Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments

Andreas Herzig
Université de Toulouse, CNRS, IRIT
http://www.irit.fr/~Andreas.Herzig

Abstract

We examine several belief change operations in the light of Dynamic Logic of Propositional Assignments DL-PA. We show that we can encode in a systematic way update operations (such as Winslett's 'Possible Models Approach') and revision operations (such as Dalal's) as particular DL-PA programs. Every DL-PA formula being equivalent to a boolean formula, we obtain syntactical counterparts for all these belief change operations.

Introduction

When a belief base B is modified in order to take into account a new piece of information A then we want to build a new belief base $B \circ A$. In that enterprise at least two reasoning problems are interesting.

- The *belief change problem* is a function problem: find a logical formula representing the modified base $B \circ A$.
- The *belief change consequence problem* (called implication problem in [Eiter and Gottlob, 1992]) is a decision problem: for a given formula C, decide whether C is a consequence of $B \circ A$, i.e., decide whether the implication $(B \circ A) \to C$ is valid.

These problems are omnipresent in computer science, e.g. in databases, reasoning about actions, planning, logic programming, and in semantic web ontologies.

Many belief change operations can be found in the literature. Among the most prominent are Dalal's revision operation [Dalal, 1988], Winslett's 'Possible Models Approach' (PMA) [Winslett, 1988; 1990], Forbus's update operation [Forbus, 1989], and Winslett's Standard Semantics (WSS) [Winslett, 1995]. We refer to [Herzig and Rifi, 1999] for a detailed overview and comparison. According to Katsuno and Mendelzon's classical distinction between update and revision operations, the first three are update operations, while Dalal's is a revision operation [Katsuno and Mendelzon, 1992]. A recent discussion of the distinction between the two concepts and their relationship can be found in [Lang, 2007].

Beyond the above concrete belief change operations, there exist more abstract frameworks. Alchourrón, Gärdenfors and Makinson (AGM) studied the properties that 'good' revision operations should have [Alchourrón, Gärdenfors, and Makinson, 1985]. Similarly, Katsuno and Mendelzon (KM) studied the properties 'good' update operations should have [Katsuno and Mendelzon, 1992]. The difference between revision and update can be illustrated by the example of a new edition of a dictionary that is advertised as being 'updated and revised': revision corrects information that turned out to be wrong (about a static world), while update takes into account new usages of words, i.e., the dynamics of the world.

Dalal's operation satisfies the AGM postulates for revision, and Winslett's and Forbus's operations satisfy the KM postulates for update. This is however not enough to *characterise* these operations: they validate properties beyond the AGM/KM postulates, such as Parikh's relevance postulate that is based on language splitting [Parikh, 1999; Kourousias and Makinson, 2007]. Only few papers investigated this issue. Similarly, the related problem of syntactically constructing the new belief base $B \circ A$ has only received little attention. Indeed, the above concrete belief change operations were almost exclusively studied from a semantical perspective: instead of viewing $B \circ A$ as a formula, it is viewed as a set of models of classical propositional logic, alias a set of valuations. Therefore $\circ$ is not part of the object language but is in the metalanguage. We call $\circ$ a metalanguage operat*ion*, as opposed to object language operat*or*, i.e., a logical connective.

If the set of propositional variables $\mathbb{P}$ is finite then there is an easy recipe to construct a formula representing $B \circ A$: describe each valuation $v \in B \circ A$ by the conjunction of literals $\mathsf{Fml}(v) = \left(\bigwedge_{p \in v} p\right) \land \left(\bigwedge_{p \in \mathbb{P} \setminus v} \neg p\right)$ and take the disjunction of these big conjunctions $\mathsf{Fml}(v)$: the set of models of $\bigvee_{v \in B \circ A} \mathsf{Fml}(v)$ equals $B \circ A$, and therefore the former is the syntactical counterpart of the latter.

Is there a better, syntactic procedure building the new belief base? The answer is positive for WSS, which is the simplest update operation: an update procedure based on variable forgetting is in [Herzig and Rifi, 1999] and its generalisation to literal forgetting is in [Herzig, Lang, and Marquis, 2013]. As to the PMA, there is an axiomatisation in [Herzig, 1996] that can be turned into a decision procedure for the belief change consequence problem; it is however based on formulas in disjunctive normal form and is as such very close to the semantics. Finally, there are syntactical belief change operations that are based on the computation of prime implicants and prime implicates and do not have semantics [Bienvenu, Herzig, and Qi, 2008; Marchi, Bittencourt, and Perrussel, 2010].

In this paper we propose a powerful yet simple logical framework for belief change: Dynamic Logic of Propositional Assignments, abbreviated DL-PA [Herzig et al., 2011; Balbiani, Herzig, and Troquard, 2013]. DL-PA is a simple instantiation of Propositional Dynamic Logic PDL [Harel, 1984; Harel, Kozen, and Tiuryn, 2000]: instead of PDL's abstract atomic programs, its atomic programs are assignments of propositional variables to either true or false, written $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. Just as in PDL, these atomic programs can be combined by means of program operators: sequential and nondeterministic composition, finite iteration, and test. In the present paper we moreover make use of a program operator that is less frequently considered in PDL, namely the converse operator. While DL-PA programs describe the evolution of the world, DL-PA formulas describe the state of the world. In particular, formulas of the form $\langle\pi\rangle\varphi$ express that φ is true after *some* possible execution of π, and $[\pi]\varphi$ expresses that φ is true after *every* possible execution of π.

The models of DL-PA are considerably simpler than PDL's Kripke models: valuations of classical propositional logic are enough. The assignment $p{\leftarrow}\top$ updates the current valuation by p, while the assignment $p{\leftarrow}\bot$ updates it by $\neg p$. For example, the program $\mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot$ updates the current valuation by the fact that Ann is not an employee of the CS department. The other way round, the program $(\mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot)^-$ undoes that update. Consider the formula

$$\langle \mathsf{emp}_{\mathsf{Ann,CS}}{\leftarrow}\bot^-\rangle(\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}})$$

It says that possibly, before Ann was fired from the CS department, both she and Bob were employees there. This means that it is possible that the belief base $\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}}$ has been updated by $\neg\mathsf{emp}_{\mathsf{Ann,CS}}$, resulting in the current belief base. The current base is therefore nothing but

$$(\mathsf{emp}_{\mathsf{Ann,CS}} \wedge \mathsf{emp}_{\mathsf{Bob,CS}}) \circ \neg\mathsf{emp}_{\mathsf{Ann,CS}}$$

This can be generalised: we have

$$B \circ p = ||\langle (p{\leftarrow}\top)^-\rangle B||$$
$$B \circ \neg p = ||\langle (p{\leftarrow}\bot)^-\rangle B||$$

for every propositional variable p and each of the update operations among PMA, WSS, and Forbus, where the function $||.||$ associates to each formula the set of valuations where it is true. (The corresponding equality for Dalal's revision operation is a bit more involved and will be given later.)

The first contribution of the present paper is to go beyond such updates by literals and to associate to every input formula A a program π_A implementing the update by A. Precisely, these programs π_A depend on the belief change operation under concern, and WSS, PMA and Forbus's operation will be implemented by different programs π_A^{wss}, π_A^{forbus}, and π_A^{pma}. For instance, we will show that the WSS update by the disjunction $p{\vee}q$ is captured by the program

$$\pi_{p\vee q}^{\mathsf{wss}} = (p{\leftarrow}\top \cup p{\leftarrow}\bot); (q{\leftarrow}\top \cup q{\leftarrow}\bot); p{\vee}q?$$

which is equivalent to

$$(p{\leftarrow}\top; q{\leftarrow}\top) \cup (p{\leftarrow}\top; q{\leftarrow}\bot) \cup (p{\leftarrow}\bot; q{\leftarrow}\top).$$

We establish that being in a state where B was updated by A is the same as being in a state where B was true before the update program π_A was executed. Formally, we prove that

$$B \circ A \;=\; ||\langle (\pi_A)^-\rangle B||$$

for each of the belief change operations $\circ$ and associated family of programs π_A that we consider: WSS, PMA, Forbus, and Dalal.[1]

It is shown in [Herzig et al., 2011; Balbiani, Herzig, and Troquard, 2013] that every DL-PA formula can be reduced to an equivalent propositional formula. It is shown in the latter paper that the reduction can be extended to the converse operator. We thereby obtain a syntactical representation of the modified belief base in terms of a boolean formula. This is the second contribution of our paper.

In the next two sections we settle some notations and introduce DL-PA. In the remaining sections we embed several belief change semantics into DL-PA: Winslett's WSS, Winslett's PMA, and Forbus's update operation, and finally Dalal's belief revision operation.

Background: propositional logic

Here are some notations and conventions for propositional logic, as well as the definitions of symmetric difference and Hamming distance between propositional valuations.

A *valuation* associates a truth value to each propositional variable in the countable set $\mathbb{P} = \{p, q, \ldots\}$. We identify valuations with subsets of $\mathbb{P}$ and use v, v_1, v_2, etc. to denote them. The set of all valuations is $\mathbb{V} = 2^{\mathbb{P}}$. Sometimes we write $v(p) = 1$ when $p \in v$ and $v(p) = 0$ when $p \notin v$.

The *symmetric difference* between two valuations v_1 and v_2 is the set of all those p such that $v_1(p) \neq v_2(p)$. Formally, it is defined as

$$v_1 \dot{-} v_2 = (v_1 \setminus v_2) \cup (v_2 \setminus v_1)$$

For example, $\{p, q\} \dot{-} \{q, r, s\} = \{p, r, s\}$.

The *Hamming distance* between v_1 and v_2 is is the number of all those p such that $v_1(p) \neq v_2(p)$. Formally, it is defined to be $\mathsf{card}(v_1 \dot{-} v_2)$, i.e., the cardinality of the symmetric difference between v_1 and v_2. For example, the Hamming distance between $\{p, q\}$ and $\{q, r, s\}$ is $\mathsf{card}(\{p, r, s\}) = 3$. Note that the Hamming distance might be infinite: for instance, $\mathsf{card}(\emptyset \dot{-} \mathbb{P}) = \infty$.

Boolean formulas (also called *propositional formulas*) are built from propositional variables by means of the standard boolean connectives. We will in particular make use of the exclusive disjunction, denoted by $\oplus$. Boolean formulas are

[1]In the case of Dalal's revision operation, the program actually also depends on the base B. This is as it should be, because AGM revision operations involve checking whether $A{\wedge}B$ is satisfiable.

denoted by A, B, C, etc. Contrasting with that, *modal formulas*—to be defined in the next section—will be denoted by φ, ψ, etc. For a given boolean formula A, the set of variables occurring in A is denoted by $\mathbb{P}_A$. For example, $\mathbb{P}_{p\vee\neg q} = \{p, q\}$.

A given valuation determines the truth value of each boolean formula. An *A-valuation* is a valuation where the boolean formula A is true.

Dynamic Logic of Propositional Assignments

In this section we define syntax and semantics of Dynamic Logic of Propositional Assignments DL-PA and state decidability and complexity results. Based on earlier work by Tiomkin and Makowski and by van Eijck [Tiomkin and Makowsky, 1985; van Eijck, 2000], star-free DL-PA was studied in [Herzig et al., 2011] and full DL-PA in [Balbiani, Herzig, and Troquard, 2013].

Language The language of DL-PA is defined by the following grammar:

$$\begin{array}{lll} \varphi & ::= & p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\pi\rangle\varphi \\ \pi & ::= & p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \pi;\pi \mid \pi \cup \pi \mid \pi^* \mid \pi^- \mid \varphi? \end{array}$$

where p ranges over $\mathbb{P}$. So the *atomic programs* of the language of DL-PA are of the form $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. The operators of sequential composition (";"), nondeterministic composition ("$\cup$"), finite iteration ("$(.)^*$", the so-called Kleene star), and test ("$(.)?$") are familiar from PDL. The operator "$(.)^-$" is the converse operator. The *star-free fragment* of DL-PA is the subset of the language made up of formulas without the Kleene star "$(.)^*$".

The set of variables occurring in φ is denoted by $\mathbb{P}_\varphi$, just as for boolean formulas. The *length* of a formula φ, denoted by $|\varphi|$, is the number of symbols used to write down φ, without "$\langle$", "$\rangle$", and parentheses. For example, $|\langle q{\leftarrow}\top\rangle(q \vee r)| = 3{+}3 = 6$. The length of a program π, denoted by $|\pi|$, is defined in the same way. For example, $|p{\leftarrow}\bot; p?| = 6$. We have $\mathsf{card}(\mathbb{P}_\varphi) \le |\varphi|$ for every φ.

We abbreviate the logical connectives $\wedge$, $\to$, $\leftrightarrow$, and $\oplus$ in the usual way. Moreover, $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. Several program abbreviations are familiar from PDL. First, skip abbreviates $\top?$ ("nothing happens"). Second, the conditional "if φ then π_1 else π_2" is expressed by $(\varphi?;\pi_1) \cup (\neg\varphi?;\pi_2)$. Third, the loop "while φ do π" is expressed by $(\varphi?;\pi)^*;\neg\varphi?$. Let us moreover recursively define the n-th iteration of π and the iteration up to n of π, for $n \ge 0$:

$$\pi^n = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ \pi;\pi^{n-1} & \text{if } n \ge 1 \end{cases}$$

$$\pi^{\le n} = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ (\mathsf{skip}\cup\pi);\pi^{\le n-1} & \text{if } n \ge 1 \end{cases}$$

Finally, we introduce assignments of literals to variables by means of the following two abbreviations:

$$\begin{aligned} p{\leftarrow}q &= \text{if } q \text{ then } p{\leftarrow}\top \text{ else } p{\leftarrow}\bot \\ &= (q?;p{\leftarrow}\top) \cup (\neg q?;p{\leftarrow}\bot) \\ p{\leftarrow}\neg q &= \text{if } q \text{ then } p{\leftarrow}\bot \text{ else } p{\leftarrow}\top \\ &= (q?;p{\leftarrow}\bot) \cup (\neg q?;p{\leftarrow}\top) \end{aligned}$$

$$\begin{aligned} ||p|| &= \{v \ : \ p \in v\} \\ ||\top|| &= \mathbb{V} \ = \ 2^{\mathbb{P}} \\ ||\bot|| &= \emptyset \\ ||\neg\varphi|| &= 2^{\mathbb{P}} \setminus ||\varphi|| \\ ||\varphi \vee \psi|| &= ||\varphi|| \cup ||\psi|| \\ ||\langle\pi\rangle\varphi|| &= \{v \ : \ \text{there is } v_1 \text{ s.t. } (v, v_1) \in ||\pi|| \text{ and } v_1 \in ||\varphi||\} \\ ||p{\leftarrow}\top|| &= \{(v_1, v_2) \ : \ v_2 = v_1 \cup \{p\}\} \\ ||p{\leftarrow}\bot|| &= \{(v_1, v_2) \ : \ v_2 = v_1 \setminus \{p\}\} \\ ||\pi;\pi'|| &= ||\pi|| \circ ||\pi'|| \\ ||\pi \cup \pi'|| &= ||\pi|| \cup ||\pi'|| \\ ||\pi^*|| &= \bigcup_{k\in\mathbb{N}_0} (||\pi||)^k \\ ||\pi^-|| &= ||\pi||^{-1} \\ ||\varphi?|| &= \{(v, v) \ : \ v \in ||\varphi||\} \end{aligned}$$

Table 1: Interpretation of the DL-PA connectives.

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. In particular, the program $p{\leftarrow}\neg p$ flips the truth value of p. Note that the abbreviations $p{\leftarrow}q$ and $p{\leftarrow}\neg q$ have constant length (namely 14).

Semantics DL-PA programs are interpreted by means of a (unique) *relation between valuations*. The atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ update valuations in the obvious way, and complex programs are interpreted just as in PDL by mutual recursion. Table 1 gives the interpretation of the DL-PA connectives.

Two formulas φ_1 and φ_2 are *formula equivalent* if $||\varphi_1|| = ||\varphi_2||$. Two programs π_1 and π_2 are *program equivalent* if $||\pi_1|| = ||\pi_2||$. For example, $\mathsf{skip};\pi$ is program equivalent to π, and $\mathsf{skip} \cup p{\leftarrow}\neg p$ is program equivalent to $p{\leftarrow}\top \cup p{\leftarrow}\bot$. We write $\pi_1 \equiv \pi_2$ when π_1 and π_2 are progam equivalent.

Proposition 1 (Proposition 10 in [Balbiani, Herzig, and Troquard, 2013])**.** *The following program equivalences are* DL-PA *valid:*

$$\begin{aligned} p?;p{\leftarrow}\top &\equiv p? \\ \neg p?;p{\leftarrow}\bot &\equiv \neg p? \\ p{\leftarrow}\tau;p{\leftarrow}\top &\equiv p{\leftarrow}\top \\ p{\leftarrow}\tau;p{\leftarrow}\bot &\equiv p{\leftarrow}\bot \\ p{\leftarrow}\tau;q{\leftarrow}\top &\equiv q{\leftarrow}\top;p{\leftarrow}\tau \qquad \textit{if } p \neq q \\ p{\leftarrow}\tau;q{\leftarrow}\bot &\equiv q{\leftarrow}\bot;p{\leftarrow}\tau \qquad \textit{if } p \neq q \end{aligned}$$

where τ is a placeholder for either true or false.

An *expression* is a formula or a program. When we say that two expressions are equivalent we mean program equivalence if we are talking about programs, and formula equivalence otherwise. Equivalence is preserved under replacement of a sub-expression by an equivalent expression [Balbiani, Herzig, and Troquard, 2013, Proposition 7].

A formula φ is DL-PA *valid* if it is formula equivalent to $\top$, i.e., if $||\varphi|| = 2^{\mathbb{P}}$. It is DL-PA *satisfiable* if it is not formula equivalent to $\bot$, i.e., if $||\varphi|| \neq \emptyset$. For example, the formulas $\langle p{\leftarrow}\bot\rangle\top$ and $\langle p{\leftarrow}\top\rangle\varphi \leftrightarrow \neg\langle p{\leftarrow}\top\rangle\neg\varphi$ are DL-PA valid. Other examples of DL-PA validities are $\langle p{\leftarrow}\top\rangle p$ and $\langle p{\leftarrow}\bot\rangle\neg p$. The valid schemas $\varphi \to [\pi]\langle\pi^-\rangle\varphi$ and $\varphi \to [\pi^-]\langle\pi\rangle\varphi$ are inherited from converse PDL; they are called the *conversion axioms*. Moreover, $\varphi \to [\pi]\varphi$ is valid if and only if $\langle\pi^-\rangle\varphi \to \varphi$ is valid. The two senses of the "if and only if" correspond to the two so-called *conversion rules*.

Observe that if p does not occur in φ then both $\varphi \to \langle p{\leftarrow}\top\rangle\varphi$ and $\varphi \to \langle p{\leftarrow}\bot\rangle\varphi$ are valid. This is due to the following semantical property that is instrumental in the proof of several results in the rest of the paper.

Proposition 2. *Suppose $\mathbb{P}_\varphi \cap P = \emptyset$, i.e., none of the variables in P occurs in φ. Then $v \cup P \in ||\varphi||$ iff $v \setminus P \in ||\varphi||$.*

Reduction to propositional logic In PDL, all program operators but the Kleene star can be eliminated. In contrast, *all* program operators can be eliminated in DL-PA. We describe this in detail now.

The first step is to eliminate the converse operator. Table 2 lists the program equivalences by means of which we can eliminate the converse operator with only linear increase in program length.[2]

Proposition 3. *For every program π there is an equivalent π' such that no converse operator occurs in π' and such that the length of π' is linear in the length of π.*

For example, the converse of assignments of literals to variables are reduced as follows:

$$p{\leftarrow}q^- \equiv \begin{cases} \mathsf{skip} & \text{if } q = p \\ p{\leftrightarrow}q?; (p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise} \end{cases}$$

$$p{\leftarrow}\neg q^- \equiv \begin{cases} p{\leftarrow}\neg p & \text{if } q = p \\ p{\oplus}q?; (p{\leftarrow}\top \cup p{\leftarrow}\bot) & \text{otherwise} \end{cases}$$

The second step is to eliminate the Kleene star.

Proposition 4 (Theorem 1 of [Balbiani, Herzig, and Troquard, 2013])**.** *For every converse-free π there is an equivalent π' such that no Kleene star occurs in π'.*

The resulting formula is in star-free PDL, for which it is known that all the program operators can be eliminated. Once this has been done, modal operators only contain atomic programs. Such modal operators are both serial and deterministic and therefore distribute over negation and disjunction: $\langle p{\leftarrow}\bot\rangle\neg\varphi$ is equivalent to $\neg\langle p{\leftarrow}\bot\rangle\varphi$, and $\langle p{\leftarrow}\bot\rangle(\varphi_1 \vee \varphi_2)$ is equivalent to $\langle p{\leftarrow}\bot\rangle\varphi_1 \vee \langle p{\leftarrow}\bot\rangle\varphi_2$. They can finally be eliminated when they face a propositional variable, according to the following formula equivalences:

$$\langle p{\leftarrow}\top\rangle q \leftrightarrow \begin{cases} \top & \text{if } q = p \\ q & \text{otherwise} \end{cases}$$

$$\langle p{\leftarrow}\bot\rangle q \leftrightarrow \begin{cases} \bot & \text{if } q = p \\ q & \text{otherwise} \end{cases}$$

For example, $\langle p{\leftarrow}\bot\rangle p \vee \langle p{\leftarrow}\bot\rangle\neg r$ is equivalent to $\bot \vee \neg r$.

Together, these three steps make up a complete set of reduction axioms: for every formula there is an equivalent boolean formula.

$$\begin{aligned} p{\leftarrow}\top^- &\equiv p?; (\mathsf{skip} \cup p{\leftarrow}\bot) \\ &\equiv p? \cup (p?; p{\leftarrow}\bot) \\ p{\leftarrow}\bot^- &\equiv \neg p?; (\mathsf{skip} \cup p{\leftarrow}\top) \\ &\equiv \neg p? \cup (\neg p?; p{\leftarrow}\top) \\ (\pi_1; \pi_2)^- &\equiv \pi_2^-; \pi_1^- \\ (\pi_1 \cup \pi_2)^- &\equiv \pi_1^- \cup \pi_2^- \\ (\pi^*)^- &\equiv (\pi^-)^* \\ (\varphi?)^- &\equiv \varphi? \end{aligned}$$

Table 2: Reduction axioms for the converse operator.

Complexity In this paper we only need complexity results for star-free DL-PA. The complexity of both model checking and satisfiability checking for the converse-free and star-free fragment of the language of DL-PA is shown to be PSPACE complete in [Herzig et al., 2011]. By Proposition 3 these results transfer to star-free DL-PA with the converse operator.

Some useful DL-PA programs Table 3 collects some DL-PA programs that are going to be convenient in our enterprise. In that table, P is the set of variables $\{p_1, \ldots, p_n\}$, and P' is the set of variables p'_k such that p_k is in P and p'_k is *fresh*: we suppose that p'_k does not occur in P (or rather, in the formula under consideration). In order to alleviate notation we drop set parentheses and write $\mathsf{flip}^1(p)$ instead of $\mathsf{flip}^1(\{p\})$, etc. Note that the length of each program of Table 3 is linear in the cardinality of the set P.

The first two programs of Table 3 change the truth values of some of the variables in the set P. The program $\mathsf{flip}^1(P)$ flips some nondeterministically chosen variable of P. So the iteration $\mathsf{flip}^1(P)^n$ does this n times and $\mathsf{flip}^1(P)^{\leq n}$ does this

[2]This simplifies the original proof of [Balbiani, Herzig, and Troquard, 2013, Section VII.B], which instead of directly stating a reduction axiom for the Kleene star rather relies on its preliminary elimination.

We might as well apply de Giacomo's general elimination technique[de Giacomo, 1996]. It is however more complicated than ours. He proceeds by first applying the four last reduction axioms of Table 2 to the original formula φ_0 in order to push down the converse operator. For the resulting formula φ_1, a theory Γ_{φ_1} is built that is made up of instances of the conversion axioms for all symbols occurring in φ and that are prefixed by a master modality relative to the symbols of φ. Finally, it is proved that the original φ_0 is a theorem if and only if $\Gamma \models \varphi'_1$, where φ'_1 is obtained from φ_1 by considering the converse of atomic programs as new program names (so φ'_1 is without converse operators). Overall, this is less direct and not as simple as the present reduction axioms.

$$\begin{aligned}
\mathsf{flip}^1(P) &= p_1{\leftarrow}\neg p_1 \cup \cdots \cup p_n{\leftarrow}\neg p_n\\
\mathsf{flip}^{\geq 0}(P) &= (p_1{\leftarrow}\top \cup p_1{\leftarrow}\bot); \cdots ; (p_n{\leftarrow}\top \cup p_n{\leftarrow}\bot)\\
\mathsf{store}(P) &= p'_1{\leftarrow}p_1; \cdots ; p'_n{\leftarrow}p_n\\
\mathsf{restore}^1(P) &= (p_1{\oplus}p'_1?; p_1{\leftarrow}p'_1) \cup \cdots \cup (p_n{\oplus}p'_n?; p_n{\leftarrow}p'_n)\\
\mathsf{restore}^{\geq 0}(P) &= (\mathsf{skip} \cup p_1{\leftarrow}p'_1); \cdots ; (\mathsf{skip} \cup p_n{\leftarrow}p'_n)
\end{aligned}$$

Table 3: Some useful DL-PA programs, where $P = \{p_1, \ldots, p_n\}$. For $n = 0$ all programs are supposed to equal skip. In store(P), restore$^1(P)$, and restore$^{\geq 0}(P)$, the p'_i are supposed to be fresh.

at most n times. For example, we have

$$\begin{aligned}
\mathsf{flip}^1(p)^1 &= p{\leftarrow}\neg p; \mathsf{skip}\\
&\equiv p{\leftarrow}\neg p\\
\mathsf{flip}^1(p)^{\leq 1} &= (\mathsf{skip} \cup p{\leftarrow}\neg p); \mathsf{skip}\\
&\equiv p{\leftarrow}\top \cup p{\leftarrow}\bot\\
&= \mathsf{flip}^{\geq 0}(p)
\end{aligned}$$

The program flip$^{\geq 0}(P)$ nondeterministically changes the truth value of some of the variables in P. It is program equivalent to flip$^1(P)^{\leq n}$ while being shorter.[3] It actually implements the operation of *forgetting* the values of the variables in P [Lang, Liberatore, and Marquis, 2003]. Note that flip$^{\geq 0}(\mathbb{P}_\varphi); \varphi?$ relates any valuation v to all φ-valuations where the variables outside φ have the same truth values.

The last three programs of Table 3 store or restore some of the values of the variables of P. The program store(P) stores the truth value of each variable p_k of P by means of a fresh variable p'_k. When the truth values of p_k and p'_k differ then we say that p_k is marked; else we say that p_k is unmarked. The program restore$^1(P)$ restores one of the marked variables p_k of P: after it, some p_k has the truth value of p'_k. Finally, restore$^{\geq 0}(P)$ restores zero or more marked variables of P.

Proposition 5. *The following hold:*

1. $(v_1, v_2) \in ||\mathsf{flip}^1(p)||$ *iff* $\mathsf{card}(v_1 \dot{-} v_2) = 1$.
2. $(v_1, v_2) \in ||\mathsf{flip}^1(P)^{\leq n}||$ *iff* $\mathsf{card}(v_1 \dot{-} v_2) \leq n$.
3. $(v_1, v_2) \in ||\mathsf{flip}^{\geq 0}(P)||$ *iff* $v_1 \dot{-} v_2 \subseteq P$.
4. $(v_1, v_2) \in ||\mathsf{store}(P)||$ *iff* $v_2 = (v_1 \setminus P') \cup \{p' : p \in v_1 \cap P\}$.
5. $(v_1, v_2) \in ||\mathsf{restore}^1(P)||$ *iff there is* $p_k \in P$ *such that* $v_1 \dot{-} v_2 = \{p_k\}$ *and* $v_1(p_k) \neq v_1(p'_k)$.
6. $(v_1, v_2) \in ||\mathsf{restore}^{\geq 0}(P)||$ *iff there is* $S \subseteq P$ *such that* $v_1 \dot{-} v_2 = S$ *and* $v_1(p_k) \neq v_1(p'_k)$ *for every* $p_k \in S$.

Note that flip$^1(P)^n$ may flip less than exactly n variables: for example, flip$^1(P)^2$ may flip the same variable twice and in that case has the same effect as skip. So $(v_1, v_2) \in ||\mathsf{flip}^1(P)^n||$ does not imply that the Hammming distance between v_1 and v_2 is n. But what matters is that in Proposition 8 below (and Theorem 3 that is based on it), the occurrence of flip$^1(\mathbb{P}_A)^m$ is preceded by the test H$(A, \geq m)?$, which when successfully executed guarantee that the Hamming distance is n.

[3]The length of flip$^1(P)^n$ and flip$^1(P)^{\leq n}$ is quadratic in n. The length of the latter would have been cubic had we adopted the somewhat more natural definition $\bigcup_{k\leq n} \mathsf{flip}^1(P)^k$.

$$\begin{aligned}
\mathsf{Valid}(\varphi) &= [\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi)]\varphi\\
\mathsf{Sat}(\varphi) &= \langle \mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi)\rangle\varphi\\
\mathsf{D}(\varphi, P) &= \langle \mathsf{flip}^{\geq 0}(P)\rangle\varphi \land \neg\langle \bigcup_{p\in P} \mathsf{flip}^{\geq 0}(P \setminus \{p\})\rangle\varphi\\
\mathsf{H}(\varphi, \geq m) &= \begin{cases} \top & \text{if } m = 0\\ \neg\langle \mathsf{flip}^1(\mathbb{P}_\varphi)^{\leq m-1}\rangle\varphi & \text{if } m \geq 1 \end{cases}
\end{aligned}$$

Table 4: Some useful DL-PA formulas, where φ is a formula, $P \subseteq \mathbb{P}_\varphi$ is a nonempty set of propositional variables, and $m \leq \mathsf{card}(\mathbb{P}_\varphi)$ is a non-negative integer.

Proposition 6. *The following programs are equivalent:*

$$\begin{aligned}
(\mathsf{flip}^{\geq 0}(P))^- &\equiv \mathsf{flip}^{\geq 0}(P)\\
(\mathsf{flip}^1(P)^n)^- &\equiv \mathsf{flip}^1(P)^n\\
(\mathsf{flip}^1(P)^{\leq n})^- &\equiv \mathsf{flip}^1(P)^{\leq n}
\end{aligned}$$

Some useful DL-PA formulas Table 4 collects some interesting DL-PA formulas: Valid(φ) expresses that the formula φ is valid and Sat(φ) expresses that φ is satisfiable. The former is equivalent to $[\mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi); \varphi?]\top$ and the latter is equivalent to $\langle \mathsf{flip}^{\geq 0}(\mathbb{P}_\varphi); \varphi?\rangle\top$. The formula D$(\varphi, P)$ is true at a valuation v exactly when there is a closest φ-valuation whose symmetric difference with v is P. The formula H$(\varphi, \geq m)$ is true at a valuation v exactly when the closest φ-valuations in the sense of the Hamming distance differ in at least m variables from v. For example:

$$\begin{aligned}
\mathsf{H}(p, \geq 1) &= \neg\langle \mathsf{flip}^1(p)^{\leq 0}\rangle p\\
&= \neg\langle \mathsf{skip}\rangle p\\
&\leftrightarrow \neg p\\
\mathsf{H}(p \lor q, \geq 1) &= \neg\langle \mathsf{flip}^1(p, q)^{\leq 0}\rangle(p \lor q)\\
&\leftrightarrow \neg p \land \neg q\\
\mathsf{H}(p \lor q, \geq 2) &= \neg\langle \mathsf{flip}^1(p, q)^{\leq 1}\rangle(p \lor q)\\
&\leftrightarrow \neg\langle \mathsf{skip} \cup p{\leftarrow}\neg p \cup q{\leftarrow}\neg q\rangle(p \lor q)\\
&\leftrightarrow \neg(p \lor q \lor \langle p{\leftarrow}\neg p\rangle(p \lor q) \land \langle q{\leftarrow}\neg q\rangle(p \lor q))\\
&\leftrightarrow \neg((p \lor q) \lor (\neg p \lor q) \lor (p \lor \neg q))\\
&\leftrightarrow \bot
\end{aligned}$$

The first two items of the next proposition show that both validity checking and satisfiability checking can both be reduced to model checking.

Proposition 7. *Let v a valuation and φ a formula.*

1. $v \in ||\mathsf{Valid}(\varphi)||$ *iff φ is valid.*

2. $v \in ||\mathsf{Sat}(\varphi)||$ *iff φ is satisfiable.*

3. $v \in ||\mathsf{D}(\varphi, P)||$ *iff there is a valuation* $v_1 \in ||\varphi||$ *such that* $v \dot{-} v_1 = P$ *and there is no valuation* $v_2 \in ||\varphi||$ *such that* $v \dot{-} v_2 \subset P$.

4. $v \in ||\mathsf{H}(\varphi, \geq m)||$ *iff there is no valuation* $v_1 \in ||\varphi||$ *such that* $\mathsf{card}(v \dot{-} v_1) < m$.

The length of each formula of Table 4 is polynomial in the length of φ. (For $\mathsf{Sat}(\varphi)$ it is linear, for $\mathsf{D}(\varphi, P)$ and $\mathsf{H}(\varphi, \geq m)$ it is quadratic.)

From the above programs we can build a program which checks whether the Hamming distance to the closest A-valuation is m.

Proposition 8. *Let* $m \leq \mathsf{card}(\mathbb{P}_A)$. *Then*

$$(v_1, v_2) \in ||\mathsf{H}(A, \geq m)?; \mathsf{flip}^1(\mathbb{P}_A)^m; A||$$

if and only if $v_2 \in ||A||$, $\mathsf{card}(v_1 \dot{-} v_2) = m$, *and there is no* $v_2' \in ||A||$ *such that* $\mathsf{card}(v_1 \dot{-} v_2') < m$.

WSS

Winslett's standard semantics (WSS) [Winslett, 1995] is a syntax-dependent update operation that is less conservative than the other operations that we are going to examine, in the sense that each of the other operations preserves at least as much information of the original knowledge base as the WSS.

Semantics of the WSS Let v be a valuation and A be a propositional formula. The *WSS update of v by A* is

$$\begin{aligned} v \diamond^{\mathsf{wss}} A &= \{v_1 \in ||A|| \ : \ v \dot{-} v_1 \subseteq \mathbb{P}_A\} \\ &= \{v_1 \in ||A|| \ : \ \text{for every } p \notin \mathbb{P}_A, v(p) = v_1(p)\} \end{aligned}$$

So the set $v \diamond^{\mathsf{wss}} A$ is the set of A-valuations that agree with v on all the variables that do not occur in A. For example,

$$\emptyset \diamond^{\mathsf{wss}} p \vee q \ = \ \{\{p\}, \{q\}, \{p, q\}\}$$

The WSS update operation is syntax-dependent: depending on their language, logically equivalent formulas A_1 and A_2 may update the same valuation v in different ways.

Let A and B be propositional formulas. The *WSS update of B by A* is obtained by collecting the updates of every B-valuation by A:

$$B \diamond^{\mathsf{wss}} A = \bigcup_{v \in ||B||} v \diamond^{\mathsf{wss}} A$$

The operation $\diamond^{\mathsf{wss}}$ therefore takes two propositional formulas and returns a set of valuations. For example:

$$\begin{aligned} \neg p \wedge \neg q \diamond^{\mathsf{wss}} p &= ||p \wedge \neg q|| \\ \neg p \wedge \neg q \diamond^{\mathsf{wss}} p \vee q &= ||p \vee q|| \\ \neg q \diamond^{\mathsf{wss}} p &= ||p \wedge \neg q|| \\ \neg p \vee \neg q \diamond^{\mathsf{wss}} p &= ||p|| \end{aligned}$$

Expressing the WSS operation in DL-PA Let us turn to the problem of expressing WSS updates in DL-PA.

Theorem 1. *Let A, B be propositional formulas. Let* π_A^{wss} *be the* DL-PA *program*

$$\mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?$$

Then $B \diamond^{\mathsf{wss}} A = ||\langle (\pi_A^{\mathsf{wss}})^- \rangle B||$.

Via the program equivalences for the converse operator of Proposition 6, it follows that $B \diamond^{\mathsf{wss}} A = ||\langle A?; \mathsf{flip}^{\geq 0}(\mathbb{P}_A) \rangle B||$.

For example, consider the input formula $A = p$. Then

$$\begin{aligned} \pi_p^{\mathsf{wss}} &= \mathsf{flip}^{\geq 0}(P); p? \\ &= (p \leftarrow \top \cup p \leftarrow \bot); p? \\ &\equiv (p \leftarrow \top; p?) \cup (p \leftarrow \bot; p?) \\ &\equiv p \leftarrow \top \\ (\pi_p^{\mathsf{wss}})^- &\equiv (p \leftarrow \top)^- \\ &\equiv p? \cup (p?; p \leftarrow \bot) \end{aligned}$$

For the base $B = \neg p \wedge q$ and the input $A = p$ we have:

$$\begin{aligned} \langle (\pi_A^{\mathsf{wss}})^- \rangle B &\leftrightarrow \langle p?; (p \leftarrow \top \cup p \leftarrow \bot) \rangle (\neg p \wedge q) \\ &\leftrightarrow p \wedge (\langle p \leftarrow \top \rangle (\neg p \wedge q) \vee \langle p \leftarrow \bot \rangle (\neg p \wedge q)) \\ &\leftrightarrow p \wedge (\bot \vee (\top \wedge q)) \\ &\leftrightarrow p \wedge q \end{aligned}$$

So the update of $\neg p \wedge q$ by p is equivalent to $p \wedge q$, as expected.

Observe that the length of π_A^{wss} is linear in the length of A.

PMA

Winslett's possible models approach (PMA) operation [Winslett, 1988; 1990] is the belief update operation that is most cited and used, e.g. in database theory [Winslett, 1995; Chomicki and Marcinkowski, 2004], reasoning about actions and planning [Baral, 1995], logic programming [Slota and Leite, 2010] and in description logics and the semantic web [Baader et al., 2005; Liu et al., 2006; 2011].

The PMA is more conservative than the WSS, in the sense that it preserves more information from the base B. This is achieved by a principle of minimal change.

Semantics of the PMA operation Let v be a valuation and A be a propositional formula. The *PMA update of v by a formula A* is

$$v \diamond^{\mathsf{pma}} A = \{v_1 \in ||A|| \ : \ \text{there is no } v_2 \in ||A|| \text{ s.t. } v \dot{-} v_2 \subset v \dot{-} v_1\}$$

So the set $v \diamond^{\mathsf{pma}} A$ is the set of A-valuations that are closest to v w.r.t. symmetric difference. For example,

$$\begin{aligned} \emptyset \diamond^{\mathsf{pma}} p \vee q &= \{\{p\}, \{q\}\} \\ \emptyset \diamond^{\mathsf{pma}} (p \wedge q) \vee r &= \{\{p, q\}, \{r\}\} \end{aligned}$$

Let A and B be propositional formulas. Just as all the update operations *à la* Katsuno-Mendelzon [Katsuno and Mendelzon, 1992], the *PMA update of B by A* collects the updates of each B-valuation by A:

$$B \diamond^{\mathsf{pma}} A = \bigcup_{v \in ||B||} v \diamond^{\mathsf{pma}} A$$

For example:

$$\begin{aligned}
\neg p \land \neg q \diamond^{\mathsf{pma}} p &= ||p \land \neg q||\\
\neg q \diamond^{\mathsf{pma}} p &= ||p \land \neg q||\\
\neg p \land \neg q \diamond^{\mathsf{pma}} p \lor q &= ||p \oplus q||\\
\neg p \land \neg q \land \neg r \diamond^{\mathsf{pma}} (p \land q) \lor r &= ||(p \land q) \oplus r||
\end{aligned}$$

Expressing the PMA operation in DL-PA We now polynomially translate the update problem $B \diamond^{\mathsf{pma}} A$ into DL-PA: we define a family of update programs π_A^{pma} whose length is linear in the length of A.

It is tempting to define π_A^{pma} to be the program $\mathsf{while}\ \neg A\ \mathsf{do}\ \mathsf{flip}^1(\mathbb{P}_A)$, where while is as defined above. However, it does not have the exact behaviour of $\diamond^{\mathsf{pma}}$. Indeed, the interpretation of

$$\mathsf{while}\ \neg((p \land q) \lor r)\ \mathsf{do}\ \mathsf{flip}^1(\{p, q, r\})$$

relates the empty valuation to the four valuations $\{p, q\}$, $\{r\}$, $\{p, r\}$, and $\{q, r\}$, while we have seen above that the PMA update of the empty valuation only outputs the first two of these valuations. Actually the above program defines an update operation with strength between WSS and PMA; it seems that it has not been considered in the literature up to now.

The exact DL-PA counterpart of the PMA updates is a bit more involved. The most straightforward way is to define π_A^{pma} to be the program

$$A? \cup \left(\neg A?; \bigcup_{\emptyset \subset P \subseteq \mathbb{P}_A} \mathsf{D}(A, P)?\ ;\ \mathsf{flip}^{\geq 0}(P)\ ;\ A?\right)$$

where $\mathsf{D}(A, P)$ is defined in Table 4. While it does the same job as $\diamond^{\mathsf{pma}}$, its length is exponential in the length of A due to the nondeterministic choice in the powerset of $\mathbb{P}_A$.

Our polynomial embedding of the PMA involves storing the values of variables as defined in Table 3.

Theorem 2. *Let A, B be propositional formulas. Let* $\mathbb{P}_A = \{p_1, \ldots, p_n\}$ *be set of variables occurring in A. Let* $\mathbb{P}'_A = \{p'_1, \ldots, p'_n\}$ *be such that the* p'_k *neither occur in A nor in B. Let* π_A^{pma} *be the following program:*

$$\begin{aligned}
&\mathsf{store}(\mathbb{P}_A); \mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?;\\
&[\mathsf{restore}^1(\mathbb{P}_A); \mathsf{restore}^{\geq 0}(\mathbb{P}_A)]\neg A?;\\
&\mathsf{flip}^{\geq 0}(\mathbb{P}'_A)
\end{aligned}$$

Then $B \diamond^{\mathsf{pma}} A = ||\langle(\pi_A^{\mathsf{pma}})^{-}\rangle B||$ *for every valuation v.*

The length of the program π_A^{pma} is linear in the length of A. It starts by saving the truth values of the variables of A by means of $\mathsf{store}(\mathbb{P}_A)$. It then goes to an A-valuation by $\mathsf{flip}^{\geq 0}(\mathbb{P}_A); A?$. The next step is to check whether all the ways one can get there are exclusively made up of $\neg A$-valuations: an A-valuation is disregarded if another A-valuation can be attained by fewer changes. Finally the copies p'_k of the variable p_k of A are forgotten by $\mathsf{flip}^{\geq 0}(\mathbb{P}'_A)$. For example,

$$\begin{aligned}
\pi_p^{\mathsf{pma}} &= p' {\leftarrow} p; (p {\leftarrow} \top \cup p {\leftarrow} \bot); p?;\\
&\quad [p \oplus p'?; p {\leftarrow} p'; (\mathsf{skip} \cup p {\leftarrow} p')]\neg p?; (p' {\leftarrow} \top \cup p' {\leftarrow} \bot)\\
&\equiv p' {\leftarrow} p; p {\leftarrow} \top; [p \oplus p'?; p {\leftarrow} p']\neg p?; (p' {\leftarrow} \top \cup p' {\leftarrow} \bot)\\
&\equiv p' {\leftarrow} p; p {\leftarrow} \top; [p \oplus p'?]\neg p'?; (p' {\leftarrow} \top \cup p' {\leftarrow} \bot)\\
&\equiv p' {\leftarrow} p; p {\leftarrow} \top; (p' {\leftarrow} \top \cup p' {\leftarrow} \bot)\\
&\equiv p {\leftarrow} \top; (p' {\leftarrow} \top \cup p' {\leftarrow} \bot)
\end{aligned}$$

As p' is fresh, the 'net effect' of the update program π_p^{pma} is just $p {\leftarrow} \top$, as expected. (Observe that while the programs π_p^{pma} and $p {\leftarrow} \top$ are not equivalent; what matters to us is that the formulas $\langle(\pi_p^{\mathsf{pma}})^{-}\rangle B$ and $\langle p {\leftarrow} \top^{-}\rangle B$ are so.)

Just as for the other update operators, the converse operator can be eliminated from π_A^{pma}. We do not state it because we were not able to simplify it further.

Forbus

Forbus's update operation [Forbus, 1989] is even more conservative than Winslett's. It is based on minimisation of the Hamming distance between valuations.

Semantics of Forbus's operation *Forbus's update of v by A* is defined as:

$$v \diamond^{\mathsf{forbus}} A = \{v_1 \in ||A|| \ : \text{there is no } v_2 \in ||A|| \text{ such that } \mathsf{card}(v \dot{-} v_2) < \mathsf{card}(v \dot{-} v_1)\}$$

So the set $v \diamond^{\mathsf{forbus}} A$ is the set of A-valuations that are closest to v w.r.t. the Hamming distance. For example,

$$\begin{aligned}
\emptyset \diamond^{\mathsf{forbus}} p \lor q &= \{\{p\}, \{q\}\}\\
\emptyset \diamond^{\mathsf{forbus}} (p \land q) \lor r &= \{\{r\}\}
\end{aligned}$$

The last example illustrates the difference with the PMA update operation.

Let A and B be propositional formulas. Just as WSS and PMA updates, *Forbus's update of B by A* is the pointwise update of the B-valuations by A:

$$B \diamond^{\mathsf{forbus}} A = \bigcup_{v \in ||B||} v \diamond^{\mathsf{forbus}} A$$

For example:

$$\begin{aligned}
\neg p \land \neg q \diamond^{\mathsf{forbus}} p \lor q &= ||p \oplus q||\\
\neg p \land \neg q \land \neg r \diamond^{\mathsf{forbus}} (p \land q) \lor r &= ||\neg p \land \neg q \land r||
\end{aligned}$$

Expressing Forbus's operation in DL-PA We polynomially transform update problems of the form $B \diamond^{\mathsf{forbus}} A$ into DL-PA: we define a family of update programs π_A^{forbus} whose length is cubic in the length of A.

Theorem 3. *Let A, B be propositional formulas. Let* π_A^{forbus} *be the following program:*

$$\left(\bigcup_{0 \leq m \leq \mathsf{card}(\mathbb{P}_A)} \mathsf{H}(A, \geq m)?; \mathsf{flip}^1(\mathbb{P}_A)^m\right); A?$$

Then $B \diamond^{\mathsf{forbus}} A = ||\langle(\pi_A^{\mathsf{forbus}})^{-}\rangle B||$.

The length of the program π_A^{forbus} is cubic in the length of A. It nondeterministically selects an integer m, checks whether the Hamming distance to A is at least m, flips m variables, and checks whether A is true. For example:

$$\begin{aligned}
\pi_p^{\mathsf{forbus}} &= \left((\mathsf{H}(p, \geq 0)?; \mathsf{flip}^1(p)^0) \cup (\mathsf{H}(p, \geq 1)?; \mathsf{flip}^1(p)^1)\right); p?\\
&\equiv \left((\top?; \mathsf{skip}) \cup (\neg\langle \mathsf{flip}^1(p)^{\leq 0}\rangle p?; p {\leftarrow} \neg p)\right); p?\\
&\equiv \left(\mathsf{skip} \cup (\neg p?; p {\leftarrow} \neg p)\right); p?\\
&\equiv p? \cup (\neg p?; p {\leftarrow} \neg p; p?)\\
&\equiv p {\leftarrow} \top
\end{aligned}$$

Therefore $B\diamond^{\mathsf{forbus}}p = ||\langle p{\leftarrow}\top^{-}\rangle B||$. Here is another example:

$$\begin{aligned}\pi^{\mathsf{forbus}}_{p\vee q} &\equiv (\mathsf{H}(p\vee q, \geq 0)?; \mathsf{flip}^1(p,q)^0; p\vee q?) \cup \\ &\quad (\mathsf{H}(p\vee q, \geq 1)?; \mathsf{flip}^1(p,q)^1; p\vee q?) \cup \\ &\quad (\mathsf{H}(p\vee q, \geq 2)?; \mathsf{flip}^1(p,q)^2; p\vee q?) \\ &\equiv (\top?; \mathsf{skip}; p\vee q?) \cup \\ &\quad (\neg(p\vee q)?; (p{\leftarrow}\neg p \cup q{\leftarrow}\neg q); p\vee q?) \cup \\ &\quad (\bot?; \mathsf{flip}^1(p,q)^2; p\vee q?) \\ &\equiv p\vee q? \cup (\neg(p\vee q)?; (p{\leftarrow}\neg p \cup q{\leftarrow}\neg q))\end{aligned}$$

The identity can be simplified by means of the program equivalences for the converse operator:

$$B\diamond^{\mathsf{forbus}}A = ||\langle A?; \Big(\bigcup_{0\leq m\leq \mathsf{card}(\mathbb{P}_A)} \mathsf{flip}^1(\mathbb{P}_A)^m; \mathsf{H}(A, \geq m)?\Big)\rangle B||.$$

Dalal

According to Katsuno and Mendelzon's distinction [Katsuno and Mendelzon, 1992], Dalal's belief change operation [Dalal, 1988] is not an update operation but rather a revision operation. We follow the usage in the literature and denote it by $*^{\mathsf{dalal}}$ (and not by $\diamond^{\mathsf{dalal}}$).

We are going to capture $*^{\mathsf{dalal}}$ by a DL-PA program $\pi^{\mathsf{dalal}}_{A,B}$ which depends not only on the input A, but also on the base B. One of the reasons for that is that Dalal's definition distinguishes two cases: when B is satisfiable and when it is unsatisfiable. In the latter case the result of the update by A is stipulated to be $||A||$. This guarantees that the revised base is satisfiable as soon as the input formula A is so. This however does not square well with our embedding into DL-PA: in modal logic, when B is unsatisfiable then $\langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle B$ is unsatisfiable, too. We therefore consider a variant where the revision of an unsatisfiable base is unsatisfiable, too. In the end of the section we embed Dalal's original operation.

Semantics of Dalal's operation Just as Forbus's update operation, Dalal's revision operation is based on minimisation of the Hamming distance between valuations. However, as Dalal's is a revision operator we minimise globally instead of valuation-wise.

Let A and B be propositional formulas. *Dalal's revision of B by A* is defined as:

$$\begin{aligned}B*^{\mathsf{dalal}}A = \{v_A \in ||A|| \ :\ & \text{there is } v_B \in ||B|| \text{ such that} \\ & \mathsf{card}(v_A \dot{-} v_B) \leq \mathsf{card}(v'_A \dot{-} v'_B) \\ & \text{for all } v'_A \in ||A||, v'_B \in ||B||\ \}\end{aligned}$$

It follows that when B is unsatisfiable then $B*^{\mathsf{dalal}}A$ is unsatisfiable. As mentioned above, our definition differs from Dalal's original definition in this point.

Here is an example:

$$\neg p\vee\neg q *^{\mathsf{dalal}} p \ = \ ||p\wedge\neg q||$$

It illustrates that revision operations satisfy the preservation postulate, which says that when $B\wedge A$ is satisfiable then $B * A = ||B\wedge A||$. This is the main difference with update operations: WSS, PMA, and Forbus's operation all agree that the update of $\neg p\vee\neg q$ by p should equal $||p||$.

Expressing Dalal's operation in DL-PA We polynomially transform revision problems of the form $B*^{\mathsf{dalal}}A$ into DL-PA: we define a family of revision programs $\pi^{\mathsf{dalal}}_{A,B}$ whose length is cubic in the length of A and B.

Theorem 4. *Let A, B be propositional formulas. Let $\pi^{\mathsf{dalal}}_{A,B}$ be the following program:*

$$\mathsf{flip}^{\geq 0}(\mathbb{P}_B)\ ;\ B?\ ;$$
$$\left(\bigcup_{0\leq m\leq \mathsf{card}(\mathbb{P}_A)} [\mathsf{flip}^{\geq 0}(\mathbb{P}_B)\ ;\ B?]\mathsf{H}(A, \geq m)?\ ; \mathsf{flip}^1(\mathbb{P}_A)^m\right); A?$$

Then $B*^{\mathsf{dalal}}A = ||\langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle\top||$.

The length of the program $\pi^{\mathsf{dalal}}_{A,B}$ is cubic in the sum of $|A| + |B|$. It visits all the B-valuations v_B via the program $\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?$, failing if there is no such valuation. It then nondeterministically selects an integer m such that the Hamming distance between the B-valuations and the A-valuations is at least m, flips m such variables, and checks whether A is true.

When the input is a literal then we obtain:

$$\begin{aligned}\pi^{\mathsf{dalal}}_{p,B} &= \mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?\ ; \\ &\quad \Big(([\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?]\mathsf{H}(p, \geq 0)?\ ; \mathsf{flip}^1(p)^0) \cup \\ &\quad\ ([\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?]\mathsf{H}(p, \geq 1)?\ ; \mathsf{flip}^1(p)^1)\Big); p? \\ &= \mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?\ ; \\ &\quad \Big([\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?]\top?; \mathsf{skip} \cup \\ &\quad\ ([\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?]\neg p?\ ; p{\leftarrow}\neg p)\Big); p? \\ &\equiv \mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?\ ;\ \Big(p?\ \cup ([\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?]\neg p?\ ; p{\leftarrow}\top)\Big)\end{aligned}$$

and likewise for $B*^{\mathsf{dalal}}\neg p$. So when $B\wedge p$ is consistent then $\pi^{\mathsf{dalal}}_{p,B}$ goes to a $B\wedge p$-valuation, and updates by p otherwise.

The proof of Theorem 4 is in two steps: first we prove that $B*^{\mathsf{dalal}}A = ||\langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle B||$, and then that $||\langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle B|| = ||\langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle\top||$. The second step relies on the validity of the equivalence $\langle B?; \mathsf{flip}^{\geq 0}(\mathbb{P}_B)\rangle\top \leftrightarrow \langle B?; \mathsf{flip}^{\geq 0}(\mathbb{P}_B)\rangle B$.

Finally, we note that we can also capture Dalal's original definition where the revision of an unsatisfiable B by A equals $||A||$ (instead of being unsatisfiable as with our definition). The modified identity which does the job is

$$B*^{\mathsf{dalal}}A = ||(\neg\mathsf{Sat}(B)\wedge A) \vee \langle(\pi^{\mathsf{dalal}}_{A,B})^{-}\rangle\top||$$

Discussion

Let us come back to the belief change consequence problem. This is a decision problem, and tight complexity bounds are known for each of the operations that we have considered: the belief change consequence problem is

- NP complete for the WSS update operation [Herzig and Rifi, 1999],
- $\mathrm{P}^{\mathrm{NP}[O(\log(n))]}$ for Dalal's belief revision operation, i.e., polynomial with $\log(n)$ calls to an NP oracle [Eiter and Gottlob, 1992], and

- Π_p^2 complete for Winslett's PMA update and Forbus's update [Eiter and Gottlob, 1992].

Our embeddings are in the star-free fragment of the language of DL-PA. We therefore obtain a PSPACE upper bound for the complexity of deciding whether $(B \circ A) \to C$ is the case. So this is clearly suboptimal. However, the nesting of modal operators in our decision problems $B \to [\pi_A]C$ is bounded: for the WSS, there is a single $\forall$ quantification and for the PMA, $[\pi_A]C$ is of the form

$$[\pi_1; \mathsf{flip}^{\geq 0}(\mathbb{P}_A); \pi_2; [\pi_3; \mathsf{restore}^{\geq 0}(\mathbb{P}_A)]\neg A?]C$$

where π_1, π_2, π_3 are deterministic programs, which is equivalent to

$$[\pi_1; \mathsf{flip}^{\geq 0}(\mathbb{P}_A); \pi_2](\langle \pi_3; \mathsf{restore}^{\geq 0}(\mathbb{P}_A)\rangle A \vee C)$$

The latter makes the $\forall$-$\exists$-quantification obvious. For Forbus's operator, $[\pi_A]C$ is basically of the form

$$\Big[\Big(\bigcup_{0 \leq m \leq \mathsf{card}(\mathbb{P}_A)} \neg\langle \mathsf{flip}^1(\mathbb{P}_A)^{\leq m-1}\rangle A?; \mathsf{flip}^1(\mathbb{P}_A)^m\Big); A\Big]C$$

which is again a $\forall$-$\exists$-quantification. The argument for Dalal's operation is similar. By Proposition 7, checking the validity of $B \to [\pi_A]C$ can be done by checking whether $v \in ||\mathsf{flip}^{\geq 0}(\mathbb{P}_A)(B \to [\pi_A]C)||$. The latter can be checked by an alternating algorithm, and as the alternation-depth is 2 for PMA and Forbus's update and for Dalal's revision, we obtain a Π_p^2 upper bound for the respective belief change decision problems.

We leave to future work the implementation of decision procedures for DL-PA and its fragments.

Up to now, complexity characterisations of belief change operations often resorted to translations from or to QBF (see e.g. [Eiter and Gottlob, 1992]). Our logic DL-PA is clearly at least as expressive as QBF: the QBF formula $\exists p\varphi$ is equivalent to the DL-PA formula $\langle p{\leftarrow}\top \cup p{\leftarrow}\bot\rangle\varphi$, which is exploited in [Herzig et al., 2011] to show PSPACE hardness of satisfiability and model checking. We believe that the richer language of programs of DL-PA provides a more natural tool to capture domains involving action and change. The present paper is part of a research program demonstrating the applicability of DL-PA to problems involving dynamics in finite domains. In other papers we have embedded several frameworks: Coalition Logic of Propositional Control and its extension by delegation programs [van der Hoek, Walther, and Wooldridge, 2010] in [Herzig et al., 2011], equilibrium logic underlying ASP and its extension by update operations in [Fariñas del Cerro, Herzig, and Su, 2013], a simple version of separation logic in [Herzig, 2013], belief merging operations in [Herzig, Pozos Parra, and Schwarzentruber, 2014], and Dung's argumentation frameworks and their update in [Doutre, Herzig, and Perrussel, 2014].

Conclusion

We have given polynomial embeddings of several prominent belief change operations into a single framework: Dynamic Logic of Propositional Assignments DL-PA. Within that framework, we have given syntactic counterparts to the most popular semantically defined belief change operations: Winslett's update operations WSS and PMA, Forbus's operation operation, and Dalal's revision operation. The reduction of the DL-PA formula representing the updated belief base can be exponentially longer than the original formula; however, our examples illustrate that sometimes our syntactic representation of the updated belief base can be substantially reduced.

As an aside, we have also defined a new update operation when discussing the program $\mathsf{while}\ \neg A\ \mathsf{do}\ \mathsf{flip}^1(\mathbb{P}_A)$ as a candidate for the update program implementing the PMA. The corresponding update operation is a bit more conservative than the WSS and less than the PMA. It seems that it was not studied in the literature up to now.

Our approach can be extended to cover other concrete belief change operations. For example, the program implementing Satoh's revision operation [Satoh, 1988] combines the search of a 'good starting point' as performed by the prefix $\mathsf{flip}^{\geq 0}(\mathbb{P}_B); B?$ of the revision program $\pi_{A,B}^{\mathsf{dalal}}$ with the copy-based change minimisation of the program π_A^{pma} for the PMA.

Acknowledgements

The work in this paper was supported by the French Agence Nationale de la Recherche (ANR) within the DynRes project.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *J. of Symbolic Logic* 50:510–530.

Baader, F.; Lutz, C.; Milicic, M.; Sattler, U.; and Wolter, F. 2005. A description logic based approach to reasoning about web services. In *Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005)*.

Balbiani, P.; Herzig, A.; and Troquard, N. 2013. Dynamic logic of propositional assignments: a well-behaved variant of PDL. In Kupferman, O., ed., *Logic in Computer Science (LICS), New Orleans, June 25-28, 2013*. IEEE.

Baral, C. 1995. Reasoning about actions: Non-deterministic effects, constraints, and qualification. In *IJCAI*, 2017–2026. Morgan Kaufmann.

Bienvenu, M.; Herzig, A.; and Qi, G. 2008. Prime implicate-based belief revision operators. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N., eds., *European Conference on Artificial Intelligence (ECAI)*, 741–742. IOS Press.

Chomicki, J., and Marcinkowski, J. 2004. On the computational complexity of minimal-change integrity maintenance in relational databases. In *Inconsistency Tolerance*, 119–150. Springer.

Dalal, M. 1988. Investigations into a theory of knowledge base revision: preliminary report. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 475–479.

de Giacomo, G. 1996. Eliminating "converse" from Converse PDL. *Journal of Logic, Language and Information (JoLLI)* 5:193–208.

Doutre, S.; Herzig, A.; and Perrussel, L. 2014. A dynamic logic framework for abstract argumentation. In and., ed., *International Conference, Principles of Knowledge Representation and Reasoning (KR), Vienna, Austria*.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.* 57(2-3):227–270.

Fariñas del Cerro, L.; Herzig, A.; and Su, E. I. 2013. Combining equilibrium logic and dynamic logic. In Cabalar, P., and Son, T. C., eds., *LPNMR*, volume 8148 of *Lecture Notes in Computer Science*, 304–316. Springer.

Forbus, K. D. 1989. Introducing actions into qualitative simulation. In Sridharan, N. S., ed., *Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, 1273–1278. Morgan Kaufmann Publishers.

Harel, D.; Kozen, D.; and Tiuryn, J. 2000. *Dynamic Logic*. MIT Press.

Harel, D. 1984. Dynamic logic. In Gabbay, D. M., and Günthner, F., eds., *Handbook of Philosophical Logic*, volume II. D. Reidel, Dordrecht. 497–604.

Herzig, A., and Rifi, O. 1999. Propositional belief base update and minimal change. *Artificial Intelligence Journal* 115(1):107–138.

Herzig, A.; Lorini, E.; Moisan, F.; and Troquard, N. 2011. A dynamic logic of normative systems. In Walsh, T., ed., *International Joint Conference on Artificial Intelligence (IJCAI)*, 228–233. Barcelona: IJCAI/AAAI. Erratum at `http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html`.

Herzig, A.; Lang, J.; and Marquis, P. 2013. Propositional update operators based on formula/literal dependence. *ACM Transactions on Computational Logic (TOCL)* 14(3):24.

Herzig, A.; Pozos Parra, P.; and Schwarzentruber, F. 2014. Belief merging in Dynamic Logic of Propositional Assignments. In Beierle, C., and Meghini, C., eds., *International Symposium on Foundations of Information and Knowledge Systems (FoIKS) (FolKS), Bordeaux*. Springer.

Herzig, A. 1996. The PMA revisited. In Aiello, L. C., and Shapiro, S., eds., *Proc. Int. Conf. on Knowledge Representation and Reasoning (KR'96)*, 40–50. Morgan Kaufmann Publishers.

Herzig, A. 2013. A simple separation logic. In Libkin, L.; Kohlenbach, U.; and de Queiroz, R. J. G. B., eds., *WoLLIC*, volume 8071 of *Lecture Notes in Computer Science*, 168–178. Springer.

Katsuno, H., and Mendelzon, A. O. 1992. On the difference between updating a knowledge base and revising it. In Gärdenfors, P., ed., *Belief revision*. Cambridge University Press. 183–203. (preliminary version in Allen, J.A., Fikes, R., and Sandewall, E., eds., Principles of Knowledge Representation and Reasoning: Proc. 2nd Int. Conf., pages 387–394. Morgan Kaufmann Publishers, 1991).

Kourousias, G., and Makinson, D. 2007. Parallel interpolation, splitting, and relevance in belief change. *Journal of Symbolic Logic* 72(3):994–1002.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: Formula-variable independence and forgetting. *J. Artif. Intell. Res. (JAIR)* 18:391–443.

Lang, J. 2007. Belief update revisited. In *Proc. of the 10th International Joint Conference on Artificiel Intelligence (IJCAI'07)*, 2517–2522.

Liu, H.; Lutz, C.; Milicic, M.; and Wolter, F. 2006. Updating description logic ABoxes. In Doherty, P.; Mylopoulos, J.; and Welty, C., eds., *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, 46–56. AAAI Press.

Liu, H.; Lutz, C.; Milicic, M.; and Wolter, F. 2011. Foundations of instance level updates in expressive description logics. *Artificial Intelligence* 175(18):2170–2197.

Marchi, J.; Bittencourt, G.; and Perrussel, L. 2010. Prime forms and minimal change in propositional belief bases. *Ann. Math. Artif. Intell.* 59(1):1–45.

Parikh, R. 1999. Beliefs, belief revision, and splitting languages. In Moss, L. S.; Ginzburg, J.; and de Rijke, M., eds., *Logic, Language and Computation, Vol. 2*. Stanford, CA, USA: Center for the Study of Language and Information. 266–278.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *FGCS*, 455–462.

Slota, M., and Leite, J. 2010. On semantic update operators for answer-set programs. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 957–962. IOS Press.

Tiomkin, M. L., and Makowsky, J. A. 1985. Propositional dynamic logic with local assignments. *Theor. Comput. Sci.* 36:71–87.

van der Hoek, W.; Walther, D.; and Wooldridge, M. 2010. On the logic of cooperation and the transfer of control. *J. of AI Research (JAIR)* 37:437–477.

van Eijck, J. 2000. Making things happen. *Studia Logica* 66(1):41–58.

Winslett, M.-A. 1988. Reasoning about action using a possible models approach. In *Proc. 7th Conf. on Artificial Intelligence (AAAI'88)*, 89–93.

Winslett, M.-A. 1990. *Updating Logical Databases*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Winslett, M.-A. 1995. Updating logical databases. In Gabbay, D. M.; Hogger, C. J.; and Robinson, J. A., eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4. Oxford University Press. 133–174.