IJCAI 2001 Workshop on Inconsistency in Data and Knowledge*, 11–21. IJCAII and AAAI.

[Johnson & Shapiro2005a] Johnson, F. L., and Shapiro, S. C. 2005a. Dependency-directed reconsideration: Belief base optimization for truth maintenance systems. In *Proceedings of the Twentieth National Conference on Artificial Intellicence (AAAI-05)*, 313–320. Menlo Park, CA: AAAI Press.

[Johnson & Shapiro2005b] Johnson, F. L., and Shapiro, S. C. 2005b. Improving recovery for belief bases. In Morgenstern, L., and Pagnucco, M., eds., *IJCAI-05 Workshop on Nonmonotonic Reasoning, Action, and Change (NRAC'05): Working Notes*, 65–70. Edinburgh: IJCAII.

[Lehmann1992] Lehmann, F., ed. 1992. *Semantic Networks in Artificial Intelligence*. Oxford: Pergamon Press.

[Malheiro & Oliveira1996] Malheiro, B., and Oliveira, E. 1996. Consistency and context management in a multi-agent belief revision testbed. In Wooldridge, M.; Müller, J.-P.; and Tambe, M., eds., *Intelligent Agents II — Agent Theories, Architectures, and Languages (LNAI 1037)*. Springer-Verlag: Heidelberg, Germany. 361–375.

[Martins & Shapiro1983] Martins, J. P., and Shapiro, S. C. 1983. Reasoning in multiple belief spaces. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 370–373.

[Martins & Shapiro1988] Martins, J. P., and Shapiro, S. C. 1988. A model for belief revision. *Artificial Intelligence* 35:25–79.

[Martins1983] Martins, J. P. 1983. *Reasoning in Multiple Belief Spaces*. Ph.D. dissertation, Technical Report 203, Department of Computer Science, SUNY at Buffalo.

[McAllester1978] McAllester, D. A. 1978. A three valued truth maintenance system. AI Memo 473, Massachusetts Institute of Technology AI Lab, Cambridge, MA.

[Meyer2000] Meyer, T. 2000. Merging epistemic states. In Mizoguchi, R., and Slaney, J., eds., *Proceedings of PRICAI 2000*, volume 1886 of *Lecture Notes in Artificial Intelligence*, 286–296. Springer-Verlag.

[Nayak & Williams1998] Nayak, P. P., and Williams, B. C. 1998. Fast context switching in real-time propositional reasoning. In Senator, T., and Buchanan, B., eds., *Proceedings of the Fourteenth National Conference on Artificial Intelligence and the Ninth Innovative Applications of Artificial Intelligence Conference*, 50–56. Menlo Park, California: AAAI Press.

[Nayak1994] Nayak, A. C. 1994. Foundational belief change. *Journal of Philosophical Logic* 23(5):495–533.

[Nebel1989] Nebel, B. 1989. A knowledge level analysis of belief revision. In Brachman, R. J.; Levesque, H. J.; and Reiter, R., eds., *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning (KR'89)*, 301–311.

[Nebel1994] Nebel, B. 1994. Base revision operations and schemes: Semantics, representation, and complexity. In *Proc. of the 11th ECAI*, 341–345.

[Pell *et al.*1997] Pell, B.; Bernard, D. E.; Chien, S. A.; Gat, E.; Muscettola, N.; Nayak, P. P.; Wagner, M. D.; and Williams, B. C. 1997. An autonomous spacecraft agent prototype. In Johnson, W. L., and Hayes-Roth, B., eds., *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, 253–261. New York: ACM Press.

[Rapaport & Shapiro1995] Rapaport, W. J., and Shapiro, S. C. 1995. Cognition and fiction. In Duchan, J. F.; Bruder, G. A.; and Hewitt, L. E., eds., *Deixis in Narrative: A Cognitive Science Perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc. 107–128.

[Russell & Norvig2003] Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition. http://aima.cs.berkeley.edu/.

[Shapiro & Rapaport1992] Shapiro, S. C., and Rapaport, W. J. 1992. The SNePS family. *Computers & Mathematics with Applications* 23(2–5):243–275. Reprinted in [Lehmann1992][pp. 243–275].

[Shapiro & The SNePS Implementation Group2004] Shapiro, S. C., and The SNePS Implementation Group. 2004. *SNePS 2.6.1 User's Manual*. Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY. Available as http://www.cse.buffalo.edu/sneps/Manuals/manual261.ps.

[Shapiro1992] Shapiro, S. C. 1992. Relevance logic in computer science. In Anderson, A. R.; Belnap, Jr., N. D.; and Dunn, M., eds., *Entailment*, volume II. Princeton: Princeton University Press. 553–563.

[Shapiro2000] Shapiro, S. C. 2000. SNePS: A logic for natural language understanding and commonsense reasoning. In Iwańska, Ł., and Shapiro, S. C., eds., *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. Menlo Park, CA: AAAI Press/The MIT Press. 175–195.

[Stallman & Sussman1977] Stallman, R., and Sussman, G. J. 1977. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence* 9(2):135–196.

[Wassermann1999] Wassermann, R. 1999. *Resource-bounded Belief Revision*. Ph.D. Dissertation, University of Amsterdam, Amsterdam, The Netherlands.

[Williams & Sims2000] Williams, M.-A., and Sims, A. 2000. SATEN: An object-oriented web-based revision and extraction engine. In Baral, C., and Truszyński, M., eds., *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning NMR'2000*. CoRR article: cs.AI/0003059.

[Williams1994a] Williams, M.-A. 1994a. On the logic of theory base change. In MacNish, C.; Pearce, D.; and Pereira, L. M., eds., *Logics in Artificial Intelligence(JELIA)*. Berlin, Heidelberg: Springer. 86–105.

[Williams1994b] Williams, M.-A. 1994b. Transmutations of knowledge systems. In Doyle, J.; Sandwall, E.; and Torasso, P., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference*, 619–629. San Mateo, CA: Morgan Kaufmann.

[Williams1995] Williams, M.-A. 1995. Iterated theory base change: A computational model. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1541–1550. Morgan Kaufmann Publishers.

[Williams1997] Williams, M.-A. 1997. Anytime belief revision. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. 74–79.

[Zilberstein1996] Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI Magazine* 17(3):73–83.