Belief Merging in Dynamic Logic of Propositional Assignments

Andreas Herzig[1], Pilar Pozos-Parra[2], and François Schwarzentruber[3]

[1] Université de Toulouse, CNRS, IRIT, France
[2] Universidad Juárez Autónoma de Tabasco, Mexico
[3] ENS Rennes, IRISA, France

Abstract. We study syntactical merging operations that are defined semantically by means of the Hamming distance between valuations; more precisely, we investigate the Σ-semantics, Gmax-semantics and max-semantics. We work with a logical language containing merging operators as connectives, as opposed to the metalanguage operations of the literature. We capture these merging operators as programs of Dynamic Logic of Propositional Assignments DL-PA. This provides a syntactical characterisation of the three semantically defined merging operators, and a proof system for DL-PA therefore also provides a proof system for these merging operators. We explain how PSPACE membership of the model checking and satisfiability problem of star-free DL-PA can be extended to the variant of DL-PA where symbolic disjunctions that are parametrised by sets (that are not defined as abbreviations, but are proper connectives) are built into the language. As our merging operators can be polynomially embedded into this variant of DL-PA, we obtain that both the model checking and the satisfiability problem of a formula containing possibly nested merging operators is in PSPACE.

Keywords: belief merging, belief change, dynamic logic.

1 Introduction

To merge a vector of belief bases $E = \langle B_1, \cdots, B_n \rangle$ means to build a new belief base $\Delta(E)$. In the literature, E is called a *profile*, and $\Delta(E)$ is sometimes called the fusion of E. Much efforts were spent on the characterisation of 'good' merging operations Δ by means of rationality postulates [14–16]. Beyond such families of abstract belief merging operations satisfying the postulates, several *concrete* operations were also introduced and studied in the literature. Some are syntax-based and others are semantic. The former are also called 'formula-based', and the latter are called 'model-based' or 'distance-based'. An example of the former is the MCS operation [2], where each element B_i of E is viewed as a set of formulas that is not closed under logical consequence and where the construction of $\Delta(E)$ is based on the extraction of maximal consistent subsets of each B_i of E. Such operations are syntax dependent: they do not guarantee that the merging of logically equivalent profiles leads to merged bases that are logically equivalent.[1]

[1] Two profiles E and E' are logically equivalent if for every B_i in E there is a logically equivalent B'_j in E' and the other way round, for every B'_i in E' there is a logically equivalent B_j in E.

In contrast, syntax independence is guaranteed by the semantic merging operations, whose most prominent are Δ_{Σ}, $\Delta_{\max}$, and Δ_{Gmax} [19, 20]. These operations work on valuations of classical propositional logic. Indeed, even when the elements of the input profile are presented as formulas or sets thereof, the merging procedure starts by computing their models. The output set of valuations is sometimes transformed into a formula characterising the set, which can always be done because these operations are presented in terms of a finite set of propositional variables.

Contrasting with the existing literature, the present paper studies concrete semantic merging operations from a syntactic perspective: given a vector of formulas E, our aim is to obtain a syntactical representation of the merged belief base $\Delta(E)$, for Δ being Δ_{Σ}, $\Delta_{\max}$, or Δ_{Gmax}. As we have already said above, when the language is finite then it is easy to construct a formula representing $\Delta(E)$: it suffices to take the disjunction of the formulas describing the models of $\Delta(E)$, where each of these model descriptions is a conjunction of literals. Is there a better, more direct way of building a syntactic representation? In this paper we propose a powerful yet simple logical framework: Dynamic Logic of Propositional Assignments, abbreviated DL-PA [1]. DL-PA is a simple instantiation of Propositional Dynamic Logic PDL [7, 8]. Just as PDL, its language is built with two ingredients: atomic formulas and atomic programs. In both logics, atomic formulas are propositional variables. While PDL has abstract atomic programs, the atomic programs of DL-PA are assignments of propositional variables to either true or false, respectively noted $p{\leftarrow}\top$ and $p{\leftarrow}\bot$. The assignment $p{\leftarrow}\top$ corresponds to an update by p, while the assignment $p{\leftarrow}\bot$ corresponds to an update by $\neg p$. Complex programs π are built from atomic programs by the standard PDL program operators of sequential composition, nondeterministic composition, finite iteration (the so-called Kleene star), and test. Just as PDL, DL-PA has formulas of the form $\langle\pi\rangle\varphi$ and $[\pi]\varphi$, where π is a program and φ is a formula. The former expresses that φ is true after *some* possible execution of π, and the latter expresses that φ is true after *every* possible execution of π. For example, the DL-PA formula $\langle p{\leftarrow}\top \cup p{\leftarrow}\bot\rangle\varphi$ captures the propositional quantification $\exists p.\varphi$, illustrating that DL-PA naturally captures Quantified Boolean Formulas (QBF). It is shown in [1] that DL-PA formulas can be reduced to equivalent Boolean formulas. Just as for QBFs, the original formula is more compact than the equivalent Boolean formula. Star-free DL-PA has the same mathematical properties as the QBF reasoning problems; in particular, model checking, satisfiability and validity are all PSPACE complete. We believe DL-PA to be a more natural and flexible tool than QBF to reason about domains involving dynamics due to its more elaborate account in terms of programs.

Our main contributions are polynomial embeddings of semantic belief merging operators into DL-PA: to every profile E and merging operation Δ we associate a DL-PA formula $\varphi(\Delta, E)$, and we prove that the merged profile $\Delta(E)$ has the same models as $\varphi(\Delta, E)$. Then $\varphi(\Delta, E)$ may then be reduced to a Boolean formula, thus providing a syntactical representation of $\Delta(E)$ in propositional logic. A further contribution of our paper is a presentation of merging in terms of a recursive language with several merging operators Δ^{σ} in the object language, one operator per semantics σ. This contrasts with the usual presentations in terms of metalanguage operations (where we systematically use the term operat*or* for connectives in the object language, while we reserve the term operat*ion* for functions from the metalanguage).

The paper is organized as follows. In Section 2 we give the basic notation for propositional logic and recall the semantic definitions of the concrete merging operations Δ_Σ, Δ_{Gmax}, and $\Delta_{\max}$. In Section 3 we take a more syntactical stance: instead of viewing Δ as an operation in the metalanguage, we introduce a recursive language with families of n-ary merging operators in the object language and reformulate the above concrete merging operations in that language. In Section 4 we recall DL-PA. In Section 5 we embed the three merging operations into DL-PA. Section 6 concludes.

2 Background

We recall some standard notations and conventions for propositional logic, in particular distances between its valuations, as well as the definitions of the three concrete Boolean merging operators we are interested in.

2.1 Propositional Logic

Boolean formulas are built by means of the standard connectives $\neg$, $\vee$, etc. from a countable set of *propositional variables* $\mathbb{P} = \{p, q, \ldots\}$. We will in particular use the exclusive disjunction $\oplus$. We denote them by letters such as A, B, C; in particular, we use B, B_1, B_2, etc. for *Boolean belief bases*, which we identify with Boolean formulas.

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 noted $\mathbb{P}_A$. For example, $\mathbb{P}_{p\vee\neg q} = \{p, q\}$.

A *valuation* associates a truth value to each propositional variable. 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 it will be convenient to view v as a function from Boolean formulas into the set of truth values $\{0, 1\}$ and to write $v(p) = 1$ when $p \in v$ and $v(p) = 0$ when $p \notin v$.

Given a valuation v and a Boolean formula A, the truth value $v(A) \in \{0, 1\}$ is determined in the usual way. When $v(A) = 1$ then we say that v is an *A-valuation*. For example, $\{p, q\}$ is a $\neg p\vee\neg r$ valuation. The set of all A-valuations is denoted $||A||$. For example, $||p|| = \{v \in \mathbb{V} : p \in v\}$ and $||p \vee q|| = \{v \in \mathbb{V} : p \in v \text{ or } q \in v\} = ||p|| \cup ||q||$.

2.2 Distances

The *Hamming distance* between two valuations v_1 and v_2 is the cardinality of the symmetric difference between v_1 and v_2:

$$\begin{aligned} \mathsf{d_H}(v_1, v_2) &= \mathsf{card}((v_1 \setminus v_2) \cup (v_2 \setminus v_1)) \\ &= \mathsf{card}(\{p \in \mathbb{P} : v_1(p) \neq v_2(p)\}). \end{aligned}$$

So $\mathsf{d_H}(v_1, v_2)$ is the number of all those p such that $v_1(p) \neq v_2(p)$. For example, the Hamming distance between $\emptyset$ and $\{p, q\}$ is $\mathsf{card}(\emptyset \cup \{p, q\}) = 2$, and the Hamming distance between $\{p, q\}$ and $\{q, r, s\}$ is $\mathsf{card}(\{r, s\} \cup \{p\}) = \mathsf{card}(\{p, r, s\}) = 3$. Note that the Hamming distance might be infinite; for instance, $\mathsf{d_H}(\emptyset, \mathbb{P}) = \infty$.

The definition of Hamming distance can be extended to a distance between a valuation v and a set of valuations $V \subseteq \mathbb{V}$ as follows:

$$\mathsf{d_H}(v, V) = \begin{cases} 0 & \text{if } V = \emptyset \\ \min(\{\mathsf{d_H}(v, v') \ : \ v' \in V\}) & \text{otherwise} \end{cases}$$

This leads to the definition of the Hamming distance between a valuation and a Boolean formula as $\mathsf{d_H}(v, B) = \mathsf{d_H}(v, ||B||)$. For example:

$$\begin{aligned} \mathsf{d_H}(\{p, q\}, p\wedge\neg p) &= 0 \\ \mathsf{d_H}(\{p, q\}, p\wedge q) &= 0 \\ \mathsf{d_H}(\{p, q\}, \neg p\vee q) &= 0 \\ \mathsf{d_H}(\{p\}, \neg p\vee q) &= 0 \\ \mathsf{d_H}(\{p, q\}, \neg p\vee\neg q) &= 1 \\ \mathsf{d_H}(\{p, q\}, \neg p\wedge\neg q) &= 2 \\ \mathsf{d_H}(\{p, q\}, \neg p\vee\neg r) &= 0 \\ \mathsf{d_H}(\{p, q\}, (\neg p\vee\neg r)\wedge\neg q) &= 1 \end{aligned}$$

Lemma 1. *For every valuation v, $\mathsf{d_H}(v, B) \leq \mathsf{card}(\mathbb{P}_B)$.*

Proof. Let v be a valuation. If $||B|| = \emptyset$ then $\mathsf{d_H}(v, B) = \mathsf{d_H}(v, ||B||) = \mathsf{d_H}(v, \emptyset) = 0$ and the lemma is correct. Otherwise, let $v' \in ||B||$. Without loss of generality, we can assume that for all $p \notin \mathbb{P}_B$, $v(p) = v(p')$. Thus, $\mathsf{d_H}(v, v') \leq \mathsf{card}(\mathbb{P}_B)$. By definition of $\mathsf{d_H}(v, B)$ we have $\mathsf{d_H}(v, B) = \mathsf{d_H}(v, ||B||) \leq \mathsf{d_H}(v, v') \leq \mathsf{card}(\mathbb{P}_B)$.

Finally, the Hamming distance between a valuation v and a vector of Boolean belief bases $\langle B_1, \ldots, B_n\rangle$ is defined to be the vector of the distances:

$$\mathsf{d_H}(v, \langle B_1, \ldots, B_n\rangle) = \langle \mathsf{d_H}(v, B_1), \cdots, \mathsf{d_H}(v, B_n)\rangle$$

For example:

$$\begin{aligned} \mathsf{d_H}(\{p\}, \langle\neg p\vee\neg q\rangle) &= \langle 1\rangle \\ \mathsf{d_H}(\{p, q\}, \langle\neg p\vee\neg r, (\neg p\vee\neg r)\wedge\neg q\rangle) &= \langle 0, 1\rangle \\ \mathsf{d_H}(\{p, q\}, \langle\neg p\vee q, \neg p\vee\neg q, \neg p\wedge\neg q\rangle) &= \langle 0, 1, 2\rangle \end{aligned}$$

2.3 Various Merging Operations

A *profile*, typically noted E, is a vector of belief bases: $E = \langle B_1, \cdots, B_n\rangle$. The traditional definition of a belief merging operation is as a mapping Δ associating to every profile E a new belief base $\Delta(E)$. Such operations have been defined in several different ways and that is why we indicate a particular definition σ by a superscript and write $\Delta^\sigma(E)$. Throughout the present paper we suppose that there is no preference between the belief bases of a profile: we assume that $\Delta^\sigma(\varphi_1, \cdots, \varphi_n)$ is equivalent to $\Delta^\sigma(\varphi_{k_1}, \cdots, \varphi_{k_n})$, for every permutation $\langle\varphi_{k_1}, \cdots, \varphi_{k_n}\rangle$ of $\langle\varphi_1, \cdots, \varphi_n\rangle$. The reader may therefore view the vector as a set. We stick to the vector notation for two reasons: first, it is common in the merging literature, and second, it better fits the object language operators to be introduced in the next section.

Perhaps the best starting point is the merging operation that is based on minimisation of the sum of the Hamming distances to each belief base B_i of E, abbreviated Δ^{Σ}. It associates to every profile E the set of valuations such that the sum of the distances to the elements of E is minimal. Formally:

$$\Delta^{\Sigma}(E) = \Big\{v \in \mathbb{V} \ : \ \text{there is no } v' \in \mathbb{V} \text{ such that } \sum \mathsf{d_H}(v', E) < \sum \mathsf{d_H}(v, E)\Big\}.$$

For example:

$$\Delta^{\Sigma}(p, \neg p \vee q) = \{v \ : \ p, q \in v\} = ||p \wedge q||$$
$$\Delta^{\Sigma}(p \wedge q, \neg p \wedge \neg q) = 2^{\mathbb{P}} = ||\top||$$

Beyond Δ^{Σ} we consider other concrete merging operations: the Gmax merging operation Δ^{Gmax} and the max merging operator $\Delta^{\max}$. Their definitions are based on other minimisations. We do not give them here; instead, they will be presented in the next section in terms of object language operators.

Merging can also be done under integrity constraints. This leads to more general operations $\Delta^{\sigma}_{\psi}(E)$ where the formula ψ is an integrity constraint that the merged belief base should satisfy. The unconstrained $\Delta^{\sigma}(E)$ can then be identified with $\Delta^{\sigma}_{\top}(E)$. Then the Δ^{Σ} operation becomes:

$$\Delta^{\Sigma}_{\psi}(E) = \Big\{v \in ||\psi|| \ : \ \text{there is no } v' \in ||\psi|| \text{ such that } \sum \mathsf{d_H}(v', E) < \sum \mathsf{d_H}(v, E)\Big\}.$$

For example, $\Delta^{\Sigma}_{\neg r}(p, \neg p \vee q) = ||p \wedge q \wedge \neg r||$ and $\Delta^{\Sigma}_{p}(p \wedge q, \neg p \wedge \neg q) = ||p||$.

Observe that in the above definitions $\Delta_C(E)$ is a set of valuations. In contrast, the merging postulates to be given below are defined in terms of formulas: as already mentioned, papers on merging operations typically identify the set $\Delta_C(E)$ with the Boolean formula characterising it.

2.4 The Postulates for Merging with Integrity Constraints

We briefly recall the principles for merging operations that were introduced by Konieczny and Pino Pérez. We here present the version of [14], in a slightly adapted version because there, belief bases are considered to be finite sets of formulas (which are however often identified with their conjunction).

Let Δ be an mapping assigning to each belief profile E and integrity constraint C a belief base $\Delta_C(E)$. Δ is a merging operation if and only if it satisfies the following postulates.

(IC0) $\Delta_C(E) \to C$ is valid.
(IC1) If C is satisfiable then $\Delta_C(E)$ is satisfiable.
(IC2) If $C \wedge (\bigwedge E)$ is satisfiable then $\Delta_C(E) \leftrightarrow \bigwedge E$ is valid.
(IC3) For $E = \langle B_1, \cdots, B_n\rangle$ and $E' = \langle B'_1, \cdots, B'_n\rangle$, if $C \leftrightarrow C'$ and $B_i \leftrightarrow B'_i$ are valid for $1 \leq i \leq n$ then $\Delta_C(E) \leftrightarrow \Delta_{C'}(E')$ is valid.

(IC4) If $\Delta_C(\langle B, B'\rangle) \wedge B$ is satisfiable then $\Delta_C(\langle B, B'\rangle) \wedge B'$ is satisfiable.
(IC5) $\Delta_C(E) \wedge C' \rightarrow \Delta_{C\wedge C'}(E)$ is valid.
(IC6) If $\Delta_C(E) \wedge C'$ is satisfiable then $\Delta_{C\wedge C'}(E) \rightarrow \Delta_C(E)$ is valid.

In the above postulates, 'satisfiable' means 'propositionally satisfiable' and 'valid' means 'propositionally valid'.

The operations Δ^{Σ} and Δ^{Gmax} satisfy all the postulates, while the max merging operator $\Delta^{\max}$ does not. Nonetheless, many authors in the literature consider that the latter is an interesting merging operator.

3 A Modal Framework for Merging Operators

The Δ^{σ} are not logical connectives of the object language: they are part of the metalanguage. We highlight that by saying that they are operat*ions*. The merging operat*ors* to be introduced now are connectives of the object language, just as the Boolean operators $\neg$ and $\vee$ are.[2] For that reason we also write them differently as $\blacktriangle^{\sigma}$: for each semantics σ we have an object language operator $\blacktriangle^{\sigma}$.[3] It is an advantage of such a move that many things can then be proved in a formal, rigorous way inside a logical system, as opposed to lines of argument in natural language texts. Moreover, it also allows to take advantage of mathematical results such as complexity upper bounds and theorem proving methods for the logic.

If merging operators are in the object language, we have enough flexibility to nest merging operators and even talk about different semantics in the same formula, as illustrated by the well-formed formula $\blacktriangle^{\sigma_1}_{\blacktriangle^{\sigma_2}(p,q)}(p, p\vee q)$. To motivate this, consider a company whose productivity is declining and whose shareholders desire to implement a motivation policy in order to change the workers' conditions. They then have to merge the desires of every worker, while preserving several kinds of integrity constraints: job security, working environment, salary costs, job satisfaction. These different criteria have to be merged in their turn.

Formulas involving one or more kinds of merging operators may be given as an input to a reasoner. Observe that when we define the length of the input for the reasoner then one occurrence of a merging operator counts for 1 and certainly not for the length of the disjunction describing the corresponding set of valuations (as would be the case in the metalinguistic presentation).

3.1 Language

Our logical language $\mathcal{L}_{\blacktriangle}$ is defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\vee\varphi \mid \blacktriangle^{\sigma}_{\varphi}(\varphi, \cdots, \varphi)$$

[2] While the term 'merging operator' is customary in the literature, our terminology is in line with that of abstract algebra.

[3] More precisely, we do not have a single operator but a family of operators $\blacktriangle^{\sigma,n}(.)$ that is parametrized by the length n of the profile vector. We abstract away from this here.

where p ranges over the set of propositional variables $\mathbb{P}$ and where σ ranges over the set of symbols $\{\Sigma, \text{Gmax}, \max\}$. The informal reading of the formula $\blacktriangle^{\sigma}_{\psi}(E)$ is "the profile E has been merged (with merging semantics σ) under the constraint ψ".

Abusing language a bit, when the profile is $E = \langle \varphi_1, \dots, \varphi_n \rangle$ then instead of $\blacktriangle^{\sigma}_{\psi}(E)$ we write $\blacktriangle^{\sigma}_{\psi}(\varphi_1, \cdots, \varphi_n)$.

The function $\mathbb{P}_.$ associating to a formula the set of its propositional variables naturally extends to our language; in particular we have $\mathbb{P}_{\blacktriangle^{\sigma}_{\psi}(\varphi_1,\cdots,\varphi_n)} = \mathbb{P}_{\psi} \cup \big(\bigcup_{1 \le i \le n} \mathbb{P}_{\varphi_i} \big)$.

In the rest of the present section we introduce the truth conditions for the three merging operators $\blacktriangle^{\Sigma}$, $\blacktriangle^{\text{Gmax}}$, and $\blacktriangle^{\max}$. Clearly, when the profile $E = \langle B_1, \dots, B_n \rangle$ and the constraint C are Boolean then we expect the interpretation of the merging operator $\blacktriangle^{\sigma}$ under semantics σ to coincide with the merging operation Δ^{σ} defined in Section 2.3. In formulas, we expect the equality $\Delta^{\sigma}_{C}(E) = ||\blacktriangle^{\sigma}_{C}(E)||$ to hold for Boolean C and E.

3.2 The Σ-Semantics

The interpretation of $\blacktriangle^{\Sigma}$ is the set of valuations such that the sum of the distances to the elements of E is minimal. Formally:

$$||\blacktriangle^{\Sigma}_{\psi}(E)|| = \Big\{ v \in ||\psi|| \ : \ \text{there is no } v' \in ||\psi|| \text{ such that } \sum \mathsf{d}_{\mathsf{H}}(v', E) < \sum \mathsf{d}_{\mathsf{H}}(v, E) \Big\}.$$

The definition of the Hamming distance d_{H} is as in Section 2.2. The function $|| \cdot ||$ is the interpretation we are currently defining by induction over the formulas of $\mathcal{L}_{\blacktriangle}$. The integer $\sum \mathsf{d}_{\mathsf{H}}(v, E)$ is the sum of the elements of the vector $\mathsf{d}_{\mathsf{H}}(v, E)$.

For example, $||\blacktriangle^{\Sigma}_{\top}(p \land q, \neg p \land \neg q)|| = ||\top|| = 2^{\mathbb{P}}$.

3.3 The Gmax-Semantics

The interpretation of $\blacktriangle^{\text{Gmax}}$ is as follows:

$$||\blacktriangle^{\text{Gmax}}_{\psi}(E)|| = \Big\{ v \in ||\psi|| \ : \ \text{there is no } v' \in ||\psi|| \text{ such that } \mathsf{d}^{\text{sort}}_{\mathsf{H}}(v', E) <_{\text{lex}} \mathsf{d}^{\text{sort}}_{\mathsf{H}}(v, E) \Big\}$$

where $\mathsf{d}^{\text{sort}}_{\mathsf{H}}(v, E) = \text{sort}(d(v, \varphi_1), \dots, d(v, \varphi_n))$ is the list that is obtained from the vector $\langle d(v, \varphi_1), \dots, d(v, \varphi_n) \rangle$ by sorting it in descending order and where $<_{\text{lex}}$ is the lexicographical order between sequences of integers of the same length.

For example, $||\blacktriangle^{\text{Gmax}}_{\top}(p \land q, \neg p \land \neg q)|| = \{v \ : \ v(p) \neq v(q)\} = ||p \oplus q||$ because

$$\mathsf{d}^{\text{sort}}_{\mathsf{H}}(v, \langle p \land q, \neg p \land \neg q \rangle) = \begin{cases} \langle 2, 0 \rangle & \text{if } v(p) = v(q) \\ \langle 1, 1 \rangle & \text{otherwise.} \end{cases}$$

3.4 The max-Semantics

The interpretation of $\blacktriangle^{\max}$ is as follows:

$$||\blacktriangle^{\max}_{\psi}(E)||_{\max} = \Big\{ v \in ||\psi|| \ : \ \text{there is no } v' \in ||\psi|| \text{ such that } \max \mathsf{d}_{\mathsf{H}}(v', E) < \max \mathsf{d}_{\mathsf{H}}(v, E) \Big\}$$

where $\max \mathsf{d_H}(v, E)$ is the maximum of all the distances $\mathsf{d_H}(v, \varphi_i)$ between v and the elements φ_i of E.

For example, $||\blacktriangle^{\max}_{\top}(p\wedge q, \neg p\wedge\neg q)|| = \{v \ : \ v(p) \neq v(q)\} = ||p\oplus q||$ because for the valuations v such that $v(p) \neq v(q)$ we have that $\mathsf{d_H}(v, \langle p\wedge q, \neg p\wedge\neg q\rangle)$ equals $\langle 1, 1\rangle$ (and therefore the maximum of that vector is 1), while for the v such that $v(p) = v(q)$ the distance $\mathsf{d_H}(v, \langle p\wedge q, \neg p\wedge\neg q\rangle)$ is either $\langle 0, 2\rangle$ or $\langle 2, 0\rangle$ (and therefore the maximum is 2).

We recall that the max-semantics does not satisfy Konieczny and Pino Pérez's merging postulates. We also note that for the empty integrity constraint we have $||\blacktriangle^{\mathrm{Gmax}}_{\top}(E)|| \subseteq ||\blacktriangle^{\max}_{\top}(E)||$ for every profile E.

4 DL-PA: Dynamic Logic of Propositional Assignments

In this section we define syntax and semantics of dynamic logic of propositional assignments DL-PA and state complexity results. The star-free fragment of DL-PA was introduced in [9], where it was shown that it embeds Coalition Logic of Propositional Control [10–12]. The full logic with the Kleene star was further studied in [1].

4.1 Language

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

$$\pi ::= p{\leftarrow}\top \mid p{\leftarrow}\bot \mid \pi;\pi \mid \pi\cup\pi \mid \varphi? \mid \pi^*$$
$$\varphi ::= p \mid \top \mid \bot \mid \neg\varphi \mid \varphi\vee\varphi \mid \langle\pi\rangle\varphi$$

where p ranges over the set of propositional $\mathbb{P}$. 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 ("$\cup$"), unbounded iteration ("$(.)^*$", the so-called Kleene star), and test ("$(.)?$") are familiar from Propositional Dynamic Logic PDL.

The *length* of a formula φ, denoted $|\varphi|$, is the number of symbols used to write down φ, without "$\langle$", "$\rangle$", parentheses and commas. For example, $|q\wedge r| = |\neg(\neg q\vee\neg r)| = 6$ and $|\langle q{\leftarrow}\top\rangle(q\wedge r)| = 2+6 = 8$. The length of a program π, denoted $|\pi|$, is defined in the same way. For example, $|p{\leftarrow}\bot; p?| = 5$.

We abbreviate the logical connectives $\wedge$, $\rightarrow$, $\leftrightarrow$, 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 loop "while A do π" can be expressed as the DL-PA program $(A?;\pi)^*; \neg A?$. Third, for $n \geq 0$, the n-th iteration of π is defined inductively as:

$$\pi^0 = \mathsf{skip}$$
$$\pi^{n+1} = \pi^n; \pi$$

Let us now introduce the assignment of literals to variables by means of the following abbreviations that are proper to DL-PA:

$$p{\leftarrow}q = (q?; p{\leftarrow}\top) \cup (\neg q?; p{\leftarrow}\bot)$$
$$p{\leftarrow}\neg q = (q?; p{\leftarrow}\bot) \cup (\neg q?; p{\leftarrow}\top)$$

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. The length of $p{\leftarrow}q$ is $(2+1+3)+1+(3+1+3) = 14$. That of $p{\leftarrow}\neg q$ is 14, too.

The *star-free fragment* of DL-PA is the subset of the language made up of formulas without the Kleene star "$(.)^*$".

4.2 Semantics of DL-PA

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.

Table 1. Interpretation of the DL-PA connectives

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

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 $||\pi_1|| = ||\pi_2||$. In that case we write $\pi_1 \equiv \pi_2$. For example, the program equivalence $\pi;\mathsf{skip} \equiv \pi$ holds. 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 $\langle p{\leftarrow}\top\rangle\top$ 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 $\langle p{\leftarrow}\bot\rangle\neg p$.

In DL-PA, all the program operators can be eliminated: for every formula φ there is a formula equivalent φ' such that no program operator occurs in φ' [1, Theorem 1]. For example, $\langle p{\leftarrow}\top^*\rangle r$ is equivalent to $p \vee \langle p{\leftarrow}\top\rangle r$ and $\langle p{\leftarrow}\top; q{\leftarrow}\top\rangle r$ is equivalent to $\langle p{\leftarrow}\top\rangle\langle q{\leftarrow}\top\rangle r$. This contrasts with PDL, where this is not the case. Once all the program operators have been eliminated, modal operators only contain atomic programs. The latter are both serial and deterministic modal operators and therefore distribute over negation and disjunction. They can finally be eliminated when they face a propositional variable, according to the following equivalences:

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

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

All together, we have a complete set of reduction axioms: every formula reduces to a Boolean formula [1, Theorem 2].

Theorem 1. *For every* DL-PA *formula φ there is a Boolean formula φ' such that $\varphi \leftrightarrow \varphi'$ is* DL-PA *valid.*

For example, for different propositional variables r and p, the formula $\langle p{\leftarrow}q\rangle(p \vee r)$ is successively equivalent to $\langle p{\leftarrow}q\rangle p \vee \langle p{\leftarrow}q\rangle r$ and to $q \vee r$.

It is proved in [9] that both model and satisfiability checking are PSPACE complete for the star-free fragment of DL-PA.

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 we will use later.

Proposition 1. *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 $v \setminus P \in ||\varphi||$.*

In the rest of the paper we write $||\varphi||_{\mathsf{DL\text{-}PA}}$ in order to distinguish the interpretation of DL-PA formulas from the interpretation of the merging language.

4.3 Some Useful DL-PA Expressions

Table 2 collects some DL-PA expressions that are going to be convenient abbreviations.[4]

The program $\mathsf{vary}(P)$ nondeterministically changes the truth value of some of the variables in P. Its length is linear in the cardinality of P. So the program $\mathsf{vary}(\mathbb{P}_A); A?$ accesses all A-valuations that preserve the values of all those variables not occurring in A. Satisfiability of the Boolean formula A can be expressed in DL-PA by the formula $\langle\mathsf{vary}(\mathbb{P}_A); A?\rangle\top$ or the equivalent $\langle\mathsf{vary}(\mathbb{P}_A)\rangle A$. The program $\mathsf{flip}^1(P)$ changes the truth value of exactly one of the variables in P. The programs $\mathsf{flip}^{\leq m}(P)$ flip the truth value of at most m of the variables in P. The lengths of $\mathsf{flip}^m(P)$ and $\mathsf{flip}^{\leq m}(P)$ are quadratic in n. The formula $\mathsf{H}(\varphi, \geq d)$ is true in all those valuations whose Hamming distance to φ is d.

[4] 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.

Table 2. Some useful DL-PA expressions, for $P = \{p_1, \dots, p_n\}$, where $m \leq n$ in $\mathsf{flip}^m(P)$ and $\mathsf{flip}^{\leq m}(P)$, and where $d \leq \mathsf{card}(\mathbb{P}_\varphi)$ in $\mathsf{H}(\varphi, d)$

$$\mathsf{vary}(P) = (p_1{\leftarrow}\top \cup p_1{\leftarrow}\bot); \cdots ; (p_n{\leftarrow}\top \cup p_n{\leftarrow}\bot)$$

$$\mathsf{flip}^m(P) = \begin{cases} \mathsf{skip} & \text{if } m = 0 \\ (p_1{\leftarrow}\neg p_1 \cup \cdots \cup p_n{\leftarrow}\neg p_n); \mathsf{flip}^{m-1}(P) & \text{if } m \geq 1 \end{cases}$$

$$\mathsf{flip}^{\leq m}(P) = \begin{cases} \mathsf{skip} & \text{if } m = 0 \\ (\mathsf{skip} \cup \mathsf{flip}^1(P)); \mathsf{flip}^{\leq m-1}(P) & \text{if } m \geq 1 \end{cases}$$

$$\mathsf{H}(\varphi, d) = \begin{cases} \varphi & \text{if } m = 0 \\ \neg\langle \mathsf{flip}^{\leq d-1}(\mathbb{P}_\varphi)\rangle\varphi \wedge \langle \mathsf{flip}^d(\mathbb{P}_\varphi)\rangle\varphi & \text{if } m \geq 1 \end{cases}$$

For example:

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

Lemma 2. *The following hold:*

1. $\langle v_1, v_2\rangle \in ||\mathsf{vary}(P)||$ *iff* $(v_1 \setminus v_2) \cup (v_2 \setminus v_1) \subseteq P$.
2. $\langle v_1, v_2\rangle \in ||\mathsf{flip}^1(P)||$ *iff* $\langle v_1, v_2\rangle \in ||\mathsf{vary}(P)||$ *and* $\mathsf{card}(v_1 \dot{-} v_2) = 1$.
3. $\langle v_1, v_2\rangle \in ||\mathsf{flip}^{\leq m}(P)||$ *iff* $\langle v_1, v_2\rangle \in ||\mathsf{vary}(P)||$ *and* $\mathsf{card}(v_1 \dot{-} v_2) \leq m$.
4. $v \in ||\mathsf{H}(\varphi, d)||$ *iff* $\mathsf{d_H}(v, \varphi) = d$.

Note that $\mathsf{flip}^m(P)$ is nothing but the m-th iteration of $\mathsf{flip}^1(P)$, so one variable might be switched twice and therefore $\langle v_1, v_2\rangle \in ||\mathsf{flip}^m(P)||$ does not in general imply that the Hamming distance between v_1 and v_2 is m.

5 Embedding Merging Operators into DL-PA

In this section, we define a translation $tr(.)$ by induction over the formulas of our merging language $\mathcal{L}_\blacktriangle$. To every formula φ of our merging language $\mathcal{L}_\blacktriangle$ we associate a DL-PA formula $tr(\varphi)$. The Boolean part is translated as follows:

$$tr(p) = p$$
$$tr(\neg\varphi) = \neg tr(\varphi)$$
$$tr(\varphi \vee \psi) = tr(\varphi) \vee tr(\psi)$$

In the following three subsections we give the inductive cases of the definition of $tr(.)$ for $\blacktriangle^{\Sigma}$, $\blacktriangle^{\mathrm{Gmax}}$ and $\blacktriangle^{\max}$. We then prove that the translation is correct and that it gives us an algorithm to reason in $\mathcal{L}_{\blacktriangle}$ from an algorithm to reason in DL-PA. The reader may observe that our encodings are not particularly sophisticated and follow the semantic definitions in a fairly straightforward manner.

5.1 Embedding the Σ-Semantics

Let us define the translation for the $\blacktriangle^{\Sigma}$ as follows. Given a profile $E = \langle\varphi_1, \ldots, \varphi_n\rangle$, we define:

$$tr(\blacktriangle^{\Sigma}_{\psi}(E)) = tr(\psi) \wedge \bigvee_{\langle d_1,\ldots,d_n\rangle, d_k \leq \mathsf{card}(\mathbb{P}_{\varphi_k})} \Big(\Big(\bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d_i)\Big) \wedge \neg\langle\mathsf{vary}(\mathbb{P}_E)\rangle\Big(tr(\psi) \wedge \bigvee_{\langle d'_1,\ldots,d'_n\rangle, \sum_{k\leq n}(d'_k)<\sum_{k\leq n}(d_k)} \bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d'_i)\Big)\Big).$$

Intuitively, the translation does the following: first, the integrity constraint is required to be true (by $tr(\psi)$), second, it is checked that there is some vector $\langle d_1, \ldots, d_n\rangle$ of integers such that the Hamming distance from the present valuation to each $tr(\varphi_i)$-valuation is d_i (by $\mathsf{H}(tr(\varphi_i), d_i)$) and such that one cannot go to another valuation (by $\neg\langle\mathsf{vary}(\mathbb{P}_E)\rangle$) satisfying the constraint and whose sum of distances to the $tr(\varphi_i)$-valuations is smaller. As we are going to show, every model of the formula $tr(\blacktriangle^{\Sigma}_{\psi}(E))$ is indeed a model of the merged profile.

For example, $tr(\blacktriangle^{\Sigma}_{\top}(p, \neg p \vee q)))$ is

$$\begin{aligned}
\top \wedge \Big(&(\mathsf{H}(p,0) \wedge \mathsf{H}(\neg p\vee q, 0) \wedge \neg\langle\mathsf{vary}(\{p,q\})\rangle(\top \wedge \bot) \vee \\
&(\mathsf{H}(p,0) \wedge \mathsf{H}(\neg p\vee q, 1) \wedge \neg\langle\mathsf{vary}(\{p,q\})\rangle(\top \wedge \mathsf{H}(p,0) \wedge \mathsf{H}(\neg p\vee q, 0)) \vee \\
&(\mathsf{H}(p,1) \wedge \mathsf{H}(\neg p\vee q, 0) \wedge \neg\langle\mathsf{vary}(\{p,q\})\rangle(\top \wedge \mathsf{H}(p,0) \wedge \mathsf{H}(\neg p\vee q, 0)) \vee \\
&(\mathsf{H}(p,1) \wedge \mathsf{H}(\neg p\vee q, 1) \wedge \neg\langle\mathsf{vary}(\{p,q\})\rangle(\top \wedge ((\mathsf{H}(p,0) \wedge \mathsf{H}(\neg p\vee q, 0)) \vee \cdots))\Big)
\end{aligned}$$

which is equivalent to

$$\begin{aligned}
&(p \wedge (\neg p \vee q) \wedge \top) \vee \\
&(p \wedge (p \wedge \neg q) \wedge \bot) \vee \\
&(\neg p \wedge (\neg p \vee q) \wedge \bot) \vee \\
&(\neg p \wedge (p \wedge \neg q) \wedge \cdots),
\end{aligned}$$

i.e., to $p \wedge q$.

Here is another example:

$$\begin{aligned} tr(\blacktriangle^{\Sigma}_{\top}(p\wedge q, \neg p\wedge\neg q)) \leftrightarrow \top \wedge\ & (\mathsf{H}(p\wedge q, 0) \wedge \mathsf{H}(\neg p\wedge\neg q, 2)) \vee \\ & (\mathsf{H}(p\wedge q, 1) \wedge \mathsf{H}(\neg p\wedge\neg q, 1)) \vee \\ & (\mathsf{H}(p\wedge q, 2) \wedge \mathsf{H}(\neg p\wedge\neg q, 0)) \\ \leftrightarrow\ & (p\wedge q) \vee \\ & (\neg p\wedge q) \vee (p\wedge\neg q) \vee \\ & (\neg p\wedge\neg q) \\ \leftrightarrow\ & \top \end{aligned}$$

5.2 Embedding the Gmax-Semantics

The embedding of the Gmax-operator is in the same spirit as that of the previous operator. Given a profile $E = \langle\varphi_1, \ldots, \varphi_n\rangle$, we define:

$$\begin{aligned} tr(\blacktriangle^{\mathrm{Gmax}}_{\psi}(E)) = tr(\psi) \wedge \bigvee_{\langle d_1,\ldots,d_n\rangle,\ d_k\leq \mathsf{card}(\mathbb{P}_{\varphi_k})} & \Big(\big(\bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d_i)\big) \wedge \\ & \neg\langle \mathsf{vary}(\mathbb{P}_E)\rangle\Big(tr(\psi) \wedge \bigvee_{\langle d'_1,\ldots,d'_n\rangle, \mathrm{sort}(d'_1,\ldots,d'_n) <_{\mathrm{lex}} \mathrm{sort}(d_1,\ldots,d_n)} \ \bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d'_i)\Big)\Big). \end{aligned}$$

Intuitively, the translation checks the integrity constraints and checks for the vector characterising the Hamming distances to the φ_i-valuations that there exists no other valuation $tr(\psi)$ whose distance vector is smaller according to the sorted lexicographic ordering.

Table 3 contains an example. Another example is $tr(\blacktriangle^{\mathrm{Gmax}}_{\top}(p\wedge q, \neg p\wedge\neg q))$, which reduces to $p \leftrightarrow q$.

5.3 Embedding the max-Semantics

In a first try we have:

$$\begin{aligned} tr(\blacktriangle_{\psi}(E)) = tr(\psi) \wedge \bigvee_{\langle d_1,\ldots,d_n\rangle,\ d_k\leq \mathsf{card}(\mathbb{P}_{\varphi_k})} & \Big(\big(\bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d_i)\big) \wedge \\ & \neg\langle \mathsf{vary}(\mathbb{P}_E)\rangle(tr(\psi) \wedge \bigvee_{\langle d'_1,\ldots,d'_n\rangle, \max_{k\leq n}(d'_k) < \max_{k\leq n}(d_k)} \ \bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d'_i))\Big). \end{aligned}$$

This can actually be made more concise, and our official definition of the translation is as follows:

$$\begin{aligned} tr(\blacktriangle_{\psi}(E)) = tr(\psi) \wedge \bigvee_{d,\ d\leq \max_{k\leq n}(\mathsf{card}(\mathbb{P}_{\varphi_k}))} & \Big(\big(\bigwedge_{i\leq n} \langle \mathsf{flip}^{\leq d}(\mathbb{P}_{\varphi_i})\rangle tr(\varphi_i)\big) \wedge \\ & \neg\langle \mathsf{vary}(\mathbb{P}_E)\rangle\Big(tr(\psi) \wedge \bigwedge_{i\leq n} \langle \mathsf{flip}^{\leq d-1}(\mathbb{P}_{\varphi_i})\rangle tr(\varphi_i)\Big)\Big). \end{aligned}$$

Table 3. Example: translation of the Gmax merging of the profile $\langle p, p, \neg p\rangle$ under the empty integrity constraint $\top$

$$
\begin{aligned}
& tr(\blacktriangle^{\mathrm{Gmax}}_{\top}(p, p, \neg p)) \\
&\quad = \top \land (\mathsf{H}(p,0) \land \mathsf{H}(p,0) \land \mathsf{H}(\neg p,0) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\top \land \bot)) \lor \\
&\qquad (\mathsf{H}(p,0) \land \mathsf{H}(p,0) \land \mathsf{H}(\neg p,1) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\top \land \mathsf{H}(p,0) \land \mathsf{H}(p,0) \land \mathsf{H}(\neg p,0))) \lor \\
&\qquad (\mathsf{H}(p,0) \land \mathsf{H}(p,1) \land \mathsf{H}(\neg p,0) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \lor \\
&\qquad (\mathsf{H}(p,0) \land \mathsf{H}(p,1) \land \mathsf{H}(\neg p,1) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \lor \\
&\qquad (\mathsf{H}(p,1) \land \mathsf{H}(p,0) \land \mathsf{H}(\neg p,0) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \lor \\
&\qquad (\mathsf{H}(p,1) \land \mathsf{H}(p,0) \land \mathsf{H}(\neg p,1) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \lor \\
&\qquad (\mathsf{H}(p,1) \land \mathsf{H}(p,1) \land \mathsf{H}(\neg p,0) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \lor \\
&\qquad (\mathsf{H}(p,1) \land \mathsf{H}(p,1) \land \mathsf{H}(\neg p,1) \land \neg\langle \mathsf{vary}(\{p\})\rangle(\cdots)) \\
&\quad \leftrightarrow (p \land p \land \neg p \land \neg\bot) \lor \\
&\qquad (p \land p \land p \land \neg\bot) \lor \\
&\qquad (p \land \neg p \land \neg p \land \neg\bot) \lor \\
&\qquad (p \land \neg p \land p \land \neg\bot) \lor \\
&\qquad (\neg p \land p \land \neg p \land \neg\bot) \lor \\
&\qquad (\neg p \land p \land p \land \neg\bot) \lor \\
&\qquad (\neg p \land \neg p \land \neg p \land \neg\top) \lor \\
&\qquad (\neg p \land \neg p \land p \land \neg\bot) \\
&\quad \leftrightarrow p
\end{aligned}
$$

Intuitively, the integrity constrained is enforced and it is checked for some integer d that first, each φ_i in the profile has distance at most d and second, that there is no other valuation that both satisfies the integrity constraint and is strictly less than d away from each φ_i.

5.4 Correction of the Translations

Theorem 2. *Let φ be an $\mathcal{L}_{\blacktriangle}$ formula. Then $||\varphi|| = ||tr(\varphi)||_{\mathsf{DL\text{-}PA}}$.*

Proof. The proof is by induction on the form of φ. The only interesting case is that of merging operators. Let us consider the case of $\blacktriangle^{\Sigma}$. We prove in detail that $||\blacktriangle^{\Sigma}_{\psi}(E)|| = ||tr(\blacktriangle^{\Sigma}_{\psi}(E))||_{\mathsf{DL\text{-}PA}}$.

Let $v \in \mathbb{V}$ be a valuation. We have $v \in ||\blacktriangle^{\Sigma}_{\psi}(E)||$ iff $v \in ||\psi||$ and there is no other ψ-valuation v' such that $\sum \mathsf{d_H}(v', E) < \sum \mathsf{d_H}(v, E)$. The latter is the case iff $v \in ||\psi||$ and there are $\langle d_1, \ldots, d_n\rangle$ such that

1. $\mathsf{d_H}(v, \varphi_i) = d_i$ for every i, and
2. there is no ψ-valuation v' and vector $\langle d'_1, \cdots, d'_n\rangle$ such that $\mathsf{d_H}(v', \varphi_i) = d'_i$ for every i and $\sum d_i < \sum d'_i$.

By induction hypothesis, $v \in ||\psi||$ iff $v \in ||tr(\psi)||_{\mathsf{DL\text{-}PA}}$ and $v \in ||\varphi_i||$ iff $v \in ||tr(\varphi_i)||_{\mathsf{DL\text{-}PA}}$. Therefore $\mathsf{H}(\varphi_i, d_i)$ equals $\mathsf{H}(tr(\varphi_i), d_i)$.

We note that by Lemma 1 it is in order to only consider the d_i such that $d_i \leq \mathsf{card}(\mathbb{P}_{\varphi_i})$. By Lemma 2, Item 1 means that $v \in ||\mathsf{H}(tr(\varphi_i), d_i)||_{\mathsf{DL\text{-}PA}}$ for every i.

Item 2 means that the formula

$$\bigvee_{\langle d'_1,\cdots,d'_n\rangle, \sum_k(d'_k)<\sum_k(d_k)} \bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d'_i)$$

is unsatisfiable. According to Lemma 2 and Proposition 1, all the relevant valuations are accessed by the program $\mathsf{vary}(\mathbb{P}_E)$. Therefore Item 2 is equivalent to

$$v \in ||\neg\langle \mathsf{vary}(\mathbb{P}_E)\rangle(tr(\psi) \wedge \bigvee_{\langle d'_1,\cdots,d'_n\rangle, \sum_{k\leq n}(d'_k)<\sum_k(d_k)} \bigwedge_{i\leq n} \mathsf{H}(tr(\varphi_i), d'_i))||_{\mathsf{DL\text{-}PA}}.$$

Putting things together, items 1 and 2 are equivalent to $v \in ||tr(\blacktriangle^{\Sigma}_{\psi}(\varphi_1, \cdots, \varphi_n))||_{\mathsf{DL\text{-}PA}}$.

It follows from the above theorem that the merging of the Boolean profile $\langle B_1, \cdots, B_n\rangle$ under the Boolean constraint C equals $||tr(\blacktriangle^{\sigma}_{C}(B_1, \cdots, B_n))||_{\mathsf{DL\text{-}PA}}$.

The length of $tr(\varphi)$ is however exponential in the length of φ. Nevertheless, if we consider 'big disjunctions' such as $\bigvee_{\langle d_1,\ldots,d_n\rangle, d_k\leq \mathsf{card}(\mathbb{P}_{\varphi_k})}$, $\bigvee_{\langle d'_1,\ldots,d'_n\rangle, \sum_k(d'_k)<\sum_k(d_k)}$ etc. to be connectives of the object language—i.e., as symbolic disjunctions that are parametrised by sets and that are not defined as abbreviations, but are proper connectives—then the length of $tr(\varphi)$ is still polynomial in the length of φ. For instance, the length of

$$\bigvee_{\langle d'_1,\cdots,d'_n\rangle, \sum_k(d'_k)<\sum_k(d_k)} \bigwedge_{i\leq n} \mathsf{H}(\varphi_i^{\Sigma}, d'_i)$$

is $O(n)$ plus the length of $\mathsf{H}(\varphi_i^{\Sigma}, d'_i)$.

Corollary 1. *Both model checking and satisfiability checking of $\mathcal{L}_{\blacktriangle}$-formulas is in PSPACE.*

Proof. First we give the argument why both model and satisfiability checking are PSPACE-complete for the star-free fragment of DL-PA if we allow symbolic disjunctions in DL-PA formulas. We do so by adapting the proof of PSPACE membership of [9]: in order to check whether $\bigvee_{\langle d'_1,\ldots,d'_n\rangle, \sum_k(d'_k)<\sum_k(d_k)} \psi$ is true at a valuation v we backtrack and test all the choices $\langle d'_1, \ldots, d'_n\rangle$ such that $\sum_k(d'_k) < \sum_k(d_k)$. This backtrack process can be implemented as an algorithm that only uses a polynomial amount of memory. By Theorem 2 we then reduce polynomially model (satisfiability) checking of $\mathcal{L}_{\Delta}$ formulas to model (satisfiability) checking of a DL-PA-formulas, where 'big disjunctions' are viewed as being symbolic.

Note that the language of DL-PA is more succinct than that of Boolean formulas: although every formula of DL-PA is equivalent to a Boolean formula, equivalent Boolean formulas can be exponentially bigger. So SAT techniques for propositional logic do not provide interesting decision procedures for $\mathcal{L}_{\blacktriangle}$.

6 Conclusion

We have defined a single language $\mathcal{L}_{\blacktriangle}$ in which all merging operators are in the object language: they are considered to be modal operators and can be nested. This differs with other approaches such as [18] and [5]. As far as we know, the only similar approach is [17], where the merging operator (as well as the comma separating the elements of profiles) are considered to be in the object language.

We have then embedded this language into Dynamic Logic of Propositional Assignments, DL-PA. This has enabled us to give syntactic counterparts to the most popular semantically defined merging operations. Using the reduction principles of DL-PA we can therefore rewrite formulas to Boolean formulas. As our examples show, such formulas may be quite long; in particular, they typically contain a lot of disjunctions. They can however often be simplified by means of standard syntactical operations. This provides interesting syntactical representations of merged belief bases.

The logic DL-PA actually provides a sort of assembler language for merging operators. Its use avoids the design of specific tools implementing merging operators. Unfortunately, no efficient reasoning mechanisms for DL-PA exist up to now, and it would be interesting to have such tools. (It could also be based on Binary Decision Diagrams as in [5].) As we have seen, if we want the embeddings to be polynomial then such tools should be able to handle 'big disjunctions' and 'big conjunctions'.

The star-free fragment of DL-PA into which we have mapped various merging operators has PSPACE complexity (both model checking and satisfiability). This induces a result for our merging language $\mathcal{L}_{\blacktriangle}$, which is new because $\mathcal{L}_{\blacktriangle}$ authorizes arbitrary nesting of merging operators. It is possible that the translated formulas however have patterns that are less complex.

As to future work, a first perspective is to study the mathematical properties of merging operators in more detail. One example is the behaviour of iterated merging operators (which is a research project similar to that for iterated belief revision, see e.g. [3].) Reasoning should be considerably facilitated by the help of a DL-PA reasoner. For instance, suppose we want to know whether the operator $\blacktriangle_{\top}^{\max}$ is associative. We may run the following experimental protocol: first, choose some Boolean formulas A, B, C and write down the formula $\blacktriangle_{\top}^{\max}(A, \blacktriangle_{\top}^{\max}(B, C)) \leftrightarrow \blacktriangle_{\top}^{\max}(\blacktriangle_{\top}^{\max}(A, B), C)$; second, translate this formula into DL-PA; third, run a DL-PA reasoner. Note however that one cannot use the theorem proving procedure for DL-PA because it only works for formula instances and not for formula schemas. (This is related to the fact that the rule of uniform substitution does not preserve validity in DL-PA, which generally fails in dynamic logics).

Our embeddings are somewhat simpler than the embeddings of belief change operations into QBF as done in [4] since DL-PA is a logic of programs. The same argument applies to embeddings of merging problems into MSO. Our approach may also be useful to capture semantics of merging: one may think in particular of new semantics requiring loops, which can be directly captured in DL-PA by the Kleene-star operator, whereas the encoding as a QBF will most probably be trickier.

A second perspective is to focus on embeddings of other existing operations. We did not succeed yet in embedding other approaches to merging such as [13] and syntax-based operations such as MCS of [2]. Note that in principle this might however be feasible: while the Hamming distance is a semantical notion, the function $\mathbb{P}_\varphi$ is purely syntactic.

There exist also tentatives to define merging operations in first order logic [6]. In the long run, we may plan to extend DL-PA with first order constructions in order to capture those merging operations.

Acknowledgements. We wish to thank the three FOIKS reviewers for their critical comments and pointers to relevant work that we had not considered at the time of the submission.

References

1. Balbiani, P., Herzig, A., Troquard, N.: 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, IEEE (2013), `http://www.ieee.org/`
2. Baral, C., Kraus, S., Minker, J., Subrahmanian, V.S.: Combining knowledge bases consisting of first-order theories. Computational Intelligence 8, 45–71 (1992)
3. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. Artificial Intelligence 89(1), 1–29 (1997)
4. Delgrande, J.P., Schaub, T., Tompits, H., Woltran, S.: On computing belief change operations using quantified boolean formulas. Journal of Logic and Computation 14(6), 801–826 (2004)
5. Gorogiannis, N., Hunter, A.: Implementing semantic merging operators using binary decision diagrams. International Journal of Approximate Reasoning 49(1), 234–251 (2008)
6. Gorogiannis, N., Hunter, A.: Merging first-order knowledge using dilation operators. In: Hartmann, S., Kern-Isberner, G. (eds.) FoIKS 2008. LNCS, vol. 4932, pp. 132–150. Springer, Heidelberg (2008)
7. Harel, D.: Dynamic logic. In: Gabbay, D.M., Günthner, F. (eds.) Handbook of Philosophical Logic, vol. II, pp. 497–604. D. Reidel, Dordrecht (1984)
8. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
9. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems. In: Walsh, T. (ed.) International Joint Conference on Artificial Intelligence (IJCAI), IJCAI/AAAI, Barcelona, pp. 228–233 (2011), Erratum at `http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html`
10. van der Hoek, W., Walther, D., Wooldridge, M.: On the logic of cooperation and the transfer of control. J. of AI Research (JAIR) 37, 437–477 (2010)
11. van der Hoek, W., Wooldridge, M.: On the dynamics of delegation, cooperation and control: A logical account. In: Proc. AAMAS 2005 (2005)
12. van der Hoek, W., Wooldridge, M.: On the logic of cooperation and propositional control. Artif. Intell. 164(1-2), 81–119 (2005)
13. Konieczny, S.: On the difference between merging knowledge bases and combining them. In: KR, pp. 135–144 (2000)
14. Konieczny, S., Pérez, R.P.: Logic based merging. Journal of Philosophical Logic 40(2), 239–270 (2011)
15. Konieczny, S., Pérez, R.P.: On the logic of merging. In: Proc. 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 1998), pp. 488–498. Morgan Kaufmann (1998)
16. Konieczny, S., Pino Pérez, R.: Merging with integrity constraints. In: Hunter, A., Parsons, S. (eds.) ECSQARU 1999. LNCS (LNAI), vol. 1638, pp. 233–244. Springer, Heidelberg (1999)

17. Lang, J., Marquis, P.: Reasoning under inconsistency: A forgetting-based approach. Artificial Intelligence 174(12), 799–823 (2010)
18. Liberatore, P., Schaerf, M.: Brels: A system for the integration of knowledge bases. In: KR, pp. 145–152. Citeseer (2000)
19. Lin, J., Mendelzon, A.: Knowledge base merging by majority. In: Pareschi, R., Fronhoefer, B. (eds.) Dynamic Worlds: From the Frame Problem to Knowledge Management, Kluwer Academic (1999)
20. Revesz, P.Z.: On the semantics of theory change: arbitration between old and new information. In: Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS 1993, pp. 71–82. ACM, New York (1993)