Implementing Iterated Belief Change Via Prime Implicates

Zhi Qiang Zhuang[1], Maurice Pagnucco[2], and Thomas Meyer[3]

[1] School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia
zqzh390@cse.unsw.edu.au
[2] ARC Centre of Excellence for Autonomous Systems and National ICT Australia, School of Comp. Sci. and Eng., The University of New South Wales, Sydney, NSW 2052, Australia
morri@cse.unsw.edu.au
http://www.cse.unsw.edu.au/~morri/
[3] National ICT Australia and School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia
Thomas.Meyer@nicta.com.au
http://www.cse.unsw.edu.au/~tmeyer/

Belief change is concerned with modelling the way in which an idealised (rational) reasoner maintains their beliefs and the way in which those beliefs are modified as the reasoner acquires new information. The AGM [1,3,5] framework is the most widely cited belief change methodology in the literature. It models the reasoner's belief state as a set of sentences that is logically closed under deduction and provides for three belief change operations: expansion, contraction and revision. Each of the AGM belief change operations is motivated by *principles of rationality* that are formalised by way of *rationality postulates*.

Pagnucco [10] formalised a way of implementing the AGM belief change operations using a *knowledge compilation* technique involving *prime implicates* in order to improve computational efficiency. This technique exploits the *epistemic entrenchment* construction for AGM belief change by Gärdenfors and Makinson [4] by introducing the notion of a *compiled epistemic entrenchment*. It has a number of significant features: (a) the belief change operators constructed satisfy the AGM postulates; (b) when compilation has been effected only subsumption checking and some simple syntactic manipulation is required in order to contract (or revise) the reasoner's belief state.

The aim of this paper is twofold. Firstly, to supply algorithms for Pagnucco's [10] formally specified prime-implicate technique for AGM belief change thus facilitating implementation. Secondly, to provide an empirical analysis of a Java-based implementation of these algorithms (and thus the formal technique).

Implementations of AGM belief change include the work of Dixon and Wobcke [2] and Williams [16] who adopt theorem proving techniques to realise each of the operations. The work here can be seen as an attempt to improve the computational efficiency of implementations like these through the use of knowledge compilation. Prime implicates have been used in *truth maintenance systems* by Reiter and de Kleer [11] and by Kean [8]. However these approaches lack any preferential structure that allows them to be used for more general forms of belief change. Gorogiannis and Ryan [6] use an alternate form of knowledge compilation—binary decision diagrams (BDDs)—but focus on the complexity of specific operations.

1 Background

We assume a fixed finite propositional object language $\mathcal{L}$ with the standard logical connectives. Cn represents the classical consequence operator while $\vdash$ represents the classical consequence relation (i.e., $\Gamma \vdash \phi$ iff $\phi \in Cn(\Gamma)$). We also adopt the following linguistic conventions to simplify the presentation. Lower case Greek letters ϕ, ψ, χ, …denote sentences of $\mathcal{L}$. Upper case Greek letters Δ, Γ, …denote sets of formulas. Theories (closed under logical consequence) are used to represent reasoners' belief states in AGM belief change and will be denoted by upper case Roman characters H, K, …(i.e., $K = Cn(K)$) that are termed *belief sets*. The inconsistent belief set is denoted $K_\perp$. Lower case Roman characters l, k, …denote literals (both positive and negative). Upper case Roman characters C, D, …denote clauses.

Prime implicates are minimal length clauses (in terms of set inclusion) implied by a knowledge base. Transforming a knowledge base into a set of prime implicates gives a uniform, and logically equivalent, way of expressing the sentences in the knowledge base that can be exploited to enhance computational efficiency. We denote the set of prime implicates of an arbitrary set of sentences Γ by $\Pi(\Gamma)$. When Γ is a set of clauses, prime implicates are easily computed by repeated application of resolution and removing subsumed (i.e., implied) clauses [15]. This strategy can be improved by using incremental methods [9,7], and the use of efficient data structures [11]. Compiling a knowledge base into prime implicate form can lead to an exponential number of implicates in the number of atoms (see [13]).

1.1 AGM Belief Change

Alchourrón, Gärdenfors and Makinson (AGM) [1,3,5] introduced one of the most widely adopted belief change frameworks. The AGM considers three types of transformations on beliefs sets as new information is acquired: *belief expansion* by ϕ in which this new information is added to the initial belief set K without removal of any existing beliefs ($K + \phi$); *belief contraction* by ϕ where belief in ϕ is suspended ($K \dot{-} \phi$); and, *belief revision* by ϕ where ϕ is assimilated into K while existing beliefs may need to be suspended in order to maintain consistency ($K * \phi$). These operations are motivated by *rationality criteria* and characterised by a set of *rationality postulates* which—for the sake of brevity—we do not reproduce here but rather refer the interested reader to the AGM account [1,3,5].

One element of the AGM that does interest us here is the notion of *epistemic entrenchment* that can be used to construct a contraction or revision function satisfying the AGM rationality postulates. An epistemic entrenchment is an ordering over the sentences in K that can be used to determine which sentences to remove during belief contraction and revision. Formally, an epistemic entrenchment relation $\leq$ satisfies the following properties [4].

(EE1) If $\phi \leq \psi$ and $\psi \leq \gamma$ then $\phi \leq \gamma$
(EE2) If $\{\phi\} \vdash \psi$ then $\phi \leq \psi$

(EE3) For any ϕ and ψ, $\phi \leq \phi \wedge \psi$ or $\psi \leq \phi \wedge \psi$
(EE4) When $K \neq K_\perp$, $\phi \notin K$ iff $\phi \leq \psi$ for all ψ
(EE5) If $\phi \leq \psi$ for all ϕ then $\vdash \psi$

(EE1)–(EE3) entail that $\leq$ is a total pre-order. In other words, a ranking of sentences. (EE4) says that non-beliefs are minimally entrenched while (EE5) says that tautologies are maximally entrenched. The following condition [4] tells us how to define a contraction function using epistemic entrenchment.

$$(C\dot{-}) \quad \psi \in K\dot{-}\phi \text{ iff } \psi \in K \text{ and either } \phi < \phi \vee \psi \text{ or } \vdash \phi$$

In the principal case a belief ψ is retained when contracting K by ϕ provided that there is additional evidence to support it ($\phi < \phi \vee \psi$).

1.2 Compiling Epistemic Entrenchment

Pagnucco [10] introduces a theory to reduce the amount of information required to construct an AGM contraction function via epistemic entrenchment by compiling the relation using prime implicates. An observation by Rott [12] is central to this idea.

Proposition 1. *[12] For arbitrary sentence $\psi \in \mathcal{L}$, $\{\phi : \psi \leq \phi\} = Cn(\{\phi : \psi \leq \phi\})$.*

Put simply, if we were to "cut" the epistemic entrenchment relation at any level, the beliefs that are at least this entrenched would form a set that is deductively closed (i.e., a belief set)[1].

A compiled epistemic entrenchment is essentially one in which we take each cut and replace it by its prime implicates thus significantly reducing the number of sentences required to be explicitly represented in the entrenchment.

Definition 1. *($\leq_\Pi$)*

Given an epistemic entrenchment ordering $\leq$ satisfying properties (EE1)–(EE5) we define a compiled epistemic entrenchment ordering $\leq_\Pi$ as follows. For any two clauses C, D, $C \leq_\Pi D$ iff all of the following hold:

1. *$C \leq D$;*
2. *$C \in \Pi(\{\phi : \phi \leq \psi\})$ for some $\psi \in \mathcal{L}$; and,*
3. *$D \in \Pi(\{\phi : \phi \leq \chi\})$ for some $\chi \in \mathcal{L}$*

Note that the empty clause is less entrenched than all clauses which we denote $\perp \leq C$.

We omit the theorems that show the correctness of this and most of the following definitions as they can be found in Pagnucco [10].

One concept we require is the (maximal) level of entrenchment of a clause.

Definition 2. *Let C be a clause, $max_{\leq_\Pi}(C) = \{D :$ such that*

1. *D is a clause where $D \subseteq C$, and*
2. *there is no other clause $D' \subset C$ and $D \leq_\Pi D'$.$\}$*

[1] The term "cut" is due to Rott [12] and we shall adopt it here.

We can now define AGM contraction using the compiled entrenchment by appropriately modifying condition (C$\dot{-}$). The following definition deals with contraction of K by a single clause and we shall show how to extend it shortly. It is a simplified version of the definition in [10]. Note that while $max_{\leq_\Pi}(C)$ is a set of clauses, all these clauses are at the same level of entrenchment and so we abuse this notation in the following by allowing this set to be represented by any one of its elements.

Definition 3. *Given an epistemic entrenchment relation $\leq$ and a clause C we define the contracted compiled epistemic entrenchment $\leq_\Pi^C$ of a compiled epistemic entrenchment $\leq_\Pi$ by C as: $D \leq_\Pi^C E$ iff either*

1. *$D \leq E$ and $max_{\leq_\Pi}(C) < max_{\leq_\Pi}(C \cup D), max_{\leq_\Pi}(C \cup E)$, or*
2. *$A \leq B$ and $A = C \cup F$ with $max_{\leq_\Pi}(C) < C \cup D \cup F$ and $B = E \cup G$ with $max_{\leq_\Pi}(C) < C \cup E \cup G$ (note F and G may be empty).*

Condition (1) uses (C$\dot{-}$) directly to determine when clauses in the original compiled epistemic entrenchment should be retained. The second condition determines whether clauses that do not satisfy condition (1) and are removed should be replaced by weaker clauses. If we do not allow replacement by weaker clauses, the AGM postulates will not be satisfied. We see that this definition is correct as far as single clauses are concerned.

Theorem 1. *[10] Let K be a belief set, $\dot{-}$ an AGM contraction function satisfying (K$\dot{-}$1)–(K$\dot{-}$8) and $\leq$ an epistemic entrenchment relation defined from $\dot{-}$ by condition (C$\leq$). Furthermore, let C be a clause. Then $K \dot{-} C = Cn(\{D : \bot <_\Pi^C D\})$.*

Pagnucco uses some properties of epistemic entrenchment to show that contraction by an arbitrary formula can be easily achieved by converting the formula into CNF and contracting by the minimally entrenched conjuncts.

Definition 4. *Let $\phi \in \mathcal{L}$, the set of minimal conjuncts of ϕ is defined as follows: $min_{\leq_\Pi}(CNF(\phi)) = \{C_i \in CNF(\phi) : max_{\leq_\Pi}(C_i) \leq_\Pi max_{\leq_\Pi}(C_j)$ for all $C_j \in CNF(\phi)\}$.*

We now extend our previous result to show that this produces the desired contraction.

Theorem 2. *[10] Let K be a belief set, $\dot{-}$ an AGM contraction function satisfying (K$\dot{-}$1)–(K$\dot{-}$8) and $\leq$ an epistemic entrenchment relation defined from $\dot{-}$ by condition (C$\leq$). Furthermore, let ϕ be a sentence.*
Then $K \dot{-} \phi = \bigcap_{C_i \in min_{\leq_\Pi}(CNF(\phi))} Cn(\{D : \bot <_\Pi^{C_i} D\})$.

Belief revision can be implemented by combining contraction and expansion via the Levi Identity: $K * \phi = (K \dot{-} \neg\phi) + \phi$ where AGM expansion is simply deductive closure (i.e., $K + \phi = Cn(K \cup \{\phi\})$).

2 Algorithms

The basis for the knowledge compilation approach we adopt is the compiled epistemic entrenchment relation. As noted above it is a total pre-order over clauses (i.e., a ranking). We model this pre-order using ranked levels identified by integers; the greater the

Algorithm 1. Algorithm for finding the level of entrenchment of a clause C in belief set K represented as a compiled epistemic entrenchment relation. On termination the algorithm returns the level at which the clause C is entrenched, -1 if the belief set K does not contain C or ∞ if C is a tautology.

```
FIND-LEVEL($K, C$)
 1  if $C$ is a tautology then
 2      return $\infty$
 3  $L \leftarrow L_K$
 4  while $L$ is not empty do
 5      $m \leftarrow$ highest level in $L$
 6      $L \leftarrow L \setminus \{m\}$
 7      for each clause $D \in K(m)$ do
 8          if $D \subseteq C$ then
 9              return $m$
10  return $-1$
```

integer, the higher the level of epistemic entrenchment. Each formula (implicate) has an associated integer level of entrenchment.

This means that the belief change operations of expansion and revision which involve the addition of information, require that the level of entrenchment for the formula representing the new information be supplied as well. This is not how the respective AGM operations work however we note that this is required only to allow for iterated belief change. If we were only interested in a single expansion or revision, then this level of entrenchment would not be required.

We adopt the following notation to simplify the algorithms in this section. For a belief set K we denote by $K(n)$ the set of clauses *explicitly* stored at level n in the compiled epistemic entrenchment. Similarly, $K(n+)$ denotes the set of clauses explicitly retained at level n or greater while $K(n-)$ denotes those clauses explicitly retained at level n or less. L_K is a set of integers denoting the levels of the compiled epistemic entrenchment at which clauses are explicitly maintained. $L_K(n+)$ (respectively, $L_K(n-)$), denotes the set of integer levels in the compiled entrenchment containing clauses greater than or equal to (respectively, less than or equal to) n. A function *computePI()* takes a set of clauses and returns the corresponding prime implicates.

2.1 Find Level

The first algorithm that we require is one to determine the rank (i.e., level) of a clause in the compiled epistemic entrenchment relation. This is required for belief expansion and also to implement belief contraction using the (C$\dot{-}$) condition.

The heart of Algorithm 1 is lines 4–9. Starting with the highest level of the compiled entrenchment[2] the algorithm looks through each successive level of entrenchment

[2] Keep in mind that the compiled entrenchment—unlike AGM entrenchment—does not explicitly maintain tautologies.

Algorithm 2. Algorithm for expanding belief set K by clause C at level n

```
EXPANSION($K, C, n$)
 1  $h \leftarrow$ FIND-LEVEL($K, C$)
 2  if $h \geq n$ then
 3      return
 4  if $n \notin L_K$ then
 5      $L_K \leftarrow L_K \cup \{n\}$
 6  $L \leftarrow L_K(n-)$
 7  while $L$ is not empty do
 8      $m \leftarrow$ highest level in $L$
 9      $L \leftarrow L \setminus \{m\}$
10      if $m = n$ then
11          $P \leftarrow computePI(K(m+) \cup \{C\})$
12      else
13          $P \leftarrow computePI(K(m+))$
14      if $P$ is empty then
15          $K \leftarrow K_\perp$
16          return
17      $N \leftarrow P \setminus K(m+)$
18      for each clause $D \in K(m-)$ do
19          for each clause $D' \in N$ do
20              if $D' \subseteq D$ then
21                  remove $D$ from $K$
22      add all clauses in $N$ to level $m$ of $K$
```

to determine whether the clause in question is subsumed by one of the implicates at that level. As soon as such an implicate is found, the level at which this implicate is entrenched corresponds to the level at which the clause is entrenched by Proposition 1.

2.2 Belief Expansion

Belief expansion using a compiled entrenchment differs slightly from AGM expansion because we are interested in iterated belief change. This is in keeping with implementations like those of Dixon and Wobcke [2] and Williams [16] and the belief change framework of Spohn [14]. Accordingly, the belief expansion operation takes three inputs: a belief set K, a clause C to be added and a level n at which it is to be added to the compiled entrenchment. Note that if a formula is to be added rather than a clause, the formula can be converted to CNF and each clause added at that level.

Lines 2–3 of Algorithm 2 determine whether the clause is already implied by the compiled entrenchment at a level greater than or equal to n and, if so, nothing is required to be done. Lines 4–5 concern the case in which the clause needs to be added at a level that does not explicitly exist in the compiled entrenchment in which case n is simply added to the levels in L_K. The remaining lines 7–22 take care of adding C to the

Algorithm 3. Algorithm for contracting a belief set K by a single clause C. On termination it returns the resulting compiled entrenchment for belief set $K \dot{-} C$.

```
SINGLE-CLAUSE-CONTRACTION($K, C$)
 1  $n \leftarrow$ FIND-LEVEL($K, C$)
 2  if $n = -1$ or $n = \infty$ then
 3      return $K$
 4  remove $C$ from $K$
 5  $L \leftarrow L_K(n-)$
 6  while $L$ is not empty do
 7      $m \leftarrow$ highest level in $L$
 8      $L \leftarrow L \setminus \{m\}$
 9      $N \leftarrow \{\}$
10      for each clause $D \in K(m)$ do
11          $h \leftarrow$ FIND-LEVEL($K, C \cup D$)
12          if $n = h$ then
13              remove $D$ from $K$
14              for each literal $l \in C$ do
15                  $k \leftarrow$ FIND-LEVEL($K, \{\neg l\} \cup D$)
16                  if $k = m$ then
17                      $N \leftarrow N \cup \{\{\neg l\} \cup D\}$
18              for each clause $E$ in $K((n+1)+)$ do
19                  if $C \cup D \subseteq E$ then
20                      $E \leftarrow (E \setminus C) \cup D$
21                      $N \leftarrow N \cup \{E\}$
22      $P \leftarrow computePI(N \cup K(m+))$
23      $P \leftarrow P \setminus K(m+)$
24      add all clauses in $P$ to level $m$ of $K$
25  return $K$
```

compiled entrenchment and re-computing the prime implicates at each level less than or equal to n. Line 11 is executed the first time through the loop, adding C to the compiled entrenchment at level n while line 12 is executed on subsequent iterations. Lines 11 and 12 re-compute the prime implicates at each level. Lines 17–21 take care of removing any subsumed clauses from the compiled entrenchment to ensure that only those clauses that need to be represented are maintained. Lines 14–16 handle the case of expansion into inconsistency. Note that the function *computePI()* can be implemented using an incremental technique [7,9] to reduce the amount of computation required.

2.3 Belief Contraction

As with the theoretical development of belief contraction using a compiled epistemic entrenchment relation we begin by considering the algorithm for belief contraction by a single clause C. Algorithm 3 uses Definition 3 which is motivated by Gärdenfors and

Algorithm 4. Algorithm for intersecting two compiled entrenchments representing belief sets K_1 and K_2. On termination it returns the intersection of the two compiled entrenchments.

```
INTERSECTION($K_1, K_2$)
 1  $K \leftarrow$ empty belief set
 2  $L_1 \leftarrow L_{K_1}$
 3  $L_2 \leftarrow L_{K_2}$
 4  for each level $n \in L_2$
 5      if $n \notin L_1$ then
 6          $L_1 \leftarrow L_1 \cup \{n\}$
 7  while $L_1$ is not empty do
 8      $m \leftarrow$ highest level in $L_1$
 9      $L_1 \leftarrow L_1 \setminus \{m\}$
10      $N \leftarrow \{\}$
11      for each clause $C \in K_1(m)$ do
12          for each clause $D \in K_2(m+)$ do
13              if $C \subseteq D$ and FIND-LEVEL$(K, D) = -1$ then
14                  $N \leftarrow N \cup \{D\}$
15      for each clause $C \in K_2(m)$ do
16          for each clause $D \in K_1(m+)$ do
17              if $C \subseteq D$ and FIND-LEVEL$(K, D) = -1$ then
18                  $N \leftarrow N \cup \{D\}$
19      $P \leftarrow computePI(N)$
20      add all clauses in $P$ to level $m$ of $K$
21  return $K$
```

Makinson's condition (C$\dot{-}$). Lines 1–3 determine whether it is necessary to perform contraction. Definition 3 and condition (C$\dot{-}$) tell us that clauses at a higher level of entrenchment to C are unaffected by contraction. The while loop encompassing lines 6–24 considers the clauses explicitly entrenched at levels less than or equal to that of C in the compiled entrenchment, determining which are to be removed and whether they are replaced by weaker clauses in order to adhere to the AGM postulates. Definition 3(2) (and Condition (C$\dot{-}$)) tell us that a clause D in the compiled entrenchment that is at a level less than or equal to that of clause C should be removed whenever $C = C \vee D$ which is the condition tested on lines 11 and 12 and the clause is subsequently removed at line 13. Having determined to remove the clause we now need to determine whether the clause D should be replaced by logically weaker clauses. This is done in lines 18–21 of the algorithm. Lines 14–17 add clauses that are required by the Recovery postulate (K$\dot{-}$5). The remaining lines 22–24 re-compute the prime implicates as required.

To contract a compiled entrenchment by an arbitrary sentence requires a slightly special form of intersection that takes into account subsumed clauses. This is in keeping with our desire to implement iterated belief change. Lines 4–7 ensure that all levels of both compiled entrenchments are considered (which is required for subsumed clauses). A clause C at level m of K_1 is checked against all clauses at level j of K_2 for all $j \geq m$.

If there are clauses that C subsumes but C is not in the resulting belief set K then the subsumed clause is added to level m of K since it is minimal. As the same procedure has to be performed for both belief sets, we have two for loops in lines 11–14 and 15–18 of the algorithm that iterate over each of the compiled entrenchments for both belief sets achieving this task. Finally, in lines 19–20 prime implicates are re-computed.

We are now in a position to implement the algorithm to contract by arbitrary sentences converted into CNF. By Theorem 1 we only need to consider the minimally entrenched clauses in the CNF of the formula, contract K by each in turn and take the intersection of the result. Algorithm LEAST-ENTRENCHED-CLAUSE(K, S) takes care of determining the least entrenched clauses while ARBITRARY-SENTENCE-CONTRACTION(K, S) performs the individual contractions and takes the required intersections using the preceding algorithm.

Algorithm 5. Algorithm for contracting belief set K by an arbitrary sentence S in CNF

```
LEAST-ENTRENCHED-CLAUSE$(K, S)$
  $N \leftarrow \{\}$
  **for** each clause $C \in S$ **do**
      **if** $N \leftarrow \{\}$ **then**
          $N \leftarrow \{C\}$
      **else**
          $D \leftarrow$ first element in $N$
          **if** FIND-LEVEL$(K, C) >$ FIND-LEVEL$(K, D)$ **then**
              $N \leftarrow \{C\}$
          **elif** FIND-LEVEL$(K, C) =$ FIND-LEVEL$(K, D)$ **then**
              $N \leftarrow N \cup \{C\}$
  **return** $N$

ARBITRARY-SENTENCE-CONTRACTION$(K, S)$
  $I \leftarrow$ empty belief set
  $N \leftarrow$ LEAST-ENTRENCHED-CLAUSE$(K, S)$
  **for** each clause $D \in N$ **do**
      $K^{'} \leftarrow K$
      $I^{'} \leftarrow$ SINGLE-CLAUSE-CONTRACTION$(K^{'}, D)$
      **if** $I$ is empty belief set **then**
          $I \leftarrow I^{'}$
      **else**
          $I \leftarrow$ INTERSECTION$(I, I^{'})$
  $K \leftarrow I$
```

Note that belief revision is now easily achieved by combining belief contraction and belief expansion via the Levi Identity ($K * \phi = (K \dot{-} \phi) + \phi$) so we omit a specific algorithm for this straightforward operation.

3 Analysis

The foregoing algorithms were implemented in Java and subjected to empirical analysis as we describe here. The experiments performed were conducted on randomly generated 3-clauses using belief revision rather than just expansion or contraction. The reason for this latter decision is that using expansion alone may lead to an inconsistent belief set and contraction may involve a lot of degenerate cases (attempting to contract a clause that is not implied by the compiled entrenchment) which would skew the results. Adopting revision is guaranteed to require one of expansion or contraction to be performed and possibly both.

In the first experiment the compiled entrenchment is continually revised by a randomly generated 3-clause which is added at a random level of entrenchment between 1 and 20. Each 3-clause is generated from a vocabulary of 5 propositional letters to ensure that there is a reasonable chance for interaction among the clauses added to the compiled entrenchment (otherwise the experiments will tend to be uninteresting). For each number of clauses 50 trials were conducted and the average time taken recorded. The resulting graph is show in Figure 1(a). We see that initially, with few clauses, the average running time increases dramatically. At about 50 clauses the graph begins to level off. This indicates that there is indeed an advantage to compiling the entrenchment relation using prime implicates. After some initial work, changes tend to be minor. Note however that our choice to restrict the language to 5 propositional letters may also be a factor as after some time, clauses may be more likely to either already occur in the compiled entrenchment. This requires no work in terms of revision. Note however that there is also an increased likelihood of conflict occurring.

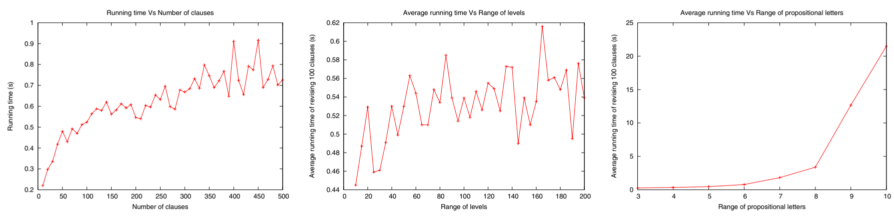

Fig. 1. (a) Running time vs number of clauses; (b) running time vs range of levels; (c) running time vs range of propositional letters

The second experiment involves adjusting the coarseness of the compiled entrenchment by varying the maximum number of levels of entrenchment allowed. In this case the compiled entrenchment is revised by 100 randomly generated 3-clauses (again, chosen from a vocabulary of 5 propositional letters) but with the maximum number of levels increasing by 10 each time. 50 trials were conducted for each data point. While there is a reasonable fluctuation in the graph—shown in Figure 1(b)—a closer look reveals a general upward trend. This is to be expected. In conducting either expansion or contraction, the crucial phase is to re-compute prime implicates. For expansion this occurs at all levels less than or equal to the one at which the clause is added. For contraction it occurs at all levels less than or equal to the one at which the clause resides. With more levels of entrenchment, the possibility of having to do this over a greater number of levels increases. Even with the use of an incremental algorithm for re-computing prime implicates this incurs a significant cost.

The final experiment varies the range of propositional letters used to generate 3-clauses from 3 to 10. Again 50 trials are conducted for each data point with 100 3-clauses generated for revision. As expected we observe a significant increase in the average running time as the number of propositional letters increases. However, we have not at this stage attempted many optimisations in (re-)computing prime implicates which may help to ameliorate this situation.

4 Conclusions

In this paper we have introduced algorithms that implement Pagnucco's [10] formal framework for implementing AGM belief revision through the use of prime implicates to compile the epistemic entrenchment relation. We have also subjected these algorithms to an initial empirical analysis. This analysis indicates that the compilation of the epistemic entrenchment relation can lead to an improvement in the average running time for belief revision. Further empirical analysis is expected to confirm these findings.

Apart from a larger range of experiments, further work involves trying to optimise parts of the algorithms we have presented, in particular the generation of prime implicates and the cost of subsumption checking. These can be achieved through a better choice of data structures and the employment of heuristic techniques commonly used in modern SAT solvers. Another avenue for future research is the extension of our algorithms to first-order logic where the definition of prime implicates becomes more problematic but where a restricted definition may afford some purchase.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. of Symbolic Logic 50, 510–530 (1985)
2. Dixon, S.E., Wobcke, W.: The implementation of a first-order logic AGM belief revision system. In: Proc. of the Fifth IEEE Int. Conf. on Tools in Art. Int. (1993)
3. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. Bradford Books, MIT Press, Cambridge Massachusetts (1988)
4. Gärdenfors, P., Makinson, D.: Revisions of knowledge systems using epistemic entrenchment. In: Proc. of 2nd Conf. on Th. Aspect of Reas. About Knowl., pp. 83–96 (1988)
5. Gärdenfors, P., Rott, H.: Belief revision. In: Handbook of Logic in AI and Logic Programming vol. IV: Epistemic and Temporal Reasoning, OUP, pp. 35–132 (1995)
6. Gorogiannis, N., Ryan, M.D.: Implementation of belief change operators using BDDs. Studia Logica 70(1), 131–156 (2002)
7. Jackson, P.: Computing prime implicates incrementally. In: Proceedings of the Eleventh Conference on Automated Deduction (June 1992)
8. Kean, A.: A formal characterisation of a domain independent abductive reasoning system. Technical Report HKUST-CS93-4, Dept. of Computer Science, HKUST (1993)
9. Kean, A., Tsiknis, G.: An incremental method for generating prime implicants/implicates. Journal of Symbolic Computation 9, 185–206 (1990)

10. Pagnucco, M.: Knowledge compilation for belief change. In: Sattar, A., Kang, B.-H. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 90–99. Springer, Heidelberg (2006)
11. Reiter, R., de Kleer, J.: Foundations of assumption-based truth maintenance systems: Preliminary report. In: Proc. of the Nat. Conf. in AI, pp. 183–188 (1987)
12. Rott, H.: Preferential belief change using generalized epistemic entrenchment. Journal of Logic, Language and Information 1, 45–78 (1992)
13. Schrag, R., Crawford, J.M.: Implicates and prime implicates in random 3-SAT. Artificial Intelligence 81(1-2), 199–222 (1996)
14. Spohn, W.: Ordinal conditional functions: A dynamic theory of epistemic states. In: Causation in Decision, Belief Change, and Statistics, II, pp. 105–134. Kluwer, Dordrecht (1988)
15. Tison, P.: Generalization of consensus theory and application to the minimization of boolean functions. IEEE Trans. on Elec. Computers 4, 446–456 (1967)
16. Williams, M.-A.: Iterated theory change: A computational model. In: Proc. of the Fourteenth International Joint Conference on Artificial Intelligence, pp. 1541–1550 (1995)