Abstract Argumentation in Dynamic Logic: Representation, Reasoning and Change

Sylvie Doutre, Andreas Herzig and Laurent Perrussel

Abstract We provide a logical analysis of Dung's abstract argumentation frameworks and their dynamics. We express attack relation and argument status by means of propositional variables and define acceptability criteria by formulas of propositional logic, which enables us to formulate the standard reasoning problems in logic. While the approaches in the literature express these problems as boolean or quantified boolean formulas, we here take advantage of a variant of Propositional Dynamic Logic PDL: Dynamic Logic of Propositional Assignments DL-PA, whose atomic programs are assignments of propositional variables to truth values. One of the benefits is that algorithms computing extensions of argumentation frameworks can be viewed as particular DL-PA programs. This allows us to formally prove the correctness of these algorithms. Another benefit is that in the same logic we can also design and study programs which modify argumentation frameworks. Indeed, the basic operations on these propositional variables, viz. change of the truth values of the attack variables and the argument status variables, are nothing but atomic programs of DL-PA. We mainly focus on how the acceptance of one or more arguments can be enforced and show how this can be achieved by changing the truth values of the propositional variables describing the attack relation in a minimal way.

1 Introduction

Argumentation is a reasoning model based on the construction and on the evaluation of arguments. The seminal approach by Dung [31] defines an argumentation

Sylvie Doutre
IRIT, Université Toulouse 1 Capitole, France, e-mail: sylvie.doutre@irit.fr

Andreas Herzig
IRIT, CNRS, France, e-mail: andreas.herzig@irit.fr

Laurent Perrussel
IRIT, Université Toulouse 1 Capitole, France, e-mail: laurent.perrussel@irit.fr

framework (henceforth abbreviated AF) as a set of arguments along with an attack relation between them. The arguments are abstract: their structure and origin are left unspecified. Dung and his followers defined various semantics for the evaluation of the acceptability of arguments. We here focus on semantics in terms of extensions: subsets of the set of arguments that are collectively acceptable. Several such semantics were proposed and discussed in the literature. We refer to [5, 6] for a comprehensive overview of the principles underlying extension-based AF semantics as well as other AF semantics.

Extension-based semantics were analysed in several logical frameworks, including propositional logic [12], Answer-Set Programming [35], Alternating-time Temporal Logic [11], and quantified Boolean formulas QBF [2, 27]. Their encodings of AFs in logic typically make use of two kinds of propositional variables: attack variables such as $\mathsf{r}_{a,b}$ express that argument a attacks b and acceptance variables such as in_a express that a is accepted. The various semantics can then be characterised in that language by boolean formulas built from attack and acceptance variables, constraining the valuations to correspond to the extensions under the semantics: each model of the formula stands for an extension of the AF.

In the present paper we advocate another logical framework: Dynamic Logic of Propositional Assignments, abbreviated DL-PA [4]. DL-PA is a simple instantiation of Propositional Dynamic Logic PDL [36, 37] whose atomic programs are assignments of propositional variables to either true or false. Complex programs are built as usual from atomic programs by the standard PDL program operators of sequential composition, nondeterministic composition, converse, and test. It is shown in [4] that every DL-PA formula can be reduced to an equivalent propositional formula. The aim of our paper is to show that DL-PA provides an interesting tool to reason about AFs and their modification.

There are two immediate benefits of the DL-PA framework. The first that a given semantics σ can be characterized by means of a DL-PA formula. While this is a priori not surprising—given that DL-PA and propositional logic have the same expressive power—, it is of interest because DL-PA formulas are sometimes more compact. This applies in particular to the preferred semantics: it is one level higher in the polynomial hierarchy than the other semantics and can therefore not be captured by a polynomial propositional logic formula; we provide a polynomial DL-PA formula.

The second benefit is that, being a dynamic logic, DL-PA can account for the dynamics that is at work in the construction of extensions. To start with, we can define a general family of non-deterministic DL-PA programs building all extensions of a given AF. These programs are parametrized by a given semantics σ and follow a generate-and-test schema: they generate all logically possible extensions and then test the (propositional or DL-PA) formula characterizing the semantics. Such programs exist for every semantics that can be characterised by a DL-PA formula. When executed in a valuation describing an AF they build all the extensions of that AF. The above generic algorithm can be improved by recasting specific more efficient algorithms into DL-PA programs, enabling formal verification of correctness.

The third and main benefit of DL-PA is that it provides a suitable framework to account for the dynamics of AFs. Indeed, authors from several places recently

investigated that issue, including [8, 7, 9, 10] from Leipzig, [15, 14, 16, 13, 47] from Toulouse, [22, 20, 23, 21, 44, 25] from Lens, [17] from Luxembourg, [32] from London, [26] from Vienna, and [45, 48] from Helsinki. They are all based on a representation of AFs in propositional logic. They start by distinguishing several kinds of AF modifications, such as the addition or the removal of attacks, or the enforcement of the acceptability of a given argument a, as well as several kinds of success criteria, e.g. such that a is part of at least one extension, or of all extensions. All these papers build on previous work in belief change, either referring to AGM theory [1], such as [17, 22], or to KM theory [41], such as [14]. They express the modification as a logical formula describing some *goal*, i.e., a property that the AF should satisfy: the task is to revise/update the AF so that this formula is true. An overview of the contributions on the dynamics of AFs can be found in [30].

The above papers do not provide a single framework encompassing at the same time an AF, the logical definition of the enforcement constraint and the change operations: there is usually one language for representing AFs and another language for representing constraints, plus some definitions in the metalanguage connecting them. As we are going to show, DL-PA provides a general, unified logical framework for both the representation and the update of AFs. We start by showing that the construction of extensions under a given semantics can be performed by a DL-PA program that is parametrized by the formula describing the semantics. Then we consider modifications of the attack relation and/or of the extensions. Modifications of the extensions are enforced by changing the attack relation only (addition or removal of attacks between the existing arguments). This can be achieved by changing the truth values of the attack variables. More precisely, to every input formula φ describing the desired modification we associate a DL-PA program π_φ implementing the update by φ. We can then check whether a formula ψ is true in all (resp. in some) extensions of (the AF resulting from) the update of the AF by the goal φ.

The paper is organized as follows. In the next section we introduce DL-PA. In Section 3 we recall the definitions of an AF and of various semantics as well as their encoding in propositional logic and in DL-PA. In Section 4 we show how extensions can be built by means of DL-PA programs. In Section 5 we discuss several possible kinds of modifications of an AF. We then focus in Section 6 on the most challenging kind of operation: modifications enforcing a goal either in all extensions or in some extension. We apply it to the modification of the attack relation and of the extensions (Section 7). After that, we discuss several ways to extend our framework in order to capture other kinds of modifications (Section 8). The last section concludes.

The present paper extends and generalises previous work that was presented at KR'2014 [28] and complemented in [29]. We have in particular elaborated how the semantics can be compactly described by means of DL-PA formulas and how it can be computed by means of DL-PA programs.

2 Dynamic Logic of Propositional Assignments

We now introduce our logical framework: Dynamic Logic of Propositional Assignments DL-PA [39, 4, 3, 38]. DL-PA is an instance of Propositional Dynamic Logic PDL [36, 37]. It has the same program operators: sequential and nondeterministic composition, converse, iteration, and test. However, the atomic programs of DL-PA are not abstract as in PDL, but concrete assignments of truth values to propositional variables: the assignment of true to p is written $p{\leftarrow}\top$ and the assignment of false to p is written $p{\leftarrow}\bot$. The formulas of DL-PA express what holds after the execution of a program.

We here base our work on the star-free version of DL-PA of [39] that we extend by the operator of converse execution of programs as done in [38]. We keep on calling that logic DL-PA.

In the rest of the section we define syntax and semantics of DL-PA, state its relevant properties and introduce some useful programs.

2.1 Language

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

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

where p ranges over the set of propositional variables $\mathbb{P}$.

The key formula $\langle\pi\rangle\varphi$ reads "φ holds after some execution of π". The atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ respectively make p true and make p false. The operators of sequential composition (";"), nondeterministic composition ("$\sqcup$") and test ("(.)?") are familiar from PDL. The operator "$(.)^{-}$" is the operator of converse execution. So $\langle\pi^{-}\rangle\varphi$ can be read "φ was true before some execution of π".

An *expression* is a formula or a program. We use ϵ to denote expressions.

Let $P \subseteq \mathbb{P}$ be a set propositional variables. The set of DL-PA formulas that can be built from P is noted $\mathsf{Lang}(P)$. Given an expression ϵ, the set of variables from P occurring in ϵ is noted $P(\epsilon)$. So $\mathbb{P}(\varphi)$ is the set of propositional variables from $\mathbb{P}$ occurring in the formula φ. For example, $\mathbb{P}(p{\leftarrow}\top \sqcup q{\leftarrow}\bot) = \{p, q\} = \mathbb{P}(\langle p{\leftarrow}\bot\rangle q)$.

The *length* of an expression ϵ, noted $|\epsilon|$, is the number of symbols used to write down ϵ, without "$\langle$", "$\rangle$", and parentheses. For example, $|\neg(q \vee \neg r)| = 1{+}(1{+}1{+}2) = 5$, $|\langle q{\leftarrow}\top\rangle(q \vee r)| = 3{+}3 = 6$, and $|p{\leftarrow}\bot; p?| = 3{+}1{+}2 = 6$.

Conjunction ($\wedge$), implication ($\rightarrow$) and equivalence ($\leftrightarrow$) are considered with their usual meaning. We also make use of the exclusive disjunction $\varphi \oplus \psi$ which abbreviates $(\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$. The formula $[\pi]\varphi$ abbreviates $\neg\langle\pi\rangle\neg\varphi$. So $[\pi]\varphi$ has to be read "φ holds after every execution of π".

2.2 Semantics of DL-PA

Models of DL-PA formulas are nothing but models of classical propositional logic, i.e., subsets of the set of propositional variables $\mathbb{P}$, alias valuations. We use $v, v', \ldots$ for valuations. We sometimes write $v(p) = 1$ when $p \in v$ and $v(p) = 0$ when $p \notin v$.

DL-PA formulas φ are interpreted as sets of valuations $||\varphi|| \subseteq 2^{\mathbb{P}}$. When $v \in ||\varphi||$ we say that v is a model of φ, or a φ-model. DL-PA programs π are interpreted as *relations between valuations* $||\pi|| \subseteq 2^{\mathbb{P}\times\mathbb{P}}$. The definitions of $||\varphi||$ and $||\pi||$ are by mutual recursion just as in PDL, where the atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ are interpreted as update operations on valuations. Table 1 gives the interpretation of the DL-PA connectives. For instance, suppose $\mathbb{P}$ is the singleton $\{p\}$. Consider the atomic program $\pi = p{\leftarrow}\top$ and the two valuations $v_1 = \emptyset$ and $v_2 = \{p\}$. The execution of π relates v_1 to v_2, and v_2 to itself. So π is interpreted as $||\pi|| = \{(v_1, v_2), (v_2, v_2)\}$.

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. When we say that two expressions are equivalent we mean program equivalence if we are talking about programs, and formula equivalence otherwise.

A formula φ is *valid* if it is equivalent to $\top$, i.e., if $||\varphi|| = 2^{\mathbb{P}}$. It is *satisfiable* if it is not a 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 both valid.

$$
\begin{aligned}
||p{\leftarrow}\top|| &= \{(v_1, v_2) \;:\; v_2 = v_1 \cup \{p\}\} \\
||p{\leftarrow}\bot|| &= \{(v_1, v_2) \;:\; v_2 = v_1 \setminus \{p\}\} \\
||\pi; \pi'|| &= ||\pi|| \circ ||\pi'|| \\
||\pi \sqcup \pi'|| &= ||\pi|| \cup ||\pi'|| \\
||\pi^-|| &= ||\pi||^{-1} \\
||\varphi?|| &= \{(v, v) \;:\; v \in ||\varphi||\} \\
||p|| &= \{v \;:\; p \in v\} \\
||\top|| &= 2^{\mathbb{P}} \\
||\bot|| &= \emptyset \\
||\neg\varphi|| &= 2^{\mathbb{P}} \setminus ||\varphi|| \\
||\varphi \vee \psi|| &= ||\varphi|| \cup ||\psi|| \\
||\langle\pi\rangle\varphi|| &= \{v \;:\; \text{there is } v' \text{ such that } (v, v') \in ||\pi|| \text{ and } v' \in ||\varphi||\}
\end{aligned}
$$

Table 1 Interpretation of the DL-PA connectives

2.3 Properties of DL-PA

As in any reasonable logic, replacement of equivalents is valid in DL-PA: equivalence is preserved under replacement of one or more sub-expressions by an equivalent expression [4].[1]

The fact that DL-PA formulas are interpreted through classical propositional logic valuations indicates that modal operators can be eliminated. Indeed, every DL-PA formula is equivalent to a boolean formula.

Theorem 1 ([4, 38]) *For every* DL-PA *formula there is an equivalent boolean formula.*

The proof uses valid equivalences that together make up a complete set of reduction axioms. For example, $\langle p{\leftarrow}\bot\rangle q$ is equivalent to $\top$ if p and q are identical, and is equivalent to q otherwise. For a more complex example, consider two different propositional variables r and p and the formula $\langle p{\leftarrow}\bot\rangle(p \vee r)$. It is successively equivalent to $\langle p{\leftarrow}\bot\rangle p \vee \langle p{\leftarrow}\bot\rangle r$, to $\bot \vee r$, and to r. Note that the boolean formula resulting from the reduction might be exponentially longer than the original formula.

The satisfiability problem is to decide whether $||\varphi|| \neq \emptyset$ for a given formula φ; the model checking problem is to decide whether $v \in ||\varphi||$ for a given formula φ and valuation v.

Theorem 2 ([3]) *The satisfiability problem and the model checking problem for* DL-PA *are both PSPACE complete.*

2.4 Some Useful DL-PA Programs

We now define some DL-PA programs that will be useful in the rest of the paper.

The first programs are familiar from PDL: $\mathsf{if}\ \varphi\ \mathsf{then}\ \pi_1\ \mathsf{else}\ \pi_2$ abbreviates $(\varphi?;\pi_1)\sqcup(\neg\varphi?;\pi_2)$ and $\mathsf{while}\ \varphi\ \mathsf{do}\ \pi$ abbreviates $(\varphi?;\pi)^*;\neg\varphi?$. Moreover, we define π^n and $\pi^{\leq n}$ inductively by

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

$$\pi^{\leq n} = \begin{cases} \mathsf{skip} & \text{if } n = 0 \\ (\,\mathsf{skip} \sqcup \pi);\pi^{\leq n-1} & \text{if } n \geq 1 \end{cases}$$

where skip stands for $\top?$ ("nothing happens").

Next, we define assignments of literals to variables:

---

[1] Note that replacements cannot be applied to variables in assignments such as to p in $p{\leftarrow}\top$: the result would not be well-formed.

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

The former assigns to p the truth value of q, while the latter assigns to p the truth value of $\neg q$. So $p{\leftarrow}p$ does nothing and $p{\leftarrow}\neg p$ flips the truth value of p. Note that both abbreviations have constant length, viz. 14.

$$\mathsf{mkTrueOne}(P) = \bigsqcup_{p\in P} (p?; p{\leftarrow}\bot) = (p_1?; p_1{\leftarrow}\top) \sqcup \cdots \sqcup (p_n?; p_n{\leftarrow}\top)$$
$$\mathsf{mkFalseOne}(P) = \bigsqcup_{p\in P} (\neg p?; p{\leftarrow}\top) = (\neg p_1?; p_1{\leftarrow}\bot) \sqcup \cdots \sqcup (\neg p_n?; p_n{\leftarrow}\bot)$$
$$\mathsf{flipOne}(P) = \bigsqcup_{p\in P} (p{\leftarrow}\neg p) = p_1{\leftarrow}\neg p_1 \sqcup \cdots \sqcup p_n{\leftarrow}\neg p_n$$
$$\mathsf{mkTrueSome}(P) = \mathop{;}_{p\in P} (p{\leftarrow}\top \sqcup \ \mathsf{skip}) = (p_1{\leftarrow}\top \sqcup \ \mathsf{skip}) ; \cdots ; (p_n{\leftarrow}\top \sqcup \ \mathsf{skip})$$
$$\mathsf{mkFalseSome}(P) = \mathop{;}_{p\in P} (p{\leftarrow}\bot \sqcup \ \mathsf{skip}) = (p_1{\leftarrow}\bot \sqcup \ \mathsf{skip}) ; \cdots ; (p_n{\leftarrow}\bot \sqcup \ \mathsf{skip})$$
$$\mathsf{flipSome}(P) = \mathop{;}_{p\in P} (p{\leftarrow}\top \sqcup p{\leftarrow}\bot) = (p_1{\leftarrow}\top \sqcup p_1{\leftarrow}\bot) ; \cdots ; (p_n{\leftarrow}\top \sqcup p_n{\leftarrow}\bot)$$

Table 2 DL-PA programs modifying variables in $P = \{p_1, \ldots, p_n\}$

It will be useful to associate sequences of atomic assignments to sets of propositional variables $P = \{p_1, \ldots, p_n\}$ by identifying $\mathop{;}_{p\in P} p{\leftarrow}\top$ with $p_1{\leftarrow}\top; \cdots ; p_n{\leftarrow}\top$ and likewise $\mathop{;}_{p\in P} p{\leftarrow}\bot$ with $p_1{\leftarrow}\bot; \cdots ; p_n{\leftarrow}\bot$. Note that the interpretation does not depend on the order of the variables in P. Note also that this no longer holds if we generalise from atomic assignment programs to complex programs.

Finally, Table 2 collects programs that are going to be instrumental in the rest of the paper. They modify the truth values of one or more propositional variables in a given finite subset $P = \{p_1, \ldots, p_n\}$ of $\mathbb{P}$: We understand that all these programs equal skip when n is zero. The program $\mathsf{mkTrueOne}(P)$ makes exactly one variable true that was not true yet; symmetrically for $\mathsf{mkFalseOne}(P)$. $\mathsf{flipOne}(P)$ nondeterministically chooses a variable from P and nondeterministically sets it to either true or false. It is equivalent to $\mathsf{mkTrueOne}(P) \sqcup \mathsf{mkFalseOne}(P)$. The program $\mathsf{mkTrueSome}(P)$ makes zero or more variables of P true; symmetrically for $\mathsf{mkFalseSome}(P)$. $\mathsf{flipSome}(P)$ nondeterministically sets the value of each variable p_i to either to true or false.

Observe that the length of each program is linear in the cardinality of P. Observe also that the order of the variables in P does not matter. This can be seen from the following lemma characterising the behaviour of the programs. It uses the symmetric difference between two valuations, which is the number of variables whose truth values differs: $v_1 \dot{-} v_2$ is the set of all those p such that $v_1(p) \neq v_2(p)$. This is formally defined as $v_1 \dot{-} v_2 = (v_1 \setminus v_2) \cup (v_2 \setminus v_1)$. For example, the symmetric difference between the valuations $\{p, q\}$ and $\{q, r, s\}$ is $\{p, r, s\}$.

Lemma 1 *Let $P \subseteq \mathbb{P}$ be some set of propositional variables. Then the following hold.*

$$
\begin{aligned}
||\mathsf{mkTrueOne}(P)|| &= \{(v, v') \ : \ v' = v \cup \{p_k\} \textit{ for some } p_k \in P\} \\
||\mathsf{mkFalseOne}(P)|| &= \{(v, v') \ : \ v' = v \setminus \{p_k\} \textit{ for some } p_k \in P\} \\
||\mathsf{flipOne}(P)|| &= \{(v, v') \ : \ v \dot{-} v' = \{p_k\} \textit{ for some } p_k \in P\} \\
||\mathsf{mkTrueSome}(P)|| &= \{(v, v') \ : \ v' = v \cup P \textit{ for some } P \subseteq \mathbb{P}\} \\
||\mathsf{mkFalseSome}(P)|| &= \{(v, v') \ : \ v' = v \setminus P \textit{ for some } P \subseteq \mathbb{P}\} \\
||\mathsf{flipSome}(P)|| &= \{(v, v') \ : \ v \dot{-} v' \subseteq P\}
\end{aligned}
$$

Proof We only prove the cases of $\mathsf{flipSome}(P)$ and $\mathsf{flipOne}(P)$. They are stated without proof in [38, Proposition 5].

For $\mathsf{flipSome}(P)$, if the set $P = \{p_1, \ldots, p_n\}$ is empty then $\mathsf{flipSome}(P) = \mathsf{skip}$ and we are done. If it is a singleton then we have

$$
\begin{aligned}
||\mathsf{flipSome}(\{p_1\})|| &= ||p_1{\leftarrow}\top \sqcup p_1{\leftarrow}\bot|| \\
&= ||p_1{\leftarrow}\top|| \cup ||p_1{\leftarrow}\bot|| \\
&= \big\{(v, v') \ : \ v' = v \cup \{p\}\big\} \cup \big\{(v, v') \ : \ v' = v \setminus \{p\}\big\} \\
&= \big\{(v, v') \ : \ v \dot{-} v' \subseteq \{p\}\big\}
\end{aligned}
$$

Then the result for the general case of an arbitrary set of variables P should be clear (even if the proof is a bit lengthy to spell out).

For $\mathsf{flipOne}(P)$, if P is empty then $\mathsf{flipOne}(P) = \mathsf{skip}$. For a single flip we have:

$$
\begin{aligned}
||p_k{\leftarrow}\neg p_k|| &= ||(p_k?; p_k{\leftarrow}\bot) \sqcup (\neg p_k?; p_k{\leftarrow}\top)|| \\
&= ||p_k?; p_k{\leftarrow}\bot|| \cup ||\neg p_k?; p_k{\leftarrow}\top|| \\
&= \big\{(v, v') \ : \ p_k \in v \text{ and } p_k \notin v'\big\} \ \cup \ \big\{(v, v') \ : \ p_k \notin v \text{ and } p_k \in v'\big\} \\
&= \big\{(v, v') \ : \ v \dot{-} v' = \{p_k\}\big\}
\end{aligned}
$$

Therefore $||\mathsf{flipOne}(P)||$ equals $\{(v, v') \ : \ v \dot{-} v' = \{p_k\} \text{ for some } p_k \in P\}$. □

Consider the set of propositional variables $\mathbb{P} = \{p, q, r\}$. The set of valuations for that language is $2^{\mathbb{P}}$. Consider the formula $\varphi = p \wedge \neg q$. Then $\mathsf{flipSome}(\mathbb{P}(\varphi)) = \mathsf{flipSome}(\{p, q\}) = (p{\leftarrow}\top \sqcup p{\leftarrow}\bot)\,;\,(q{\leftarrow}\top \sqcup q{\leftarrow}\bot)$. Its interpretation varies the truth values of p and q in all possible ways: $||\mathsf{flipSome}(\mathbb{P}(\varphi))|| = \big\{(v, v') \ : \ v \dot{-} v' \subseteq \{p, q\}\big\}$. The interpretation of the test $\varphi?$ is $\{(\{p\}, \{p\}), (\{p, r\}, \{p, r\})\}$. The sequential composition of $\mathsf{flipSome}(\mathbb{P}(\varphi))$ and $\varphi?$ therefore is:

$$
||\mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi?|| = \big\{(v, v') \ : \ v' \in ||\varphi|| \text{ and } v \dot{-} v' \subseteq \{p, q\}\big\}.
$$

So the pair $(\{p\}, \{q\})$ is a possible execution of the program $\mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi?$.

More generally, programs of the form $\mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi?$ accesses all relevant φ-models, where 'relevant' means that the truth values of variables not occurring in φ are keep constant. It follows that the satisfiability of a boolean formula φ can be expressed in DL-PA. The result below is stated without proof in [38].

Lemma 2 *Let φ be a* DL-PA *formula. φ is satisfiable iff* $\langle \mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi? \rangle \top$ *is valid.*

Proof Suppose $\langle \mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi? \rangle \top$ is DL-PA valid. Let v be some valuation. By the interpretation of the dynamic operator there exists a valuation v' such that $(v, v') \in ||\mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi?||$. This means that there exists a valuation v'' such that $(v, v'') \in ||\mathsf{flipSome}(\mathbb{P}(\varphi))||$ and $(v'', v') \in ||\varphi?||$. The latter means that $v' \in ||\varphi||$: we have found a valuation where φ is true.

The other way round, suppose φ is propositionnally satisfiable, i.e., there is some model v_φ of φ. Let v be an arbitrary valuation. Let v'_φ be the valuation which interprets the variables of φ in the same way as v_A and interprets the other variables in the same way as v:

$$v'_\varphi = (v_\varphi \cap \mathbb{P}(\varphi)) \cup (v \cap (\mathbb{P} \setminus \mathbb{P}(\varphi)))$$

This is clearly also an φ-model. (Indeed, for every p that does not occur in φ we have $v \cup \{p\} \in ||\varphi||$ iff $v \setminus \{p\} \in ||\varphi||$.) As $v \dot{-} v'_\varphi \subseteq \mathbb{P}(\varphi)$ we have $(v, v'_\varphi) \in ||\mathsf{flipSome}(\mathbb{P}(\varphi))||$ by Item 1 of Lemma 1. And as v'_φ is an φ-model we have $(v'_\varphi, v'_\varphi) \in ||\varphi?||$. So $(v, v'_\varphi) \in ||\mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi?||$, from which it follows that $v \in ||\langle \mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi? \rangle \top||$. As v was arbitrary, $\langle \mathsf{flipSome}(\mathbb{P}(\varphi)); \varphi? \rangle \top$ is DL-PA valid. □

3 Describing AFs and their Semantics by Formulas

In the present section we recall the main definitions of abstract AFs. We do so in terms of the language of DL-PA. We show in particular that minimization-based and maximization-based semantics can be handled in an elegant way by taking advantage of the programs of DL-PA.

3.1 AFs and their Representation in Propositional Logic

We suppose given a finite set of arguments $\mathbb{A} = \{a_1, \ldots, a_n\}$. We associate to each couple of arguments $(a, b) \in \mathbb{A}^2$ an *attack variable* $\mathsf{r}_{a,b}$. Furthermore, we associate to every argument $a \in \mathbb{A}$ an *acceptance variable* in_a expressing that a is accepted. So the respective sets of such variables are:

$$\mathtt{ATT}^{\mathbb{A}} = \{\mathsf{r}_{a,b} \ : \ (a, b) \in \mathbb{A} \times \mathbb{A}\}$$
$$\mathtt{IN}^{\mathbb{A}} = \{\mathsf{in}_a \ : \ a \in \mathbb{A}\}$$

In the rest of the paper we suppose that the set of propositional variables is $\mathbb{P} = \mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}}$. So our language is $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$; the set $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$ are the formulas that can be built from attack variables only, and $\mathsf{Lang}(\mathtt{IN}^{\mathbb{A}})$ are those that can be built from acceptance variables only.

An abstract AF as defined by Dung in [31] is a pair $\mathsf{G} = (\mathbb{A}, \mathsf{R})$ where $\mathsf{R} \subseteq \mathbb{A} \times \mathbb{A}$ is the attack relation. So AFs are nothing but finite directed graphs with arguments as vertices and attacks as edges. The *theory of* $\mathsf{G} = (\mathbb{A}, \mathsf{R})$ is the boolean formula

$$\theta(\mathsf{G}) = \left(\bigwedge_{(a,b)\in\mathsf{R}} \mathsf{r}_{a,b} \right) \wedge \left(\bigwedge_{(a,b)\in(\mathbb{A}\times\mathbb{A})\setminus\mathsf{R}} \neg\mathsf{r}_{a,b} \right).$$

We clearly have that $\mathsf{r}_{a,b} \in ||\theta(\mathsf{G})||$ if and only if $(a, b) \in \mathsf{R}$.

Just as $\theta(\mathsf{G})$ represents the attack relation in propositional logic, an extension $E \subseteq \mathbb{A}$ can be represented by the formula

$$\theta(E, \mathbb{A}) = \left(\bigwedge_{a\in E} \mathsf{in}_a \right) \wedge \left(\bigwedge_{a\in\mathbb{A}\setminus E} \neg\mathsf{in}_a \right).$$

3.2 Semantics of an AF

Many semantics were defined for acceptability. Some of them are extension-based, some others are labelling-based. We here only consider the first: they prevail in the literature, and moreover labelling-based semantics have equivalent extension-based formulations [6].

As proposed in [12] and extensively discussed in [34], many definitions of extensions can be captured in propositional logic. In this paper we consider the following.

A *stable extension of* $(\mathbb{A}, \mathsf{R})$ is a set of arguments $E \subseteq \mathbb{A}$ such that it does not exist two arguments a and b in E such that $(a, b) \in \mathsf{R}$ (that is, E is *conflict-free*), and for each argument $b \notin E$, there exists $a \in E$ such that $(a, b) \in \mathsf{R}$ (any argument outside the extension is attacked by the extension).

An *admissible set of* $(\mathbb{A}, \mathsf{R})$ is a conflict-free set $E \subseteq \mathbb{A}$ that *defends* all its elements: for all $a \in E$, if there exists b such that $(b, a) \in \mathsf{R}$, then there is some argument $c \in E$ such that $(c, b) \in \mathsf{R}$.[2]

A *complete extension of* $(\mathbb{A}, \mathsf{R})$ is an admissible set $E \subseteq \mathbb{A}$ that contains all the arguments it defends, that is: if an argument a is such that, for all b such that $(b, a) \in \mathsf{R}$, there exists some $c \in E$ such that $(c, b) \in \mathsf{R}$, then $a \in E$.

A *preferred extension of* $(\mathbb{A}, \mathsf{R})$ is an admissible set $E \subseteq \mathbb{A}$ that is maximal w.r.t. inclusion.

A *grounded extension of* $(\mathbb{A}, \mathsf{R})$ is a complete extension $E \subseteq \mathbb{A}$ that is minimal w.r.t. inclusion.

Their definitions in propositional logic are collected in Table 3. Given a definition of a semantics σ (stable, admissible, or complete) and a set of arguments $\mathbb{A}$, $\theta(\sigma, \mathbb{A})$ denotes the associated logical characterisation of σ. Computing the extensions of

[2] Admissibility is usually considered to be a building brick of other, stronger semantics, but we consider it here as a semantics on its own, the construction of admissible sets following the same process as the construction of extensions.

$$\theta(\text{stable}, \mathbb{A}) = \bigwedge_{a\in\mathbb{A}} \mathsf{in}_a \leftrightarrow \bigwedge_{b\in\mathbb{A}} (\mathsf{r}_{b,a} \rightarrow \neg\mathsf{in}_b)\Big)$$

$$\theta(\text{admissible}, \mathbb{A}) = \bigwedge_{a\in\mathbb{A}} \mathsf{in}_a \rightarrow \bigwedge_{b\in\mathbb{A}} \Big(\mathsf{r}_{b,a} \rightarrow \Big(\neg\mathsf{in}_b \wedge \bigvee_{c\in\mathbb{A}} (\mathsf{in}_c \wedge \mathsf{r}_{c,b})\Big)\Big)\Big)$$

$$\theta(\text{complete}, \mathbb{A}) = \bigwedge_{a\in\mathbb{A}} \Big(\Big(\mathsf{in}_a \rightarrow \bigwedge_{b\in\mathbb{A}} (\mathsf{r}_{b,a} \rightarrow \neg\mathsf{in}_b)\Big) \wedge \Big(\mathsf{in}_a \leftrightarrow \bigwedge_{b\in\mathbb{A}} \Big(\mathsf{r}_{b,a} \rightarrow \bigvee_{c\in\mathbb{A}} (\mathsf{in}_c \wedge \mathsf{r}_{c,b})\Big)\Big)\Big)$$

$$\theta(\text{preferred}, \mathbb{A}) = \theta(\text{admissible}, \mathbb{A}) \wedge \Big[\mathsf{mkTrueOne}(\mathbb{A}); \mathsf{mkTrueSome}(\mathbb{A})\Big]\neg\theta(\text{admissible}, \mathbb{A})$$

$$\theta(\text{grounded}, \mathbb{A}) = \theta(\text{complete}, \mathbb{A}) \wedge \Big[\mathsf{mkFalseOne}(\mathbb{A}); \mathsf{mkFalseSome}(\mathbb{A})\Big]\neg\theta(\text{complete}, \mathbb{A})$$

Table 3 Argumentation semantics $\theta(\sigma, \mathbb{A})$ in propositional logic, for σ being the stable, admissible, complete, preferred, or grounded semantics

an AF G under semantics σ amounts to finding the models of the conjunction $\mathsf{G} \wedge \theta(\sigma, \mathbb{A})$. The following proposition connects extensions and valuations for all the above semantics.

Proposition 1 *Let $\mathsf{G} = (\mathbb{A}, \mathsf{R})$ be an AF and let $E \subseteq \mathbb{A}$. Let σ be either the stable, admissible, complete, preferred or grounded semantics. E is a σ-extension of G if and only if $\big(\theta(\mathsf{G}) \wedge \theta(E, \mathbb{A})\big) \rightarrow \theta(\sigma, \mathbb{A})$ is* DL-PA *valid.*

Proof The result for stable, admissible, and complete semantics follow from propositions 5, 6, and 8 in [12].

The results for the preferred and the grounded semantics follows from the following program equivalences:

$$||\mathsf{mkTrueOne}(P); \mathsf{mkTrueSome}(P)|| = \big\{(v, v \cup P') \ : \ v \subseteq 2^{\mathbb{P}} \text{ and } \emptyset \subset P' \subseteq P\big\}$$

$$||\mathsf{mkFalseOne}(P); \mathsf{mkFalseSome}(P)|| = \big\{(v \cup P', v) \ : \ v \subseteq 2^{\mathbb{P}} \text{ and } \emptyset \subset P' \subseteq P\big\}$$

where P is some set of propositional variables. □

Corollary 1 *Let $\mathsf{G} = (\mathbb{A}, \mathsf{R})$ be an AF. Let σ be either the stable, admissible, complete, preferred or grounded semantics.*

1. *$E \subseteq \mathbb{A}$ is a σ-extension of G if and only if there is a $v \in ||\theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A})||$ such that $E = \{a \ : \ \mathsf{in}_a \in v\}$.*
2. *Let $\gamma \in \mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ be a formula describing some property of $\mathbb{A}$. All σ-extensions of G have property γ if and only if $\theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A}) \rightarrow \gamma$ is* DL-PA *valid.*

Proof The first item summarises the preceding proposition. (Observe that when $v \in ||\mathsf{G}||$ then $E = \{a \ : \ \mathsf{in}_a \in v\}$ implies that $v = v_E$.)

As to the second item, by Proposition 1, γ is true in all extensions of G if and only if γ is true in every model of $\theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A})$, i.e., iff $\theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A}) \rightarrow \gamma$ is DL-PA valid. □

An AF may have none, one or several extensions, depending on the number of models of $\theta(\mathsf{G}) \wedge \theta(E, \mathbb{A})$.

Example

Consider the set of arguments $\mathbb{A}_1 = \{a, b\}$ and the two AFs $\mathsf{G}_1 = (\mathbb{A}_1, \mathsf{R}_1)$ and $\mathsf{G}_2 = (\mathbb{A}_1, \mathsf{R}_2)$, with $\mathsf{R}_1 = \{(a, b)\}$ and $\mathsf{R}_2 = \{(a, b), (b, a)\}$. They are depicted in Figure 1.

Fig. 1 G_1 (left) and G_2 (right)

The associated propositional theories are:

$$\theta(\mathsf{G}_1) = \mathsf{r}_{a,b} \wedge \neg \mathsf{r}_{b,a} \wedge \neg \mathsf{r}_{a,a} \wedge \neg \mathsf{r}_{b,b}$$
$$\theta(\mathsf{G}_2) = \mathsf{r}_{a,b} \wedge \mathsf{r}_{b,a} \wedge \neg \mathsf{r}_{a,a} \wedge \neg \mathsf{r}_{b,b}$$

The AF G_1 has a single stable extension $\{a\}$, two admissible extensions $\emptyset$ and $\{a\}$, and one complete extension $\{a\}$; that is:

$$\theta(\mathsf{G}_1) \wedge \theta(\mathsf{stable}, \mathbb{A}_1) \longrightarrow \mathsf{in}_a \wedge \neg \mathsf{in}_b$$
$$\theta(\mathsf{G}_1) \wedge \theta(\mathsf{admissible}, \mathbb{A}_1) \longrightarrow \neg \mathsf{in}_b$$
$$\theta(\mathsf{G}_1) \wedge \theta(\mathsf{complete}, \mathbb{A}_1) \longrightarrow \mathsf{in}_a \wedge \neg \mathsf{in}_b$$

The *justification state* of an argument [6] depends on the extensions it belongs to: it is *credulously justified in AF* G *under semantics* σ if it belongs to at least one of the σ-extensions of G. It is *skeptically justified in* G *under* σ if it belongs to every σ-extension of G.

4 Building Extensions by Programs

In the last section we have seen how to define by formulas what an extension is. In practice such extensions are constructed by algorithms. The DL-PA programs provide a means to recast these algorithms in a logical framework. Remember that such programs modify valuations: our aim is to define programs which when executed at a valuation describing an AF modify the truth values of the acceptance variables, leading to valuations where the formula characterizing the semantics holds.

Remember that $\mathtt{ATT}^{\mathbb{A}}(\varphi)$ and $\mathtt{IN}^{\mathbb{A}}(\varphi)$ respectively are the set of attack variables and the set of acceptance variables occurring in the formula φ. For example, $\mathtt{ATT}^{\mathbb{A}}(\neg \mathsf{r}_{a,b} \vee \mathsf{in}_a) = \{\mathsf{r}_{a,b}\}$ and $\mathtt{IN}^{\mathbb{A}}(\neg \mathsf{r}_{a,b} \vee \mathsf{in}_a) = \{\mathsf{in}_a\}$.

4.1 A Generic DL-PA Program

The generic DL-PA program $\mathsf{mkExt}^{\sigma}_{\mathbb{A}}$ below is indexed by a set of arguments $\mathbb{A}$ and a semantics σ and builds for every AF $\mathtt{G}$ over $\mathbb{A}$ the valuations representing the extensions of $\mathtt{G}$ w.r.t. σ:

$$\mathsf{mkExt}(\sigma, \mathbb{A}) = \mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}}); \theta(\sigma, \mathbb{A})?$$

where n is the cardinality of $\mathbb{A}$, σ is some semantics, and $\theta(\sigma, \mathbb{A})$ is the DL-PA formula characterising σ.

Proposition 2 *Let $\mathbb{A}$ be a set of arguments. Let σ be either the stable, complete, admissible, preferred, or grounded semantics. Then*

$$||\mathsf{mkExt}(\sigma, \mathbb{A})|| = \left\{(v_1, v_2) \ : \ v_2 \in \theta(\sigma, \mathbb{A}) \textit{ and } v_1 \cap \mathtt{ATT}^{\mathbb{A}} = v_2 \cap \mathtt{ATT}^{\mathbb{A}}\right\}.$$

Proof When $(v_1, v_2) \in ||\mathsf{mkExt}(\sigma, \mathbb{A})||$ then there is a valuation that is accessible from v_1 via the relation $||\mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}})||$ and from which v_2 can be accessed via $||\theta(\sigma, \mathbb{A})?||$. As the test program $\theta(\sigma, \mathbb{A})?$ does not modify any truth value that valuation must be v_2 itself, which moreover must be a model of $\theta(\sigma, \mathbb{A})$. By Item 1 of Lemma 1, v_2 differs from v_1 only by the truth values of the $\mathtt{IN}^{\mathbb{A}}$ variables. It therefore has the same truth values for the $\mathtt{ATT}^{\mathbb{A}}$ variables, i.e., $v_1 \cap \mathtt{ATT}^{\mathbb{A}} = v_2 \cap \mathtt{ATT}^{\mathbb{A}}$. □

It follows that the DL-PA program $\mathsf{mkExt}(\sigma, \mathbb{A})$ constructs all the extensions of a given AF w.r.t. σ. Indeed, when the set of attack variables of v_1 describes $\mathtt{G}$ then the set of v_2 such that $(v_1, v_2) \in ||\mathsf{mkExt}(\sigma, \mathbb{A})||$ characterises all the extensions of $\mathtt{G}$ w.r.t. σ.

Corollary 2 *Let $\mathtt{G} = (\mathbb{A}, \mathtt{R})$ be an AF and let φ be a formula. The formula $\theta(\mathtt{G}) \wedge \theta(\sigma, \mathbb{A}) \rightarrow \varphi$ is valid if and only if $\theta(\mathtt{G}) \rightarrow [\mathsf{mkExt}(\sigma, \mathbb{A})]\varphi$ is valid. Moreover, the equivalence*

$$\theta(\mathtt{G}) \wedge \theta(\sigma, \mathbb{A}) \leftrightarrow \langle(\mathsf{mkExt}(\sigma, \mathbb{A}))^{-}\rangle\theta(\mathtt{G})$$

is valid.

According to Corollary 2, given a description of the attack relation, $\mathsf{mkExt}(\sigma, \mathbb{A})$ constructs all the valuations where the formula describing the semantics is true, and so in a way such that only acceptance variables are changed, while the attack variables do not change. Note that the equivalence takes care of cases where $\mathtt{G}$ has no extension.

Suppose the formula φ is a goal, i.e., a formula that should be satisfied by all or some of the extensions of some AF. By means of the program $\mathsf{mkExt}(\sigma, \mathbb{A})$ one

can check what has been called σ-*consistency* in [22]: whether there exists an AF G such that some or every extension w.r.t. σ satisfies φ. Adopting the standard terms for characterizing a justification state, we distinguish the notions of credulous and skeptical consistency.

Definition 1 Let σ be the stable, admissible, complete, preferred, or grounded semantics. Let φ be an $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ formula. Then

- φ is σ-*credulously consistent for* $\mathbb{A}$ iff $\langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\varphi$ is satisfiable;
- φ is σ-*skeptically consistent for* $\mathbb{A}$ iff $[\mathsf{mkExt}(\sigma, \mathbb{A})]\varphi$ is satisfiable.

Example

Consider the set of arguments $\mathbb{A}_2 = \{a, b\}$ and the goal $\varphi = \neg\mathsf{in}_a \wedge \neg\mathsf{in}_b \wedge \neg\mathsf{r}_{a,a}$. Then $\langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\varphi$ is unsatisfiable; in particular, it is false in every valuation for $\theta(\mathsf{G}_2)$ of Figure 1. Therefore the formula φ is stable-credulously inconsistent for $\mathbb{A}_2$.

The length of $\mathsf{mkExt}(\sigma, \mathbb{A})$ is linear in the length of the logical description of the semantics $\theta(\sigma, \mathbb{A})$ and in the cardinality of the set of arguments $\mathbb{A}$, The program is however heavily nondeterministic. There are other, more efficient algorithms that can also be implemented in DL-PA. We turn to that in the rest of the section.

4.2 More Efficient Algorithms

The above programs $\mathsf{mkExt}(\sigma, \mathbb{A})$ implement a generate-and-test schema where all logically possible valuations are generated. An improvement is the program in Table 4: before accepting an argument it checks whether this would lead to a conflict with previously accepted arguments. (Remember that the set of arguments $\mathbb{A}$ is $\{a_1, \ldots, a_n\}$.)

$$\mathsf{mkExt}_1(\sigma, \mathbb{A}) = \Big(\mathop{;}_{\mathsf{in}_a \in \mathtt{IN}^{\mathbb{A}}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \bigwedge_{b \in \{a_1\}} \Big(\mathsf{in}_b \rightarrow (\neg\mathsf{r}_{a_1,b} \wedge \neg\mathsf{r}_{b,a_1})\Big)?; \mathsf{in}_{a_1} \leftarrow \top\Big); \cdots;$$
$$\Big(\mathsf{skip} \sqcup \bigwedge_{b \in \{a_1, \ldots, a_n\}} \Big(\mathsf{in}_b \rightarrow (\neg\mathsf{r}_{a_n,b} \wedge \neg\mathsf{r}_{b,a_n})\Big)?; \mathsf{in}_{a_n} \leftarrow \top\Big); \theta(\sigma, \mathbb{A})?$$

Table 4 A more efficient DL-PA program

Theorem 3 *The program equivalence* $\mathsf{mkExt}_1(\sigma, \mathbb{A}) \equiv \mathsf{mkExt}(\sigma, \mathbb{A})$ *holds.*

Proof The proof takes the form of a sequence of DL-PA programs that are all DL-PA equivalent:

1. $\mathsf{mkExt}_1(\sigma, \mathbb{A})$
2. $\Big(;_{\mathsf{in}_a \in \mathtt{IN}^{\mathbb{A}}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \bigwedge_{b \in \{a_1\}} \Big(\mathsf{in}_b \rightarrow (\neg \mathsf{r}_{a_1,b} \wedge \neg \mathsf{r}_{b,a_1})\Big)?; \mathsf{in}_{a_1} \leftarrow \top\Big); \cdots;$
$\Big(\mathsf{skip} \sqcup \bigwedge_{b \in \{a_1,\ldots,a_n\}} \Big(\mathsf{in}_b \rightarrow (\neg \mathsf{r}_{a_n,b} \wedge \neg \mathsf{r}_{b,a_n})\Big)?; \mathsf{in}_{a_n} \leftarrow \top\Big); \theta(\sigma, \mathbb{A})?$
3. $\Big(;_{\mathsf{in}_a \in \mathbb{A}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \Big(\mathsf{in}_{a_1} \leftarrow \top; \bigwedge_{b \in \{a_1\}} \Big((\mathsf{in}_{a_1} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_1,b} \wedge \neg \mathsf{r}_{b,a_1})\Big)?\Big)\Big); \cdots;$
$\Big(\mathsf{skip} \sqcup \Big(\mathsf{in}_{a_n} \leftarrow \top; \bigwedge_{b \in \{a_1,\ldots,a_n\}} \Big((\mathsf{in}_{a_n} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_n,b} \wedge \neg \mathsf{r}_{b,a_n})\Big)?\Big)\Big); \theta(\sigma, \mathbb{A})?$
4. $\Big(;_{\mathsf{in}_a \in \mathbb{A}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \mathsf{in}_{a_1} \leftarrow \top\Big); \bigwedge_{b \in \{a_1\}} \Big((\mathsf{in}_{a_1} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_1,b} \wedge \neg \mathsf{r}_{b,a_1})\Big)?; \cdots;$
$\Big(\mathsf{skip} \sqcup \mathsf{in}_{a_n} \leftarrow \top\Big); \bigwedge_{b \in \{a_1,\ldots,a_n\}} \Big((\mathsf{in}_{a_n} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_n,b} \wedge \neg \mathsf{r}_{b,a_n})\Big)?; \theta(\sigma, \mathbb{A})?$
5. $\Big(;_{\mathsf{in}_a \in \mathbb{A}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \mathsf{in}_{a_1} \leftarrow \top\Big); \cdots; \Big(\mathsf{skip} \sqcup \mathsf{in}_{a_n} \leftarrow \top\Big);$
$\bigwedge_{b \in \{a_1\}} \Big((\mathsf{in}_{a_1} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_1,b} \wedge \neg \mathsf{r}_{b,a_1})\Big)?; \cdots;$
$\bigwedge_{b \in \{a_1,\ldots,a_n\}} \Big((\mathsf{in}_{a_n} \wedge \mathsf{in}_b) \rightarrow (\neg \mathsf{r}_{a_n,b} \wedge \neg \mathsf{r}_{b,a_n})\Big)?; \theta(\sigma, \mathbb{A})?$
6. $\Big(;_{\mathsf{in}_a \in \mathbb{A}} \mathsf{in}_a \leftarrow \bot\Big); \Big(\mathsf{skip} \sqcup \mathsf{in}_{a_1} \leftarrow \top\Big); \cdots; \Big(\mathsf{skip} \sqcup \mathsf{in}_{a_n} \leftarrow \top\Big); \theta(\sigma, \mathbb{A})?$
7. $\mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}}); \theta(\sigma, \mathbb{A})?$ □

The above theorem states a program equivalence of the logic that is parametrised by the cardinality n of the set of arguments $\mathbb{A}$. While the above proof was by hand, it could as well be done (for a given n) by means of some automated theorem prover for DL-PA. While theoretical results establishing the complexity of model checking and satisfiability exist, there is for the time being no implemented theorem prover for DL-PA.

Actually many algorithms were proposed in the literature to build extensions, such as those of [19] or of [46] (the latter for the preferred semantics). We claim that all of them can be implemented in DL-PA. While we do not work this out here and leave it to future work, we nevertheless sketch how labelling-based algorithms can be captured in DL-PA. We use new propositional variables dec_a, standing for "the status of a is settled (decided)". The following abbreviations will be useful.

$$\mathsf{AttackedByAcc}(a) = \bigvee_{b \in \mathbb{A}} \Big(\mathsf{dec}_b \wedge \mathsf{in}_b \wedge \mathsf{r}_{b,a}\Big)$$

$$\mathsf{DefendedByAcc}(a) = \bigwedge_{b \in \mathbb{A}} \Big(\mathsf{r}_{b,a} \rightarrow \bigvee_{c \in \mathbb{A}} \Big(\mathsf{dec}_c \wedge \mathsf{in}_c \wedge \mathsf{r}_{c,b}\Big)\Big)$$

Then we define the following program:

$$
\begin{aligned}
\mathsf{mkExt}_2(\sigma, \mathbb{A}) = \ & \mathop{;}_{\mathsf{dec}_a} \mathsf{dec}_a \leftarrow \bot \ ; \\
& \Big(\mathsf{if} \bigwedge_{b \in \mathbb{A}} \neg \mathsf{r}_{b,a_1} \ \mathsf{then} \ \mathsf{in}_{a_1} \leftarrow \top; \mathsf{dec}_{a_1} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \cdots ; \\
& \Big(\mathsf{if} \bigwedge_{b \in \mathbb{A}} \neg \mathsf{r}_{b,a_n} \ \mathsf{then} \ \mathsf{in}_{a_n} \leftarrow \top; \mathsf{dec}_{a_n} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \\
& \mathsf{while} \bigvee_a \neg \mathsf{dec}_a \ \mathsf{do} \\
& \quad \mathsf{while} \bigvee_a \Big((\mathsf{AttackedByAcc}(a) \vee \mathsf{DefendedByAcc}(a))\Big) \ \mathsf{do} \\
& \qquad \Big(\mathsf{if} \ \mathsf{AttackedByAcc}(a_1) \ \mathsf{then} \ \mathsf{in}_{a_1} \leftarrow \bot; \mathsf{dec}_{a_1} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \\
& \qquad \Big(\mathsf{if} \ \mathsf{DefendedByAcc}(a_1) \ \mathsf{then} \ \mathsf{in}_{a_1} \leftarrow \top; \mathsf{dec}_{a_1} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \cdots ; \\
& \qquad \Big(\mathsf{if} \ \mathsf{AttackedByAcc}(a_n) \ \mathsf{then} \ \mathsf{in}_{a_n} \leftarrow \bot; \mathsf{dec}_{a_n} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \\
& \qquad \Big(\mathsf{if} \ \mathsf{DefendedByAcc}(a_n) \ \mathsf{then} \ \mathsf{in}_{a_n} \leftarrow \top; \mathsf{dec}_{a_n} \leftarrow \top \ \mathsf{else} \ \ \mathsf{skip}\Big) \ ; \\
& \quad \mathsf{if} \bigwedge_a \mathsf{dec}_a \ \mathsf{then} \ \ \mathsf{skip} \ \mathsf{else} \bigsqcup_{a \in \mathbb{A}} \Big(\neg \mathsf{dec}_a?; (\mathsf{in}_a \leftarrow \top \sqcup \mathsf{in}_a \leftarrow \bot); \mathsf{dec}_a \leftarrow \top\Big); \\
& \theta(\sigma, \mathbb{A})?
\end{aligned}
$$

We do not prove the correctness of the program here.

To sum it up, we have seen up to now how extensions of a given AF can be constructed via DL-PA programs. In the rest of the paper we will use DL-PA programs to modify an AF and its extensions in order to fulfil some goal.

5 What is Argumentation Framework Modification?

We now introduce and discuss the problem of AF modification. We first distinguish three different kinds of modification and then discuss whether they correspond to a revision or to an update operation.

5.1 What changes?

Cayrol et al. [18] proposed to distinguish several kinds of modifications of a given AF $\mathsf{G} = (\mathbb{A}, \mathsf{R})$, according to the goal that is pursued:

1. Modifications of the set of arguments $\mathbb{A}$ (by adding or removing an argument);
2. Modifications of the attack relation R (by adding or removing an edge between two arguments);
3. Modifications of the extensions of G in order to satisfy some property.

Modifications of the semantics that is applied to the AF, to produce the set of extensions, can also be considered, as outlined in [30]. Such modifications are left for future work.

5.1.1 Changes on the set of arguments

The first kind of modification presented here, viz. adding an argument to the set of arguments $\mathbb{A}$ or removing it, requires additional linguistic resources that have to be added to our language for it to be implemented.

First of all, we have to add a further ingredient to AFs: let us consider triples $\mathtt{G} = (\mathbb{A}, \mathbb{A}^{En}, \mathtt{R})$ where $\mathbb{A}$ is the finite background set of all possible arguments, while $\mathbb{A}^{En} \subseteq \mathbb{A}$ is the set of all arguments that are currently under consideration (arguments that are *enabled*). We add to the logical language a set of enablement variables

$$\mathtt{EN}^{\mathbb{A}} = \{\mathsf{En}_a \ : \ a \in \mathbb{A}\}$$

where En_a means that a is enabled (or "considered"). $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{EN}^{\mathbb{A}})$ being the set of all formulas that are built from variables $\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{EN}^{\mathbb{A}}$, the theory of $\mathtt{G} = (\mathbb{A}, \mathbb{A}^{En}, \mathtt{R})$ is the boolean formula

$$\theta(\mathtt{G}) = \left(\bigwedge_{(a,b)\in \mathtt{R}} \mathsf{r}_{a,b} \right) \wedge \left(\bigwedge_{(a,b)\notin \mathtt{R}} \neg \mathsf{r}_{a,b} \right) \wedge \left(\bigwedge_{a\in \mathbb{A}^{En}} \mathsf{En}_a \right) \wedge \left(\bigwedge_{a\notin \mathbb{A}^{En}} \neg \mathsf{En}_a \right)$$

Note that in the theory, $\mathsf{r}_{a,b}$ is true even if a or b are not enabled.

The semantic definitions have to be adapted, too, and should only quantify over arguments in $\mathbb{A}^{En}$ and not over those in $\mathbb{A}$. Also, attacks must be considered only if they link enabled arguments. To this end, we define the following formula:

$$\mathsf{ATT}^{En}_{a,b} = \mathsf{r}_{a,b} \wedge \mathsf{En}_a \wedge \mathsf{En}_b$$

Now we can easily transform the semantic formulas: we check if the argument is enabled, otherwise it will not be included in the extension, and we replace attacks variables $\mathsf{r}_{a,b}$ by formulas $\mathsf{ATT}^{En}_{a,b}$ to ensure they are indeed present. We illustrate this transformation to capture the stable semantics. Let $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{EN}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ be the language of formulas built from $\mathbb{P} = \mathtt{ATT}^{\mathbb{A}} \cup \mathtt{EN}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}}$. The following formula captures the stable semantics[3]:

$$\theta(\mathsf{stable}, \mathbb{A}) = \bigwedge_{a\in\mathbb{A}} \left(\left(\mathsf{En}_a \rightarrow (\mathsf{in}_a \leftrightarrow \neg \bigvee_{b\in\mathbb{A}} (\mathsf{in}_b \wedge \mathsf{ATT}^{En}_{b,a})) \right) \wedge \left(\neg \mathsf{En}_a \rightarrow \neg \mathsf{in}_a \right) \right)$$

[3] An equivalent way to express these formulas would be to use the set of enabled arguments. For example, to describe the stable extensions we would write:
$\theta(\mathsf{stable}, \mathbb{A}, \mathbb{A}^{\mathsf{En}}) = \bigwedge_{a\in\mathbb{A}^{En}} (\mathsf{in}_a \leftrightarrow \neg \bigvee_{b\in\mathbb{A}^{En}} (\mathsf{in}_b \wedge \mathsf{r}_{b,a})) \wedge \bigwedge_{a\notin\mathbb{A}^{En}} \neg \mathsf{in}_a$.
This highlights the fact that when all arguments are enabled, i.e., when $\mathbb{A}^{En} = \mathbb{A}$, we indeed retrieve the formulas presented earlier in the paper.

In this setting, the mere addition or deletion of an argument a can be achieved straightforwardly, viz. by changing the status of a from 'disabled' to 'enabled' by means of the assignments $\mathsf{En}_a \leftarrow \top$ and $\mathsf{En}_a \leftarrow \bot$. When π is a DL-PA program describing one or a sequence of modifications of that kind then the modification of G by π is described in DL-PA by the formula

$$\langle \pi^- \rangle \theta(\mathsf{G})$$

which says that the program π was possibly executed and $\theta(\mathsf{G})$ was true before. Since attacks are already in the theory even if arguments are not considered, we do not need to include them in our update: all the attacks from (resp. to) a to (resp. from) other enabled arguments, are considered. Attack can however be removed or added; this is the kind of modification that we present in the next section.

A full description and an illustration of the setting presented here for the modification of the set of arguments can be found in [29].

5.1.2 Changes on the set of attacks

The second kind of modification, addition and removal of an attack edge between two arguments $a, b \in \mathbb{A}$, is rather simple: we just add or subtract (a, b) from R. In our logical representation, this operation corresponds to making propositional variables true or false, which can be immediately captured by DL-PA programs assigning the attack variables to true and false, $\mathsf{r}_{a,b} \leftarrow \top$ and $\mathsf{r}_{a,b} \leftarrow \bot$. Like for the addition and the removal of arguments, when π is a DL-PA program describing one or a sequence of modifications of attacks, then the modification of G by π is described in DL-PA by the formula

$$\langle \pi^- \rangle \theta(\mathsf{G})$$

which says that the program π was possibly executed and $\theta(\mathsf{G})$ was true before.

To illustrate this let us take up our above AF $\mathsf{G}_2 = (\mathbb{A}_2, \mathsf{R}_2)$ with $\mathbb{A}_2 = \{a, b\}$ and $\mathsf{R}_2 = \{(a, b), (b, a)\}$. The removal of the attack $\mathsf{r}_{b,a}$ results in $(\mathbb{A}_2, \mathsf{R}'_2)$ with $\mathsf{R}'_2 = \{(a, b)\}$. In DL-PA, the modification of G_2 by $\mathsf{r}_{b,a} \leftarrow \bot$ is described by the formula $\langle \mathsf{r}_{b,a} \leftarrow \bot^- \rangle \theta(\mathsf{G}_2)$.

5.1.3 Changes on the extensions

The third kind of modification, adding arguments to or removing them from extensions, is more involved because it has to be achieved indirectly, by changing the underlying attack relation of G (or by changing the set of arguments, but as we said, we disregard this option for the time being). There are moreover two different options here: when an AF has several extensions one may wish to change the justification status of a, so that it be skeptically justified (that is, added in all the extensions), or credulously justified (added in at least one extension), or credulously justified but

not skeptically justified (added in at least one extension, but removed from at least one), or even not credulously justified (removed from all extensions).

In the rest of the section we explore the issue of extension modification a bit further. Consider again the above AF $\mathsf{G}_2 = (\mathbb{A}_2, \mathsf{R}_2)$ and the stable semantics. G_2 has two stable extensions $E_a = \{a\}$ and $E_b = \{b\}$. We have seen that the associated boolean formula $\theta(\mathsf{G}_2) \wedge \theta(\mathsf{stable}, \mathbb{A}_2)$ has two $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ models:

$$v_a = \{\mathsf{r}_{a,b}, \mathsf{r}_{b,a}, \mathsf{in}_a\}$$
$$v_b = \{\mathsf{r}_{a,b}, \mathsf{r}_{b,a}, \mathsf{in}_b\}$$

Suppose we wish to modify G_2 such that a is in none of its stable extensions. What we would like to do is to adapt the attack relation of G_2 in a way that is minimal and that guarantees that a does not occur in any of its extensions. We view minimality as minimality of the number of modifications of the attack relation; in contrast, the current extensions may be modified in a non-minimal way. The papers [22, 20] take a different perspective: the minimization of the changes on the attack relation is considered to be secondary, whereas the changes on the current extensions should be minimal.

More generally, we are interested in enforcing some constraint α on an AF $\mathsf{G} = (\mathbb{A}, \mathsf{R})$, where we understand that α is a formula in the language $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$, and that enforcing α consists in minimally modifying the attack relation R of G to R' in a way such that α holds in all of the extensions of $\mathsf{G}' = (\mathbb{A}, \mathsf{R}')$.

Another perspective is however possible, where enforcing consists in minimally modifying R such that α holds not in all but only in *some* extension of the resulting $\mathsf{G}' = (\mathbb{A}, \mathsf{R}')$. So 'enforcement' can be understood in two different ways, leading to two different definitions.

In both cases, there is a key difference with standard revision and update operations: in classical belief change, output is limited to one belief set, while here, several G' may be produced by the enforcement operations.

To sum it up, we identify two kinds of modification operations $\diamond^{\mathsf{Cred},\sigma}$ and $\diamond^{\mathsf{Skep},\sigma}$. Both map an AF and a boolean formula of $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ to a set of AFs. The set $\mathsf{G} \diamond^{\mathsf{Cred},\sigma} \alpha$ is the set of *credulous* enforcements of α and $\mathsf{G} \diamond^{\mathsf{Skep},\sigma} \alpha$ is the set of *skeptical* enforcements of α.[4]

5.2 Which Postulates?

In line with the classical definitions of belief change operations, we now define some postulates that 'reasonable' operations $\diamond$ should satisfy. They are mainly inspired by the work of Coste-Marquis et al. [22].

[4] Actually it is possible to refine these operations further. We might e.g. define $\mathsf{G} \diamond^{\mathsf{Cred},\sigma,\exists} \alpha$ where α is credulously enforced in *some* $\mathsf{G}' \in \mathsf{G} \diamond^{\mathsf{Cred},\sigma,\exists} \alpha$ on the one hand, and $\mathsf{G} \diamond^{\mathsf{Cred},\sigma,\forall} \alpha$ where α is credulously enforced in *every* $\mathsf{G}' \in \mathsf{G} \diamond^{\mathsf{Cred},\sigma,\exists} \alpha$ on the other.

Definition 2 Let σ be any semantics. Let $\diamond$ be an operation mapping an AF over $\mathbb{A}$ and an $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ formula to a set of AFs over $\mathbb{A}$.

The operation $\diamond^{\sigma}$ is a *credulous enforcement* operation iff it satisfies the following three postulates:

E1.C $\theta(\mathsf{G}') \wedge \theta(\sigma, \mathbb{A}) \wedge \alpha$ is satisfiable for every $\mathsf{G}' \in \mathsf{G} \diamond^{\sigma} \alpha$.[5]
E2.C If $\theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A}) \wedge \alpha$ is satisfiable then $\mathsf{G} \diamond^{\sigma} \alpha = \{\mathsf{G}\}$.
E3 If $\models \alpha_1 \leftrightarrow \alpha_2$ then for every $\mathsf{G}_1 \in \mathsf{G} \diamond^{\sigma} \alpha_1$ there exists $\mathsf{G}_2 \in \mathsf{G} \diamond^{\sigma} \alpha_2$ such that $\mathsf{G}_1 = \mathsf{G}_2$.

The operation $\diamond$ is a *skeptical enforcement* operation iff it satisfies postulate **E3** plus the following:

E1.S $\models \theta(\mathsf{G}') \wedge \theta(\sigma, \mathbb{A}) \rightarrow \alpha$ for every $\mathsf{G}' \in \mathsf{G} \diamond^{\sigma} \alpha$.
E2.S If $\models \theta(\mathsf{G}) \wedge \theta(\sigma, \mathbb{A}) \rightarrow \alpha$ then $\mathsf{G} \diamond^{\sigma} \alpha = \{\mathsf{G}\}$.

The postulates **E1.C** and **E1.S** say that success in required for credulous and skeptical enforcement. **E2** represents a minimal change principle: it states that if α already holds then G is unchanged. Postulate **E3** is the postulate of syntax independence: enforcement should be based on the logical content of a goal and not on its syntax.

Additional postulates may be formulated; see [14, 22] for more details. A key difference is that we do not consider postulates based on the expansion operation. The main reason is that this operation is actually useless: first, if an attack has to be changed then expansion cannot be used because, as we have seen above, $\theta(\mathsf{G})$ is complete for $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$. Now consider that an argument has to be enforced in a credulous way. We face two cases: (i) either the argument is already credulously acceptable and there is no reason to change the AF, or (ii) the argument is not credulously acceptable. Then as all possible extensions are considered, some attacks must be changed so that new extensions can be constructed (and α will then hold). The same reasoning can be made for skeptical acceptance.

5.3 Which Belief Change Operation?

In the literature two different families of belief change operations were studied: AGM revision operations [1] and KM update operations [41]. In a nutshell, the former models the incorporation of a new piece of information about a static world, while the latter models a dynamic world.

Neither AGM theory [1] nor KM theory [41] provide a single, concrete belief change operation: they rather constrain the set of 'reasonable' belief change operations by means of a set of postulates. Another way of saying this is: both AGM and KM rely on underlying orderings in order to construct $\diamond$. In the case of AFs it is not immediately clear where such information comes from. (One may think of

[5] This is the same as: there exists a σ-extension E of G' such that $\models \big(\theta(\mathsf{G}) \wedge \theta(E)\big) \rightarrow \alpha$.

preference relations between arguments, it is however not clear whether this matches intuitions.)

Several concrete belief change operations satisfying the AGM or KM postulates have been defined in the literature. Among the most prominent are Winslett's 'possible models approach' (PMA) [49, 50], Forbus's update operation [33], Winslett's standard semantics (WSS) [51], and Dalal's revision operation [24]; see [40, 42] for an overview. According to Katsuno and Mendelzon's distinction between update and revision operations [41], the first three are update operations, usually written $\diamond$, while Dalal's is a revision operation, usually written $*$.

Actually the models of Forbus's update and Dalal's revision coincide in the case of *complete* belief bases, and the same is the case for Winslett's update and Satoh's revision. Complete bases contain either p or $\neg p$, for every propositional variable p. This observation applies to the modification of AFs: the theory $\theta(\mathtt{G})$ is complete for $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$, and this is what matters: the variables of $\mathsf{Lang}(\mathtt{IN}^{\mathbb{A}})$ only encode particular extensions. Therefore the question whether AF modification is an update or a revision does not really play a role here.

We dedicate the next sections to two problems: modifying the framework and modifying an extension.

6 Enforcing a Constraint on All/Some Extensions

The aim of this section is to provide a formal definition of two extension change operations. Both are based on Forbus's update operation [33], where minimal change involves counting how many variables change their truth value. Our operations satisfy the enforcement postulates that we have defined in Section 5.2.

We start by observing that the update of AFs has some specificities: first, we are going to modify only the attack variables while leaving the accept variables unchanged; second, the target formula is not going to be a boolean formula, but a formula saying that α will be the case *after building extensions*. When we construct the extensions we are going to minimize the modifications of $\mathtt{ATT}^{\mathbb{A}}(\alpha)$, while those of the set $\mathtt{IN}^{\mathbb{A}}(\alpha)$ will not be minimized: given the truth values of the attack variables, the truth values of the accept variables are determined by the semantics (or rather, the possible combinations of accept variables, because there may be several extensions).

6.1 The Hamming Distance

The *Hamming distance* between two valuations v and v' w.r.t. a set of propositional variables P is the cardinality of the set of all those variables from P whose truth value differs:

$$\mathcal{H}^P(v, v') = \mathsf{card}(P \cap (v \dot{-} v')).$$

For example, consider $v_1 = \{\mathsf{r}_{a,b}, \mathsf{r}_{b,a}, \mathsf{in}_b\}$ and $v_2 = \{\mathsf{r}_{a,b}, \mathsf{in}_a\}$. Then $v_1 \dot{-} v_2 = \{\mathsf{r}_{b,a}, \mathsf{in}_a, \mathsf{in}_b\}$ and therefore $\mathcal{H}^{\mathtt{ATT}^{\mathbb{A}}}(v_1, v_2) = 1$ and $\mathcal{H}^{\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}}}(v_1, v_2) = 3$.

When P equals $\mathbb{P}$ then $\mathcal{H}^P(v, v')$ is nothing but the Hamming distance between v and v'. Forbus's update operation is based on minimization of that distance: first, the Forbus update of a valuation v by α is the set of those α-models whose Hamming distance to v is minimal; second, the Forbus update of a belief base β by α collects the Forbus updates of all models of β by α:

$$v \diamond^{\mathsf{forbus}} \alpha = \left\{v' \ : \ v' \models \alpha \text{ and there is no } v'' \text{ such that } v'' \models \alpha \text{ and } \mathcal{H}^{\mathbb{P}}(v, v'') < \mathcal{H}^{\mathbb{P}}(v, v')\right\}$$

$$\beta \diamond^{\mathsf{forbus}} \alpha = \bigcup_{v \models \beta} \left(v \diamond^{\mathsf{forbus}}_{P} \alpha\right)$$

6.2 Argumentation Framework Update

Just as the original Forbus update operation, our two update operations are defined in two steps: first the update of a valuation in terms of the Hamming distance and then the update of an AF as the union of the updates of each of its models. But first we need a definition: given a valuation v, the AF associated to v is defined as

$$\mathsf{G}(v) = \{(a, b) \ : \ \mathsf{r}_{a,b} \in v\}.$$

Then the *skeptical Forbus update* by a $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ formula α under semantics σ is defined as follows:

$$v \diamond^{\sigma}_{\mathsf{skep}} \alpha = \Big\{v' \ : \ \text{every } \sigma\text{-extension of } \mathsf{G}(v') \text{ satisfies } \alpha \text{ and}$$
$$\text{there is no } v'' \text{ such that } \mathcal{H}^{\mathtt{ATT}^{\mathbb{A}}}(v, v'') < \mathcal{H}^{\mathtt{ATT}^{\mathbb{A}}}(v, v')$$
$$\text{and every } \sigma\text{-extension of } \mathsf{G}(v'') \text{ satisfies } \alpha \ \Big\}.$$

$$\mathsf{G} \diamond^{\sigma}_{\mathsf{skep}} \alpha = \bigcup_{v \in ||\theta(\mathsf{G})||} v \diamond^{\sigma}_{\mathsf{skep}} \alpha$$

So $v \diamond^{\sigma}_{\mathsf{skep}} \alpha$ is the set of valuations whose σ-extensions all satisfy α and whose Hamming distance to v is minimal w.r.t. the variables in $\mathtt{ATT}^{\mathbb{A}}$.[6] This can be viewed as a circumscription policy with fixed and varying propositional variables [43].

Symmetrically we define the *credulous Forbus update* as:

[6] The expression 'the set of valuations whose σ-extensions all satisfy α' is a bit imprecise here; more precisely, it is the set of valuations v' such that $\models \big(\theta((\mathbb{A}, v' \cap (\mathbb{A} \times \mathbb{A}))) \wedge \theta(\sigma, \mathbb{A})\big) \rightarrow \alpha$.

$$v \diamond^{\sigma}_{\mathsf{cred}} \alpha = \Big\{ v' \ : \ \text{some } \sigma\text{-extension of } \mathsf{G}(v') \text{ satisfies } \alpha \text{ and}$$
$$\text{there is no } v'' \text{ such that } \mathcal{H}^{\mathtt{ATT}^{\mathbb{A}}}(v, v'') < \mathcal{H}^{\mathtt{ATT}^{\mathbb{A}}}(v, v')$$
$$\text{and some } \sigma\text{-extension of } \mathsf{G}(v'') \text{ satisfies } \alpha \Big\}.$$
$$\mathsf{G} \diamond^{\sigma}_{\mathsf{cred}} \alpha = \bigcup_{v \in ||\theta(\mathsf{G})||} v \diamond^{\sigma}_{\mathsf{cred}} \alpha$$

Note that these operations resemble the Forbus update operation but cannot be reduced to them. The reason is that the input would have to be a *counterfactual statement* of the form "*if* the current valuation is transformed into a σ-extension by modifying the acceptance variables then α results".

Both operations coincide for updates by attack variables and their negation.

Proposition 3 *Let α be an attack literal, i.e., a propositional variable of the form $\mathsf{r}_{a,b}$ or its negation. Then $\mathsf{G} \diamond^{\sigma}_{\mathsf{skep}} \alpha = \mathsf{G} \diamond^{\sigma}_{\mathsf{cred}} \alpha$.*

The identity fails to hold for arbitrary formulas.

7 Expressing Extension Modification in DL-PA

The aim of this section is to define modification programs in DL-PA implementing the two operations of modification of AFs that we have defined. The exposition follows [38].

7.1 Modifications of the Attack Relation Only

To warm up, observe that addition and removal of an attack edge between two arguments $a, b \in \mathbb{A}$ can be directly implemented by the atomic DL-PA programs: the addition of (a, b) to R is obtained by executing $\mathsf{r}_{a,b} \leftarrow \top$, and the removal of (a, b) from R is obtained by executing $\neg \mathsf{r}_{a,b} \leftarrow \bot$. Indeed, we have:

$$\mathsf{G} \diamond \ \mathsf{r}_{a,b} = \langle \mathsf{r}_{a,b} \leftarrow \top^{-} \rangle \theta(\mathsf{G})$$
$$\mathsf{G} \diamond \neg \mathsf{r}_{a,b} = \langle \mathsf{r}_{a,b} \leftarrow \bot^{-} \rangle \theta(\mathsf{G})$$

where $\diamond$ is any of the above belief change operations $\diamond^{\sigma}_{\mathsf{skep}}$ and $\diamond^{\sigma}_{\mathsf{cred}}$. Once $\theta(\mathsf{G})$ has been updated by some $\mathsf{r}_{a,b}$ or $\neg \mathsf{r}_{a,b}$, the extensions of the resulting framework can be obtained by conjoining the result with $\theta(\sigma, \mathbb{A})$.

This can be generalised to input formulas α that are conjunctions of literals in the language $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$. More generally, when α is a formula in the language $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$ then it can be shown that both the skeptical and the credulous update of G by α are nothing but the classical Forbus update.

Proposition 4 *Let α be a formula in the language* $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}})$. *Then*

$$\mathsf{G} \diamond^{\sigma}_{\mathsf{skep}} \alpha = \theta(\mathsf{G}) \diamond^{\mathsf{forbus}} \alpha.$$

7.2 The Hamming Distance Predicate in DL-PA

Let us define the DL-PA formula $\mathsf{H}(\alpha, P, \geq m)$, where α is a $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ formula, P is a set of propositional variables, and $m \geq 0$ is an integer:

$$\mathsf{H}(\alpha, P, \geq m) = \begin{cases} \top & \text{if } m = 0 \\ \neg\langle (\mathsf{flipOne}(P))^{\leq m-1} \rangle \alpha & \text{if } m \geq 1 \end{cases}$$

We call $\mathsf{H}(\alpha, P, \geq m)$ the *Hamming distance predicate w.r.t. the set of variables P*: it is true at a valuation v exactly when the α-models v' that are closest to v in the sense of the Hamming distance differ in at least m variables from v, where the computation of the distance only considers variables from P while the other variables in $\mathbb{P} \setminus P$ cannot be modified.

Proposition 5 *Let v a valuation, α a boolean formula, P some set of propositional variables, and $m \geq 0$. Then*

1. *$v \in ||\mathsf{H}(\alpha, P, \geq m)||$ iff the α-models that are closest to v w.r.t. P have Hamming distance at least m, i.e., iff $\mathcal{H}(v, v') \geq m$ for every $v' \in ||A||$ such that $v \dot{-} v' \subseteq P$.*
2. *$(v, v') \in ||\mathsf{H}(\alpha, P, \geq m)?; (\mathsf{flipOne}(P))^{m}; \alpha?||$ iff v' is one of the α-models that is closest to v w.r.t. P, i.e., iff $v' \in ||\alpha||$ and $\mathcal{H}(v, v') = m$ for every $v' \in ||A||$ such that $v \dot{-} v' \subseteq P$.*

Proof For Item 1, things are clear for $m = 0$, and we only consider the case $m > 1$.

From the left to the right, suppose v is a model of $\mathsf{H}(\alpha, P, \geq m)$, i.e., of the formula $\neg\langle (\mathsf{flipOne}(P))^{\leq m-1} \rangle \alpha$. Then there is no α-model v' such that $(v, v') \in (||\mathsf{flipOne}(P)||)^{k}$, for some $k < m$. So by Lemma 1, for every valuation $v' \in ||A||$ such that $v \dot{-} v' \subseteq P$ we must have $\mathcal{H}(v, v') \geq m$.

From the right to the left, suppose for every valuation $v' \in ||A||$ such that $v \dot{-} v' \subseteq P$ we have $\mathcal{H}(v, v') \geq m$. Then by Lemma 1, there cannot be a v' such that $(v, v') \in ||\mathsf{flipOne}(P)||^{k}$ for some $k < m$ and $v' \in ||\alpha||$. Therefore v must be a model of the formula $\neg\langle (\mathsf{flipOne}(P))^{\leq m-1} \rangle \alpha$.

Item 2 then follows from Item 1 and Lemma 1. □

It follows from the first item of Proposition 5 that when P equals $\mathbb{P}(\alpha)$ then $v \in ||\mathsf{H}(\alpha, \mathbb{P}(\alpha), \geq m)||$ iff the α-models that are closest to v have Hamming distance at least m, i.e., iff $\mathcal{H}(v, v') \geq m$ for every $v' \in ||A||$. This is used in Forbus's udpate operation.

The following DL-PA program performs Forbus's update operation:

$$\mathsf{forbus}(\alpha) = \bigsqcup_{m \leq \mathsf{card}(\mathbb{P}(\alpha))} \Big(\mathsf{H}(\alpha, \mathbb{P}(\alpha), \geq m)?; \big(\mathsf{flipOne}(\mathbb{P}(\alpha))\big)^m\Big); \alpha?$$

The program nondeterministically chooses an integer m, checks if the Hamming distance to α-models is at least m and flips m of the variables of α. Finally, the test $\alpha?$ only succeeds for α-models.

Proposition 6 *The formula γ is true after the Forbus update of β by α if and only if $\beta \rightarrow [\mathsf{forbus}(\alpha)]\gamma$ is* DL-PA *valid.*

7.3 Argumentation Framework Update

The update of AFs has some specificities: first, we only modify the attack variables while leaving the accept variables unchanged; second, the target formula is not a boolean formula but a formula saying that α will be the case *after building extensions*.

The programs $\mathsf{credEnf}^\sigma(\alpha)$ and $\mathsf{skepEnf}^\sigma(\alpha)$ minimally modify a valuation w.r.t. some semantics σ such that the boolean formula $\alpha \in \mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ becomes true in some/all σ-extensions.

$$\mathsf{credEnf}^\sigma(\alpha) = \bigsqcup_{m \leq \mathsf{card}(\mathtt{ATT}^{\mathbb{A}})} \Big(\mathsf{H}\big(\langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\alpha, \mathtt{ATT}^{\mathbb{A}}, \geq m\big)?; \big(\mathsf{flipOne}(\mathtt{ATT}^{\mathbb{A}})\big)^m\Big); \langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\alpha?$$

$$\mathsf{skepEnf}^\sigma(\alpha) = \bigsqcup_{m \leq \mathsf{card}(\mathtt{ATT}^{\mathbb{A}})} \Big(\mathsf{H}\big([\mathsf{mkExt}(\sigma, \mathbb{A})]\alpha, \mathtt{ATT}^{\mathbb{A}}, \geq m\big)?; \big(\mathsf{flipOne}(\mathtt{ATT}^{\mathbb{A}})\big)^m\Big); [\mathsf{mkExt}(\sigma, \mathbb{A})]\alpha?$$

Both programs (1) nondeterministically choose a value m for the Hamming distance to $\langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\alpha$, i.e., to valuations of attack variables having extensions satisfying α, (2) check that m satisfies $\mathsf{H}\big(\langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\alpha, \mathtt{ATT}^{\mathbb{A}}, \geq m\big)$, (3) flip m attack variables, and then (4) check that either some extension satisfies α (credulous case), or all extensions satisfy α (skeptical case).

The length of these two programs is polynomial in the cardinality of $\mathbb{A}$.[7]

The next theorem provides an embedding of both skeptical and credulous AF update into DL-PA. It is the main result of our paper.

Theorem 4 *Let* G *be an AF. Let $\alpha \in \mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ be a boolean formula. Then*

$$\mathsf{G} \diamond^\sigma_{\mathsf{cred}} \alpha = \Big\|\big\langle(\mathsf{credEnf}^\sigma(\alpha))^-\big\rangle\theta(\mathsf{G})\Big\|$$
$$\mathsf{G} \diamond^\sigma_{\mathsf{skep}} \alpha = \Big\|\big\langle(\mathsf{skepEnf}^\sigma(\alpha))^-\big\rangle\theta(\mathsf{G})\Big\|$$

[7] The cardinality of the set $\mathtt{ATT}^{\mathbb{A}}$ is quadratic in that of $\mathbb{A}$, and the length of $(\mathsf{flipOne}(\mathtt{ATT}^{\mathbb{A}}))^m$ is quadratic in that of $\mathbb{A}$.

For $\alpha, \gamma \in \mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$, to check whether γ is true in all extensions of G modified by α can then be done by checking whether the DL-PA formula $\theta(\mathsf{G}) \rightarrow [\mathsf{skepEnf}^{\sigma}(\alpha)]\gamma$ is valid. We in particular have the following result, which says that every possible execution of the two enforcement programs will succeed:

Proposition 7 *Let* G *be an AF. Let* $\alpha \in \mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ *be a boolean formula. Then*

$$\models \big[\mathsf{credEnf}^{\sigma}(\alpha)\big]\big\langle \mathsf{mkExt}(\sigma, \mathbb{A})\big\rangle\alpha$$
$$\models \big[\mathsf{skepEnf}^{\sigma}(\alpha)\big]\big[\mathsf{mkExt}(\sigma, \mathbb{A})\big]\alpha$$

The second key property is that an AF remains unchanged if the goal already holds.

Proposition 8 *For every goal* α *that is credulously justified, the credulous update program does not change anything:*

$$\models (\theta(\mathsf{G}) \wedge \langle \mathsf{mkExt}(\sigma, \mathbb{A})\rangle\alpha \wedge \gamma) \rightarrow [\mathsf{credEnf}^{\sigma}(\alpha)]\gamma.$$

For every goal α *that is skeptically justified, the skeptical update program does not change anything:*

$$\models (\theta(\mathsf{G}) \wedge [\mathsf{mkExt}(\sigma, \mathbb{A})]\alpha \wedge \gamma) \rightarrow [\mathsf{skepEnf}^{\sigma}(\alpha)]\gamma.$$

The modified AFs can be extracted from the formulas $\langle \mathsf{credEnf}^{\sigma}(\alpha)^{-}\rangle\theta(\mathsf{G})$ and $\langle \mathsf{skepEnf}^{\sigma}(\alpha)^{-}\rangle\theta(\mathsf{G})$ representing it in DL-PA (or rather, their reduction) by forgetting the accept variables, as proposed in [22]. This operation can be implemented in our framework by the program $\mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}})$.

Definition 3 Let σ be either the stable, admissible, or complete semantics. Let enf be either the $\mathsf{skepEnf}^{\sigma}$ or the $\mathsf{credEnf}^{\sigma}$ program. Let $\diamond^{\sigma,\mathsf{enf}}$ be an operation mapping an AF and an $\mathsf{Lang}(\mathtt{ATT}^{\mathbb{A}} \cup \mathtt{IN}^{\mathbb{A}})$ formula to a set of AFs. The update of G by α under σ and enf is

$$\mathsf{G} \diamond^{\sigma,\mathsf{enf}} \alpha = \big\{(\mathbb{A}, R_v) \ : \ v \in ||\langle(\mathsf{enf}(\alpha))^{-}\rangle\theta(\mathsf{G})||\big\}$$

where R_v is the attack relation extracted from v, defined as $R_v = \{(a, b) \ : \ \mathsf{r}_{a,b} \in v\}$.

The two preceding propositions guarantee that our enforcement operations satisfy the postulates.

Theorem 5 *Operation* $\diamond^{\sigma,\mathsf{credEnf}^{\sigma}}$ *satisfies* ***E1.C*** *and* ***E2.C****. Operation* $\diamond^{\sigma,\mathsf{skepEnf}^{\sigma}}$ *satisfies* ***E1.S*** *and* ***E2.S****. Both operations* $\diamond^{\sigma,\mathsf{credEnf}^{\sigma}}$ *and* $\diamond^{\sigma,\mathsf{skepEnf}^{\sigma}}$ *satisfy* ***E3****.*

This result is actually not a surprise, given that our tool for enforcement is a variant of Forbus's update operation.

Example

Let us take up the AF G_2 of Figure 1. Remember that $\mathsf{G}_2 = (\mathbb{A}_1, \mathsf{R}_2)$, with $\mathbb{A}_1 = \{a, b\}$ and $\mathsf{R}_2 = \{(a, b), (b, a)\}$ and that $\theta(\mathsf{G}_2) = \mathsf{r}_{a,b} \wedge \mathsf{r}_{b,a}$. Let us consider the stable semantics and suppose we want skeptical enforcement of a, i.e., we want to enforce that a is always acceptable. We disregard self-attacks for the sake of simplicity. The nondeterministic part $\bigsqcup_{m \leq \mathsf{card}(\mathtt{ATT}^{\mathbb{A}})}(\ldots)$ of the program $\mathsf{skepEnf}(\mathsf{in}_a)$ changes one variable from $\theta(\mathsf{G}_2)$, either $\mathsf{r}_{a,b}$ or $\mathsf{r}_{b,a}$. This corresponds to two candidate extensions: one where a only attacks b and one where b only attacks a. Only the former case gives valuations where a is always acceptable. Hence:

$$\mathsf{G} \diamond^{\mathsf{stable},\mathsf{skepEnf}^{\mathsf{stable}}} \mathsf{in}_a = \big\{(\mathbb{A}_1, \{(a, b)\})\big\}.$$

8 Going Further

We have illustrated how DL-PA offers a fruitful framework for representing AFs and reasoning about them. We now sketch several ways of extending our account.

8.1 Checking (odd/even) cycles

One may also wish to identify and modify global properties of argumentation frameworks, such as the existence of cycles, or the existence of odd or even cycles. In propositional logic the existence of a cycle in a given AF can be characterised by means of the propositional formula

$$\mathsf{ExistsCycle}_{\mathbb{A}} = \bigvee_{n \leq \mathsf{card}(\mathbb{A})} \bigvee_{a_1,\ldots,a_n \in \mathbb{A}} (\mathsf{r}_{a_1,a_2} \wedge \ldots \wedge \mathsf{r}_{a_{n-1},a_n} \wedge \mathsf{r}_{a_n,a_1})$$

The length of that formula is however exponential in the number of arguments. Fortunately, the existence of cycles can be characterised in DL-PA by a more succinct formula: it tests if the execution of a program iterating one transitive closure step can lead to a self-attack. Such a closure step is performed by the following program:

$$\mathsf{step} = \bigcup_{a,b,c \in \mathbb{A}} (\mathsf{r}_{a,b} \wedge \mathsf{r}_{b,c}?; \mathsf{r}_{a,c} \leftarrow \top).$$

Using that we characterise the existence of a self-attack in an AF by the formula $\mathsf{Loop} = \bigvee_{a \in \mathbb{A}} \mathsf{r}_{a,a}$, the formula $\mathsf{ExistsCycle}_{\mathbb{A}} = \langle \mathsf{step}^* \rangle \mathsf{Loop}$ then characterises the existence of a cycle. The length of that formula is polynomial in the number of arguments.

Actually the Kleene star can be replaced by a bounded iteration up to $\mathsf{card}(\mathbb{A}) - 1$. This will be useful to characterise even and odd cycles. Observe that when there is an odd cycle in an AF then a self-attack can be produced by an even number of closure steps; symmetrically, an even cycle can be achieved by an odd number of closure steps. Based on that observation we can check the existence of odd and even cycles by means of the following DL-PA formulas:

$$\mathsf{existsEvenCycle} = \left\langle \bigcup_{0 \leq n \leq \lceil \frac{\mathsf{card}(\mathbb{A})}{2} \rceil} \left(\neg\langle \mathsf{step}^{2n}\rangle \mathsf{Loop}?; \mathsf{step}^{2n+1}\right) \right\rangle \mathsf{Loop}$$

$$\mathsf{existsOddCycle} = \mathsf{Loop} \vee \left\langle \bigcup_{1 \leq n \leq \lceil \frac{\mathsf{card}(\mathbb{A})}{2} \rceil} \left(\neg\langle \mathsf{step}^{2n-1}\rangle \mathsf{Loop}?; \mathsf{step}^{2n}\right) \right\rangle \mathsf{Loop}$$

The formula characterising even cycles checks whether there is an integer $n \geq 0$ such that $2n+1$ transitive closure steps may lead to a self-attack. The test $\neg\langle \mathsf{step}^{2n}\rangle \mathsf{Loop}?$ makes sure that this is the smallest such number. The formula characterising odd cycles similarly checks whether there is an $n \geq 0$ such that $2n$ transitive closure steps may lead to a self-attack. The length of these two formulas is still polynomial in the number of arguments.

8.2 A Prioritised Version

We may adapt our modification operation in order to accommodate a prioritised version that was proposed in [23]. While up to now we only minimised modifications of $\mathtt{ATT}^{\mathbb{A}}$, this paper proposes to replace this policy by a "first minimise $\mathtt{IN}^{\mathbb{A}}$, then $\mathtt{ATT}^{\mathbb{A}}$" policy: first we minimally change $\mathtt{IN}^{\mathbb{A}}$ variables to make $\langle \mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}})\rangle\gamma$ true, ending up in those minimally $\mathtt{ATT}^{\mathbb{A}}$-distant states from which an extension satisfying the goal can be constructed; then we minimally change the $\mathtt{ATT}^{\mathbb{A}}$ variables in order to make the goal γ true.

We capture this in DL-PA by two Forbus updates in sequence:

$$\left((\mathbb{A}, \mathtt{ATT}^{\mathbb{A}}) \diamond^{\mathsf{forbus}}_{\mathtt{IN}^{\mathbb{A}}} \left(\langle \mathsf{flipSome}(\mathtt{IN}^{\mathbb{A}})\rangle\gamma\right)\right) \diamond^{\mathsf{forbus}}_{\mathtt{ATT}^{\mathbb{A}}} \gamma.$$

We observe that this may lead to multiple extensions and that it might be more appropriate to rather apply Dalal revision.

9 Conclusion

The main result of this paper is the encoding of AFs and their dynamics in DL-PA, extending and generalising [28, 29]. More precisely, our contribution is threefold.

First, as long as argument acceptability can be expressed in propositional logic, finding acceptable arguments and enforcing acceptability can be done in DL-PA. The DL-PA framework moreover enables formal verification of the correctness of an algorithm (which however for the time being has to await implemented DL-PA solvers). Other logical frameworks allow capturing and computing argument acceptability (see [19] for an overview), but few of them allow capturing and computing acceptability change as well.

Second, as DL-PA formulas can be rewritten as propositional logic formulas, the result of the modification of an AF is described by a propositional formula from the models of which one may retrieve the modified AFs. Our proposal is hence more operational than those of other approaches because we use a formal logic encompassing the representation of change operations. Moreover, we consider not only credulous acceptability changes, as most of the current approaches do [7], but skeptical acceptability change as well.

Third, our framework takes advantage of the complexity results for DL-PA: both model checking and satisfiability checking are in PSPACE. A closer look at the formulas expressing the modifications shows that the alternation of quantifications is bounded, which typically leads to complexity bounds at the second level of the polynomial hierarchy.

The proposed DL-PA encodings may be related to QBF encodings for argumentation [2]. As QBF has the same complexity as star-free DL-PA, all we do in DL-PA must be polynomially encodable into QBF. However, the availability of assignment programs makes a difference: update programs such as $\mathsf{forbus}(\alpha)$ and extension construction programs such as $\mathsf{mkExt}(\sigma, \mathbb{A})$ capture things in a more general, flexible, and natural way than a QBF encoding.

To conclude, the richness of our framework makes it expandable to other kinds of changes, other update semantics, and other argumentation semantics beyond those that are detailed in the present paper. We plan to investigate this research avenue in future work.

Acknowledgements This work benefited from the support of the AMANDE project (ANR-13-BS02-0004) of the French National Research Agency (ANR). We would like to thank Gabriele Kern-Isberner for triggering our work when DL-PA was presented at the Dagstuhl seminar "Belief Change and Argumentation in Multi-Agent Scenarios" in June 2013. The paper benefited from several comments by Christoph Beierle after its presentation at KR 2014. After KR 2014, the ideas in the paper were presented in a workshop on argumentation in Lens in September 2015 and at the Cardiff Argumentation Forum in July 2016.

References

1. Alchourrón, C., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. J. of Symbolic Logic **50**, 510–530 (1985)
2. Arieli, O., Caminada, M.W.: A QBF-based formalization of abstract argumentation semantics. Journal of Applied Logic **11**(2), 229–252 (2013)

3. Balbiani, P., Herzig, A., Schwarzentruber, F., Troquard, N.: DL-PA and DCL-PC: model checking and satisfiability problem are indeed in PSPACE. CoRR **abs/1411.7825** (2014). URL http://arxiv.org/abs/1411.7825
4. Balbiani, P., Herzig, A., Troquard, N.: Dynamic logic of propositional assignments: a well-behaved variant of PDL. In: Logic in Computer Science (LICS). IEEE (2013)
5. Baroni, P., Giacomin, M.: On principle-based evaluation of extension-based argumentation semantics. Artif. Intell. **171**(10-15), 675–700 (2007). DOI 10.1016/j.artint.2007.04.004. URL http://dx.doi.org/10.1016/j.artint.2007.04.004
6. Baroni, P., Giacomin, M.: Semantics of abstract argument systems. In: G. Simari, I. Rahwan (eds.) Argumentation in Artificial Intelligence, pp. 25–44. Springer US (2009). DOI 10.1007/978-0-387-98197-0_2. URL http://dx.doi.org/10.1007/978-0-387-98197-0_2
7. Baumann, R.: What does it take to enforce an argument? minimal change in abstract argumentation. In: L.D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, P.J.F. Lucas (eds.) ECAI 2012, *Frontiers in Artificial Intelligence and Applications*, vol. 242, pp. 127–132. IOS Press (2012)
8. Baumann, R., Brewka, G.: Expanding argumentation frameworks: Enforcing and monotonicity results. COMMA **216**, 75–86 (2010)
9. Baumann, R., Brewka, G.: Spectra in abstract argumentation: An analysis of minimal change. In: P. Cabalar, T.C. Son (eds.) Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings, *Lecture Notes in Computer Science*, vol. 8148, pp. 174–186. Springer (2013). DOI 10.1007/978-3-642-40564-8_18. URL http://dx.doi.org/10.1007/978-3-642-40564-8_18
10. Baumann, R., Brewka, G.: AGM meets abstract argumentation: Expansion and revision for dung frameworks. In: Yang and Wooldridge [52], pp. 2734–2740. URL http://ijcai.org/Abstract/15/387
11. Belardinelli, F., Grossi, D., Maudet, N.: Formal analysis of dialogues on infinite argumentation frameworks. In: Yang and Wooldridge [52], pp. 861–867. URL http://ijcai.org/Abstract/15/126
12. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: 10th Int. Workshop on Non-Monotonic Reasoning (NMR'2004), pp. 59–64 (2004). URL http://www.pims.math.ca/science/2004/NMR/papers/paper18.pdf
13. Bisquert, P.: Étude du changement en argumentation. De la théorie à la pratique. Phd thesis, Univ. Toulouse, Toulouse, France (2013). URL http://www.irit.fr/publis/ADRIA/ThesePierreBisquert.pdf
14. Bisquert, P., Cayrol, C., Bannay, F., Lagasquie-Schiex, M.C.: Enforcement in Argumentation is a kind of Update. In: W. Liu, V. Subrahmanian, J. Wijsen (eds.) International Conference on Scalable Uncertainty Management (SUM), Washington DC, USA, no. 8078 in LNAI, pp. 30–43. Springer Verlag (2013)
15. Bisquert, P., Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Change in argumentation systems: Exploring the interest of removing an argument. In: S. Benferhat, J. Grant (eds.) Scalable Uncertainty Management - 5th International Conference, SUM 2011, Dayton, OH, USA, October 10-13, 2011. Proceedings, *Lecture Notes in Computer Science*, vol. 6929, pp. 275–288. Springer (2011). DOI 10.1007/978-3-642-23963-2_22. URL http://dx.doi.org/10.1007/978-3-642-23963-2_22
16. Bisquert, P., Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.: Goal-driven changes in argumentation: A theoretical framework and a tool. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013, pp. 610–617. IEEE Computer Society (2013). DOI 10.1109/ICTAI.2013.96. URL http://dx.doi.org/10.1109/ICTAI.2013.96
17. Booth, R., Kaci, S., Rienstra, T., van der Torre, L.: A logical theory about dynamics in abstract argumentation. In: W. Liu, V.S. Subrahmanian, J. Wijsen (eds.) SUM, *Lecture Notes in Computer Science*, vol. 8078, pp. 148–161. Springer (2013)
18. Cayrol, C., Bannay, F., Lagasquie-Schiex, M.C.: Change in Abstract Argumentation Frameworks: Adding an Argument. Journal of Artificial Intelligence Research **38**, 49–84 (2010). URL http://www.jair.org/papers/paper2965.html

19. Charwat, G., Dvorák, W., Gaggl, S.A., Wallner, J.P., Woltran, S.: Methods for solving reasoning problems in abstract argumentation - A survey. Artif. Intell. **220**, 28–63 (2015). DOI 10.1016/j.artint.2014.11.008. URL http://dx.doi.org/10.1016/j.artint.2014.11.008
20. Coste-Marquis, S., Konieczny, S., Mailly, J., Marquis, P.: On the revision of argumentation systems: Minimal change of arguments statuses. In: C. Baral, G. De Giacomo, T. Eiter (eds.) International Conference on Principles of Knowledge Representation and Reasoning (KR). AAAI Press (2014)
21. Coste-Marquis, S., Konieczny, S., Mailly, J., Marquis, P.: Extension enforcement in abstract argumentation as an optimization problem. In: Yang and Wooldridge [52], pp. 2876–2882. URL http://ijcai.org/Abstract/15/407
22. Coste-Marquis, S., Konieczny, S., Mailly, J.G., Marquis, P.: On the revision of argumentation systems: Minimal change of arguments status. In: TAFA'13 (2013)
23. Coste-Marquis, S., Konieczny, S., Mailly, J.G., Marquis, P.: A translation-based approach for revision of argumentation frameworks. In: JELIA, pp. 397–411. Springer (2014)
24. Dalal, M.: Investigations into a theory of knowledge base revision: preliminary report. In: Proc. 7th Conf. on Artificial Intelligence (AAAI'88), pp. 475–479 (1988)
25. Delobelle, J., Haret, A., Konieczny, S., Mailly, J., Rossit, J., Woltran, S.: Merging of abstract argumentation frameworks. In: C. Baral, J.P. Delgrande, F. Wolter (eds.) Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016., pp. 33–42. AAAI Press (2016). URL http://www.aaai.org/ocs/index.php/KR/KR16/paper/view/12872
26. Diller, M., Haret, A., Linsbichler, T., Rümmele, S., Woltran, S.: An extension-based approach to belief revision in abstract argumentation. In: Yang and Wooldridge [52], pp. 2926–2932. URL http://ijcai.org/Abstract/15/414
27. Diller, M., Wallner, J.P., Woltran, S.: Reasoning in abstract dialectical frameworks using quantified boolean formulas. Argument & Computation **6**(2), 149–177 (2015). DOI 10.1080/19462166.2015.1036922. URL http://dx.doi.org/10.1080/19462166.2015.1036922
28. Doutre, S., Herzig, A., Perrussel, L.: A Dynamic Logic Framework for Abstract Argumentation. In: C. Baral, G. De Giacomo, T. Eiter (eds.) International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 62–71. AAAI Press (2014)
29. Doutre, S., Maffre, F., McBurney, P.: A dynamic logic framework for abstract argumentation: Adding and removing arguments. In: S. Benferhat, K. Tabia, M. Ali (eds.) Advances in Artificial Intelligence: From Theory to Practice - 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017, *Lecture Notes in Computer Science*, vol. 10351, pp. 295–305. Springer (2017). DOI 10.1007/978-3-319-60045-1_32. URL https://doi.org/10.1007/978-3-319-60045-1_32
30. Doutre, S., Mailly, J.G.: Constraints and changes: a survey of abstract argumentation dynamics. Argument and Computation **9**, 223–248 (2018)
31. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–357 (1995)
32. Fan, X., Toni, F.: On explanations for non-acceptable arguments. In: E. Black, S. Modgil, N. Oren (eds.) Theory and Applications of Formal Argumentation - Third International Workshop, TAFA 2015, Buenos Aires, Argentina, July 25-26, 2015, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 9524, pp. 112–127. Springer (2015). DOI 10.1007/978-3-319-28460-6_7. URL http://dx.doi.org/10.1007/978-3-319-28460-6_7
33. Forbus, K.D.: Introducing actions into qualitative simulation. In: N.S. Sridharan (ed.) Proc. 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89), pp. 1273–1278. Morgan Kaufmann Publishers (1989)
34. Gabbay, D.M.: Dung's argumentation is essentially equivalent to classical propositional logic with the peirce–quine dagger. Logica Universalis **5**(2), 255–318 (2011)
35. Gaggl, S.A., Manthey, N., Ronca, A., Wallner, J.P., Woltran, S.: Improved answer-set programming encodings for abstract argumentation. TPLP **15**(4-5), 434–448 (2015). DOI 10.1017/S1471068415000149. URL http://dx.doi.org/10.1017/S1471068415000149
36. Harel, D.: Dynamic logic. In: D.M. Gabbay, F. Günthner (eds.) Handbook of Philosophical Logic, vol. II, pp. 497–604. D. Reidel, Dordrecht (1984)

37. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
38. Herzig, A.: Belief change operations: a short history of nearly everything, told in dynamic logic of propositional assignments. In: C. Baral, G. De Giacomo (eds.) Proc. KR 2014. AAAI Press (2014)
39. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems. In: T. Walsh (ed.) International Joint Conference on Artificial Intelligence (IJCAI), pp. 228–233. IJCAI/AAAI, Barcelona (2011). URL http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html. Erratum at http://www.irit.fr/~Andreas.Herzig/P/Ijcai11.html
40. Herzig, A., Rifi, O.: Propositional belief base update and minimal change. Artificial Intelligence Journal **115**(1), 107–138 (1999). URL ../Herzig/P/aij99.html
41. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: P. Gärdenfors (ed.) Belief revision, pp. 183–203. Cambridge University Press (1992). (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)
42. Lang, J.: Belief update revisited. In: Proc. of the 10th International Joint Conference on Artificial Intelligence (IJCAI'07), pp. 2517–2522 (2007)
43. Lifschitz, V.: Circumscription. In: D.M. Gabbay, D.M. Gabbay, C. Hogger, J.A. Robinson (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 3 - Nonmonotonic Reasoning and Uncertain Reasoning, pp. 298–352. Oxford University Press (1994)
44. Mailly, J.G.: Dynamics of argumentation frameworks. ThÃĺse de doctorat, Univ. Artois, Lens, France (2015)
45. Niskanen, A., Wallner, J.P., Järvisalo, M.: Optimal status enforcement in abstract argumentation. In: S. Kambhampati (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 1216–1222. IJCAI/AAAI Press (2016). URL http://www.ijcai.org/Abstract/16/176
46. Nofal, S., Atkinson, K., Dunne, P.E.: Algorithms for decision problems in argument systems under preferred semantics. Artificial Intelligence **207**, 23–51 (2014)
47. Dupin de Saint-Cyr, F., Bisquert, P., Cayrol, C., Lagasquie-Schiex, M.: Argumentation update in YALLA (yet another logic language for argumentation). Int. J. Approx. Reasoning **75**, 57–92 (2016). DOI 10.1016/j.ijar.2016.04.003. URL http://dx.doi.org/10.1016/j.ijar.2016.04.003
48. Wallner, J.P., Niskanen, A., Järvisalo, M.: Complexity results and algorithms for extension enforcement in abstract argumentation. In: D. Schuurmans, M.P. Wellman (eds.) Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA., pp. 1088–1094. AAAI Press (2016). URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12228
49. Winslett, M.A.: Reasoning about action using a possible models approach. In: Proc. 7th Conf. on Artificial Intelligence (AAAI'88), pp. 89–93. St. Paul (1988)
50. Winslett, M.A.: Updating Logical Databases. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press (1990)
51. Winslett, M.A.: Updating logical databases. In: D.M. Gabbay, C.J. Hogger, J.A. Robinson (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4, pp. 133–174. Oxford University Press (1995)
52. Yang, Q., Wooldridge, M. (eds.): Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. AAAI Press (2015). URL http://ijcai.org/proceedings/2015