Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Demonstrations Track

GenC: A Fast Tool for Applications Involving Belief Revision

Aaron Hunter and **John Agapeyev**
British Columbia Institute of Technology, Burnaby, Canada
aaron_hunter@bcit.ca, jagapeyev@gmail.com

Abstract

The process of belief revision occurs in many applications where agents may have incorrect or incomplete information. One important theoretical model of belief revision is the well-known AGM approach. Unfortunately, there are few tools available for solving AGM revision problems quickly; this has limited the use of AGM operators for practical applications. In this demonstration paper, we describe GenC, a tool that is able to quickly calculate the result of AGM belief revision for formulas with hundreds of variables and millions of clauses. GenC uses an AllSAT solver and parallel processing to solve revision problems at a rate much faster than existing systems. The solver works for the class of *parametrised difference operators*, which is an extensive class of revision operators that use a weighted Hamming distance to measure the similarity between states. We demonstrate how GenC can be used as a stand-alone tool or as a component of a reasoning system for a variety of applications.

1 Introduction

The process of *belief revision* occurs when an agent receives new information about the world, which must be incorporated with some prior set of beliefs. In the logic-based AI community, the dominant approach to belief revision has been the AGM approach [Alchourrón *et al.*, 1985]. In this approach, a *belief revision operator* is a function that maps initial beliefs to new beliefs when new information is obtained. While the theory of belief revision has been well-studied, there has not been a great deal of emphasis on the development of implemented systems. Worse yet, the systems that do exist tend to be restricted in scope, and very slow on large test cases. This is in part due to the compexity of belief revision, which lies at the second level of the polynomial hierarchy [Eiter and Gottlob, 1992]. In this paper, we describe a new solver that outperforms existing tools on large instances for an important class of revision operators. Note that this is a demonstration paper focused on describing the software, performance and applications; a more detailed paper that includes context and theory can be found in [Hunter and Agapeyev, 2019].

2 Application Domain

GenC is a solver for AGM belief revision. In the interest of space, we give only a minimal introduction to AGM revision here, and refer the reader to [Alchourrón *et al.*, 1985] for a complete introduction.

A *belief set* is a closed set of formulas of propositional logic. An AGM revision operator takes a belief set K and a formula ϕ as input, and it returns a new belief set $K * \phi$. In addition, an AGM revision operator needs to satisfy the so-called AGM postulates, which give constraints on how $*$ operates. Roughly, for an AGM operator $*$, the set $K * \phi$ includes ϕ along with as much of K as possible.

One well-known AGM revision operator is the Dalal operator $*_d$, where the revision by ϕ results in a new belief set satisfied by all models that are minimally distant from models of K in terms of the Hamming distance [Dalal, 1988]. *Parametrised difference (PD) operators* are a generalization of the Dalal operator, where an ordering over the symbols in the vocabulary gives precedence to certain symbols [Peppas and Williams, 2018]. Every ordering defines a different AGM operator; this gives a large class of operators for different purposes.

3 Problem Scenario

Our goal was to develop a solver that quickly calculates the result of AGM revision, for an extensive set of operators. We selected the class of PD operators, as they are simple to specify, yet expressive enough to be useful for practical applications. In order to calculate the result of revision, we need to give the user the ability to enter three things: a belief set K, a formula for revision ϕ, and an (optional) ordering over variables.

GenC is a command line application, that can be run in interactive mode or in file-based mode. In interactive mode, we enter formulas at the command prompt using integers as propositional variables:

```
Entering initial belief set:
> 1 and 2
Entering revision formula:
> not 1 or not 2
Revised belief set:
(not 1 and 2) or (1 and not 2)
```

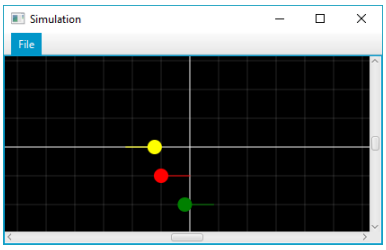

Figure 1: Moving Agents

This shows that $Cn\{(p \wedge q)\} * \neg p \vee \neg q$ is $Cn\{(\neg p \wedge q) \vee (p \wedge \neg q)\}$, using the Dalal revision operator by default.

In file-based mode, K and ϕ are given in an external file in Conjunctive Normal Form (CNF) or Disjunctive Normal Form (DNF) using the standard DIMACS format used in SAT competitions. In file-based mode, it is also possible to specify an ordering over variables to define a PD operator. This is done by simply listing the highest priority variables on the first line, and then progressively lower priority variables on subsequent lines. If we use an ordering that priotizes p over q in the previous example, the output is this:

```
Revised belief set:
(1 and not 2)
```

Note that this is correct: when p has priority, then the revision is no longer a disjunction.

We will see that the advantage of GenC is that it scales to large problems, including many of the benchmark problems in the SATLIB library [Hoos and Stützle, 2000]. But, in terms of the user interface, the simple description above is all the user needs to know to use GenC as a standalone application.

GenC can also be used as a component of larger reasoning systems that involve belief revision. Consider, for example, the robot contoller in Figure 1 that is described in [Hunter *et al.*, 2017]. In this application, agents are controlled on screen by making announcements; the agents revise their beliefs, and move accordingly. However, since the original software essentially uses a brute force revision algorithm, it can only run with 2-3 agents and 2-3 behaviours. By using GenC for the revision portion, this robot controller is able to simulate over 100 agents and behaviours. It is worth noting that this is just one example. GenC could also be used to improve other practical applications of belief revision, such as cryptographic protocol verification [Hunter and Delgrande, 2007], and negotiation [Pilotti *et al.*, 2015].

4 Technical Details

GenC is written in C++, using OpenMP for parallel processing. For testing, we used a computer running Linux Kernel 4.17 with a 4.8GHz CPU.

GenC converts the initial belief set K to a set of bit vectors. The AllSAT solver described in [Toda and Tsuda, 2015] is then used to generate all models of the revision formula ϕ, and these models are also represented by bit vectors. Dedicated hardware instructions are used to calculate the distance between models of K and models of ϕ. GenC then finds the minimally distant models of ϕ, using a suitably weighted Hamming distance. The final output is simplified using the tabular method of reduction, to ensure human-readable output. The entire process is described in more detail in [Hunter and Agapeyev, 2019].

Models	Clauses	Time to Solve (sec)
16	100K	.218
16	5M	8.581
416,492	100K	31
416,492	5M	4430

Table 1: Running Times

GenC has been tested on a range of problems from the SATLIB libary - all of which involve 100 variables. Initially, we wanted to give the running time strictly with respect to the number of clauses in the input formula. However, the performance actually depends both on the number of clauses and the number of models of ϕ (which cannot be predicted in advance). Some example running times are provided in Table 1, for either 16 models or 416,000 models. We remark that these run times are signicantly faster than any existing belief revision software described in the literature.

5 Contribution

There have been a few implemented belief revision solvers in the literature, such as the COBA system [Delgrande *et al.*, 2007] and the GenB system [Hunter and Tsang, 2016]. However, existing systems tend to be very slow on large examples and they often are restricted to isolated belief revision operators. GenC addresses both of these problems.

In terms of running time, GenC is far superior to all existing systems. The fundamental innovation here is clearly the use of the AllSat solver for the computationally hard part of the problem. However, the system is also innovative in other aspects, such as the use of parallel computation and the dedicated hardware instructions for bit comparisons.

In terms of generality, the focus on PD operators here is important; this is a very natural and expressive class of operators. While PD operators can capture many problems, we also note that the main bottleneck in using GenC for arbitrary belief revision operators is actually the difficulty of specification of an arbitrary ordering. In other words, it is hard to give a user interface that allows a user to clearly specify a revision operator. But it is actually easy to modify the source code for GenC to modify the *distance* function, thereby allowing an even wider range of revision operators to be calcuted. Hence, the system is sufficiently flexible to capture other AGM operators if they are given in advance.

6 Conclusion

GenC is a belief revision solver, designed to work efficiently with large sample problems. In fact, the speed at which revision problems are solved by GenC far exceeds any other existing systems that we have seen. The system is explicitly focused on solving PD operators, with an eye towards future applications.

References

[Alchourrón *et al.*, 1985] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 50(2):510–530, 1985.

[Dalal, 1988] Mukesh Dalal. Investigations into a theory of knowledge base revision. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 475–479, 1988.

[Delgrande *et al.*, 2007] James P. Delgrande, Daphne Liu, Torsten Schaub, and Sven Thiele. COBA 2.0: A consistency-based belief change system. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, pages 78–90, 2007.

[Eiter and Gottlob, 1992] Thomas Eiter and Georg Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[Hoos and Stützle, 2000] Holger Hoos and Thomas Stützle. SATLIB: An online resource for research on SAT, 2000.

[Hunter and Agapeyev, 2019] Aaron Hunter and John Agapeyev. An efficient solver for parametrized difference revision. In *Proceedings of the Australasian Conference on Artificial Intelligence*, pages 143–152, 2019.

[Hunter and Delgrande, 2007] Aaron Hunter and James P. Delgrande. Belief change and cryptographic protocol verification. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 427–433, 2007.

[Hunter and Tsang, 2016] Aaron Hunter and Eric Tsang. GenB: A general solver for AGM revision. In *Proceedings of the European Conference on Logics in Artificial Intelligence (JELIA)*, pages 564–569, 2016.

[Hunter *et al.*, 2017] Aaron Hunter, Francois Schwarzentruber, and Eric Tsang. Belief manipulation through propositional announcements. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1109–1115, 2017.

[Peppas and Williams, 2018] Pavlos Peppas and Mary-Anne Williams. Parametrised difference revision. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 277–286, 2018.

[Pilotti *et al.*, 2015] Pablo Pilotti, Ana Casali, and Carlos Iván Chesñevar. A belief revision approach for argumentation-based negotiation agents. *Applied Mathematics and Computer Science*, 25(3):455–470, 2015.

[Toda and Tsuda, 2015] Takahisa Toda and Koji Tsuda. BDD construction for all solutions sat and efficient caching mechanism. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1880–1886, 2015.