A Dynamic Logic Framework for Abstract Argumentation: Adding and Removing Arguments

Sylvie Doutre, Faustine Maffre, Peter Mcburney

A Dynamic Logic Framework for Abstract Argumentation: Adding and Removing Arguments

Sylvie Doutre[1(✉)], Faustine Maffre[1], and Peter McBurney[2]

[1] IRIT, Université Toulouse 1 Capitole, Toulouse, France
doutre@irit.fr
[2] King's College London, London, UK

Abstract. A dynamic framework, based on the Dynamic Logic of Propositional Assignments (DL-PA), has recently been proposed for Dung's abstract argument system. This framework allows the addition and the removal of attacks, and the modification of the acceptance status of arguments. We here extend this framework in order to capture the addition and the removal of arguments. We then apply the framework on an access control case, where an agent engages in an argued dialogue to access some information controlled by another agent.

1 Introduction

In Dung's approach to argumentation [7], an argument system is represented as a set of (abstract) arguments and a binary attack relation between these arguments. Several semantics—ways to evaluate which arguments should be accepted—have been developed from this model (see [2]), some of them referred as "extension-based". These are semantics which define acceptable sets of arguments, called extensions.

Logical representations of Dung's approach of argumentation have been presented, based for instance on propositional logic [3], or more recently on dynamic logic [6]. In these contributions, the argument system is described by a boolean formula and each type of semantics is also represented by a boolean formula; for a given semantics, any interpretation of the propositional variables for which both formulas are true characterizes an extension. In [6], the use of a dynamic logic (DL-PA—Dynamic Logic of Propositional Assignments [1]) furthermore allows us to model updates of the argument system, such as addition or removal of an attack, or modification of extensions. [6] is not the only approach which has tackled the question of the dynamics of argument systems (see [5,15] for instance), but this one provides a single framework which encompasses at the same time the argument system, the logical definition of the change to enforce, and the change operations to perform. Moreover, the logic it is based on is non-specific to argumentation since it has already been applied in various contexts [8,12].

Following an idea that was triggered in [6], we extend [6]'s framework to capture addition and removal of arguments. An extension of Dung's system is proposed to take into account, within the set of arguments, those which are currently considered, or,

say, enabled. The formulas that describe the argument system and the semantics are modified to take into account this new notion of enabled arguments. We illustrate our contribution with an example from [14], in which the authors present a protocol for agents engaging in an argued dialogue to access some information.

The paper is organized as follows. In the next section we introduce the example of access control dialogue that we will use throughout the paper. In Sect. 3 we show how to extend the framework of [6] to capture addition and removal of arguments. In Sect. 4 we show how this extension can be used to formalize updates of the argument system during the dialogue. Section 5 concludes.

2 Our Running Example

In this example taken from [14], an agent, called the *client* (here, Brussels agent), wants to access some information. He engages a dialogue with the agent controlling it—the *server* (London agent)—in order to convince him to grant authorization to access this information. The second agent explains the reasons why he cannot give this access by presenting all the arguments attacking the client's arguments that he knows.

"Robert is a British businessman visiting Brussels for a meeting. During his visit he becomes ill and is taken unconscious into hospital. The staff of the hospital suspect Robert has had a heart attack and seek to prescribe appropriate drugs for his condition. Unfortunately the safe choice of drugs depends upon various factors, including prior medical conditions that Robert might have and other drugs he may be taking. The hospital's agent is given the goal of finding out the required information about Robert, from the agent representing his London doctor.

In order to gain access to information about Robert, the agent of Brussels Hospital (B. agent) establishes the following dialogue with the London agent (L. agent):

0. **B. agent:** I would like to dialog with the agent of Robert's British doctor; I request Robert's health record.
1. **L. agent:** I cannot provide you Robert's health record because Robert has only given his British doctor limited consent to pass on his personal information (argument a_1).
2. **B. agent:** This record could possibly include information that could affect the treatment of Robert's heart failure. I request it, Robert's life may be at stake (argument a_2)!
3. **L. agent:** I cannot divulge this information, because British law prohibits passing on information without the consent of the provider of the information (argument a_3).
4. **B. agent:** EC law takes precedence over British law when it would be in the interests of the owner to divulge the information (argument a_4). You should allow me to access the record.
5. **L. agent:** Only Robert could decide what would be in his interests (argument a_5).
6. **B. agent:** Robert's doctor owes a duty of care to Robert and, should he die, the doctor might be sued by his family, or the Brussels hospital, or both (argument a_6).
7. **L. agent:** OK. I provide you the requested record: Robert's history of diabetes is..."

Some arguments are directly in favor of giving the permission to access the information (a_2 and a_6), some are directly against this permission (a_1 and a_3), while others

are not directly linked to the permission (a_4 and a_5) but contradict other arguments. The dialogue ends after London agent has implicitly considered acceptable Brussels' final argument (a_6), which supports the permission to give the requested information to Brussels. No such argument was acceptable for London agent earlier in the dialogue.

In this paper, we focus on capturing the evolution of the acceptability of arguments for London agent, after each addition of argument. This can be seen as an a posteriori analysis of the dialogue, that allows us to understand the reasons why the permission to access the information was first refused, and then given.

3 Representing Argument Systems

We extend here the definition of an argument system, and of its logical formalization [6]. We present DL-PA logic and how to compute extensions with DL-PA programs.

3.1 Logical Representation of an Argument System

Argument System. In order to capture addition and removal of arguments, we add a component to Dung's argument system [7]: the set of currently considered ("enabled") arguments, among all the arguments of the system. These arguments are those which are known, at some step, in the context of a dialogue.

Definition 1. *An* argument system for enablement *is a tuple* $\mathcal{F} = (\mathcal{A}, \mathcal{A}^{En}, R)$ *where* $\mathcal{A}$ *is a finite set of abstract arguments;* $\mathcal{A}^{En} \subseteq \mathcal{A}$ *is the set of* enabled *arguments, and* $R \subseteq \mathcal{A} \times \mathcal{A}$ *is the attack relation:* $(a, b) \in R$ *means that* a *attacks* b.

$\mathcal{A}$ contains all possible arguments that may arise during the dialogue, while $\mathcal{A}^{En}$ constitutes the set of arguments that have been uttered until now.

An argument system for enablement is represented by a directed graph whose nodes are enabled arguments and edges are attacks between enabled arguments: there is an edge between a and b if $(a, b) \in R$, $a \in \mathcal{A}^{En}$ and $b \in \mathcal{A}^{En}$. Hence the representation contains less information than the original model. Note that as soon as an argument is enabled, all its attacks from (resp. to) other arguments are considered. When $\mathcal{A}^{En} = \mathcal{A}$, the argument system for enablement comes down to Dung's argument system.

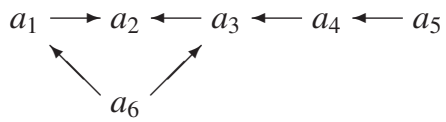
Example 1. In our running example, six arguments were presented during the dialogue: $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$. Let us write $\mathcal{F}_i = (\mathcal{A}, \mathcal{A}_i^{En}, R)$ the argument system at step i (steps are numbered as in Sect. 2, from 0 to 6). The attack relation, from the point of view of London agent, and according to [14], is $R = \{(a_1, a_2), (a_3, a_2), (a_4, a_3), (a_5, a_4), (a_6, a_1), (a_6, a_3)\}$.

At step 0, no argument is considered: $\mathcal{A}_0^{En} = \emptyset$. Then, a_1 is uttered: $\mathcal{A}_1^{En} = \{a_1\}$. This argument is not attacked nor attacks any enabled argument; the graph that represents the argument system thus is:

Then a_2 is presented ($\mathcal{A}^{En}_2 = \{a_1, a_2\}$) and the graph becomes:

$$a_1 \longrightarrow a_2$$

The dialogue goes on until all arguments are presented:



The set of enabled arguments $\mathcal{A}^{En}_6$ is then equal to $\mathcal{A}$.

In order to logically represent an argument system for enablement $\mathcal{F}$, we extend the language of [6]. As in [6], a set of attack variables is used to represent attacks:

$$\mathtt{ATT}_{\mathcal{A}} = \{\mathsf{Att}_{a,b} \ : \ (a,b) \in \mathcal{A} \times \mathcal{A}\}.$$

$\mathsf{Att}_{a,b}$ means that a attacks b. We consider in addition a set of *enablement variables*:

$$\mathtt{EN}_{\mathcal{A}} = \{\mathsf{En}_a \ : \ a \in \mathcal{A}\},$$

where En_a means that a is enabled (or "considered").

Let $\mathcal{L}_{\mathsf{Att,En}}$ be the set of all formulas that are built variables from $\mathtt{ATT}_{\mathcal{A}} \cup \mathtt{EN}_{\mathcal{A}}$. The *theory* describing the framework $\mathcal{F} = (\mathcal{A}, \mathcal{A}^{En}, R)$ is the following boolean formula:

$$\mathsf{Th}_{\mathcal{F}} = \Big(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{a\in\mathcal{A}^{En}} \mathsf{En}_a \Big) \wedge \Big(\bigwedge_{a\notin\mathcal{A}^{En}} \neg\mathsf{En}_a \Big)$$

Note that in the theory, $\mathsf{Att}_{a,b}$ is true even if a or b are not enabled. This is required to not loose any information, and will be important for updates (see Sect. 4).

Example 2. We have 6 arguments in our running example. This means that $\mathtt{ATT}_{\mathcal{A}}$ contains 36 variables and $\mathtt{EN}_{\mathcal{A}}$ contains 6 variables. Therefore, whatever the step i of the dialogue, $\mathsf{Th}_{\mathcal{F}_i}$ is a conjunction of 42 literals. For simplification, we do not write explicitly every attack literal $\mathsf{Att}_{a,b}$ and $\neg\mathsf{Att}_{a,b}$ (Att_{a_1,a_2}, Att_{a_3,a_2}, and $\neg\mathsf{Att}_{a_1,a_3}$ ...) but we keep the expression $(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b}) \wedge (\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b})$. Note that, since in our case the attack relation R is constant, this expression remains constant.

As examples, the theory at step 0 is:

$$\mathsf{Th}_{\mathcal{F}_0} = \Big(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b} \Big) \wedge \neg\mathsf{En}_{a_1} \wedge \neg\mathsf{En}_{a_2} \wedge \neg\mathsf{En}_{a_3} \wedge \neg\mathsf{En}_{a_4} \wedge \neg\mathsf{En}_{a_5} \wedge \neg\mathsf{En}_{a_6}$$

and the theory at step 3 is:

$$\mathsf{Th}_{\mathcal{F}_3} = \Big(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b} \Big) \wedge \mathsf{En}_{a_1} \wedge \mathsf{En}_{a_2} \wedge \mathsf{En}_{a_3} \wedge \neg\mathsf{En}_{a_4} \wedge \neg\mathsf{En}_{a_5} \wedge \neg\mathsf{En}_{a_6}$$

Argumentation Semantics. Given an argument system, an acceptability semantics identifies a set of extensions, i.e., acceptable sets of arguments. There may be none, one or several extensions. As in [6], we characterize extensions thanks to a set of *acceptability* variables:

$$\mathtt{IN}_{\mathcal{A}} = \{\mathsf{In}_a \ : \ a \in \mathcal{A}\}$$

where In_a stands for "argument a is in the extension".

Without the notion of enabled arguments (that is, when all arguments are always considered), it has already been shown how to encode the stable, admissible and complete semantics with propositional formulas [6]. These definitions can be adapted to consider enabled arguments only. In this case, extensions are subsets of $\mathcal{A}^{En}$ and only attacks between arguments from $\mathcal{A}^{En}$ are considered. In the corresponding formulas, we only include an argument if it is enabled. Also, attacks must be considered only if they link enabled arguments. To this end, we define the following formula: $\mathsf{Att}^{En}_{a,b} = \mathsf{Att}_{a,b} \wedge \mathsf{En}_a \wedge \mathsf{En}_b$. Now we can easily transform formulas from [6]: we check if the argument is enabled, otherwise it will not be included in the extension, and we replace attacks variables $\mathsf{Att}_{a,b}$ by formulas $\mathsf{Att}^{En}_{a,b}$ to ensure they are indeed present. We illustrate this transformation to capture the *stable* semantics. Let $\mathcal{L}_{\mathsf{Att},\mathsf{En},\mathsf{In}}$ be the language of formulas built from $\mathbb{P} = \mathtt{ATT}_{\mathcal{A}} \cup \mathtt{EN}_{\mathcal{A}} \cup \mathtt{IN}_{\mathcal{A}}$.

Definition 2. *Let $\mathcal{F} = (\mathcal{A}, \mathcal{A}^{En}, R)$ be an argument system for enablement. Let $S \subseteq \mathcal{A}^{En}$ be a set of enabled arguments. S is* conflict-free *if $\forall a, b \in S$, $(a, b) \notin R$. S is a* stable extension *if S is conflict-free and $\forall b \in \mathcal{A}^{En} \backslash S$, $\exists a \in S$ such that $(a, b) \in R$ (any considered argument outside the extension is attacked by at least one in the extension).*

Note that we do not need to restrict R to the set of considered arguments as a and b both belong to S which is a subset of $\mathcal{A}^{En}$.

The following formula captures the stable semantics[1]:

$$\mathsf{Stable}_{\mathcal{A}} = \bigwedge_{a \in \mathcal{A}} \left(\left(\mathsf{En}_a \rightarrow (\mathsf{In}_a \leftrightarrow \neg \bigvee_{b \in \mathcal{A}} (\mathsf{In}_b \wedge \mathsf{Att}^{En}_{b,a})) \right) \wedge \left(\neg \mathsf{En}_a \rightarrow \neg \mathsf{In}_a \right) \right)$$

Extensions and Valuations. A *valuation* is a subset of the set of variables $\mathbb{P} = \mathtt{ATT}_{\mathcal{A}} \cup \mathtt{EN}_{\mathcal{A}} \cup \mathtt{IN}_{\mathcal{A}}$: the variables that are currently true. The set of all valuations is $2^{\mathbb{P}}$. Valuations are denoted by v, v', v_1, v_2, etc. A given valuation determines the truth value of the boolean formulas of the language $\mathcal{L}_{\mathsf{Att},\mathsf{En},\mathsf{In}}$ in the usual way. For a formula φ, a valuation where φ is true is called a *model of* φ and the set of models of φ is denoted by $\|\varphi\|$. A formula is propositionally valid if it is true in all valuations, i.e., if $\|\varphi\| = 2^{\mathbb{P}}$. The following results are adapted from [6].

Proposition 1. *Let $\mathcal{F} = (\mathcal{A}, \mathcal{A}^{En}, R)$ be an argument system for enablement. Let $E \subseteq \mathcal{A}^{En}$. Consider $v_E = \{\mathsf{Att}_{a,b} \ : \ (a, b) \in R\} \cup \{\mathsf{En}_a \ : \ a \in \mathcal{A}^{En}\} \cup \{\mathsf{In}_a \ : \ a \in E\}$. E is a* stable *extension of $\mathcal{F}$ if and only if v_E is a model of $\mathsf{Th}_{\mathcal{F}} \wedge \mathsf{Stable}_{\mathcal{A}}$.*

[1] 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: $\mathsf{Stable}_{\mathcal{A},\mathcal{A}^{En}} = \bigwedge_{a \in \mathcal{A}^{En}} (\mathsf{In}_a \leftrightarrow \neg \bigvee_{b \in \mathcal{A}^{En}} (\mathsf{In}_b \wedge \mathsf{Att}_{b,a})) \wedge \bigwedge_{a \notin \mathcal{A}^{En}} \neg \mathsf{In}_a$. This highlights the fact that when all arguments are enabled, i.e., when $\mathcal{A}^{En} = \mathcal{A}$, we indeed retrieve formulas presented in [6].

Example 3. Let us consider $\mathcal{F}_3 = (\mathcal{A}, \mathcal{A}_3^{En}, R)$ with $\mathcal{A}$ and R as in Example 1 and $\mathcal{A}_3^{En} = \{a_1, a_2, a_3\}$.

$$a_1 \longrightarrow a_2 \longleftarrow a_3$$

Let $\nu_{\mathsf{Th}} = \{\mathsf{Att}_{a,b} : (a,b) \in R\} \cup \{\mathsf{En}_{a_1}, \mathsf{En}_{a_2}, \mathsf{En}_{a_3}\}$. Note that the theory $\mathsf{Th}_{\mathcal{F}_3}$ (see Example 2) is true in ν_{Th}. $\mathcal{F}_3$ has only one stable extension: $\{a_1, a_3\}$. Hence $\nu_{\{a_1,a_3\}} = \nu_{\mathsf{Th}} \cup \{\mathsf{In}_{a_1}, \mathsf{In}_{a_3}\}$ is a model of $\mathsf{Th}_{\mathcal{F}_3} \wedge \mathsf{Stable}_{\mathcal{A}}$.

3.2 DL-PA: Dynamic Logic of Propositional Assignments

Dynamic Logic of Propositional Assignments DL-PA [1,10] is a variant of Propositional Dynamic Logic PDL [9], with operators for sequential and nondeterministic composition of programs, test and iteration (the Kleene star), but where atomic programs are assignments of truth values to propositional variables. Modal operators associated to programs can express that some property holds after the modification of the current valuation by the program. In our case, they will allow us to update the theory associated to the argument system as the dialogue progresses.

We will see that the results from [6] are still applicable in our framework. Hence we also consider the star-free version of DL-PA [10] with the converse operator.

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

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

where p ranges over $\mathbb{P}$.

The formula $\langle\pi\rangle\varphi$ reads "after some execution of the program π formula φ holds". The formula $[\pi]\varphi$, abbreviating $\neg\langle\pi\rangle\neg\varphi$, reads "after every execution of the program π formula φ holds". 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 ("$\cup$") and test ("(.)?") are from PDL. The operator "$(.)^{-}$" is the converse operator: the formula $\langle\pi^{-}\rangle\varphi$ reads "before some execution of the program π formula φ was true". Other boolean operators are abbreviated as usual. Like in PDL, skip abbreviates $\top?$ ("nothing happens").

Semantics and Validity. Models of DL-PA formulas are subsets of the set of propositional variables $\mathbb{P}$, i.e., valuations. DL-PA programs are interpreted by means of a relation between valuations. Atomic programs $p{\leftarrow}\top$ and $p{\leftarrow}\bot$ are interpreted as update operations on valuations, and complex programs are interpreted just as in PDL by mutual recursion. Table 1 gives the interpretation of the DL-PA connectives.

Two formulas φ_1 and φ_2 are *formula equivalent* if $\|\varphi_1\| = \|\varphi_2\|$. Two programs π_1 and π_2 are *program equivalent* if $\|\pi_1\| = \|\pi_2\|$. In that case we write $\pi_1 \equiv \pi_2$. An expression is a formula or a program; equivalence is preserved under replacement of a sub-expression by an equivalent expression [1]. A formula φ is DL-PA *valid* if it is true in all valuations, i.e., if $\|\varphi\| = 2^{\mathbb{P}}$.

Table 1. Interpretation of the DL-PA connectives

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

3.3 Constructing Extensions with DL-PA

As in [6], we can build extensions of an argument system by means of DL-PA programs. We recall the program $\mathsf{vary}(P)$, from [6], with $P = \{p_1, \ldots, p_n\}$ a set of variables:

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

$\mathsf{vary}(P)$ is a sequence of subprograms, each step i of the sequence nondeterminiscally setting the value of p_i to true or to false. (Note that the ordering of the p_i does not matter.) Executing $\mathsf{vary}(P)$ from a valuation v will lead to any valuation where variables from $\mathbb{P} \setminus P$ have the same value than in v, while variables from P can have any value.

Given an argument system $\mathcal{F}$, the idea of [6] is to start from a valuation where the theory $\mathsf{Th}_{\mathcal{F}}$ is verified, and vary accessibility variables In_a; this leads to several valuations, some corresponding to an extension. To "filter" these valuations to keep stable extensions only, we test the formula capturing the semantics (see Sect. 3.1). More formally, we use the following program to build stable extensions:

$$\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}} = \mathsf{vary}(\mathtt{IN}_{\mathcal{A}}); \mathsf{Stable}_{\mathcal{A}}?$$

Example 4. In Example 3, we have seen that $v_{\{a_1,a_3\}} = v_{\mathsf{Th}} \cup \{\mathsf{In}_{a_1}, \mathsf{In}_{a_3}\}$ is the only stable extension. $\mathsf{Th}_{\mathcal{F}_3}$ is true in this valuation thanks to v_{Th}, and the values of the accessibility variables describe the extension. Executing $\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}}$ from, e.g., v_{Th}, will lead exactly to $v_{\{a_1,a_3\}}$; $\mathsf{Stable}_{\mathcal{A}}$ is not true for any other valuation linked by $\mathsf{vary}(\mathtt{IN}_{\mathcal{A}})$.

The following results are adapted from [6].

Lemma 1. *Let $\mathcal{F}$ be an argument system for enablement. Let v_1 be a model of $\mathsf{Th}_{\mathcal{F}}$. Then $(v_1, v_2) \in \|\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}}\|$ if and only if v_2 is a model of $\mathsf{Th}_{\mathcal{F}} \wedge \mathsf{Stable}_{\mathcal{A}}$.*

The main result about the construction of extensions with DL-PA is as follows.

Proposition 2. *Let $\mathcal{F} = (\mathcal{A}, \mathcal{A}^{En}, R)$ be an argument system for enablement. The following equivalence is* DL-PA *valid:*

$$\mathsf{Th}_{\mathcal{F}} \wedge \mathsf{Stable}_{\mathcal{A}} \leftrightarrow \langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}}$$

Remember that $\langle\pi^-\rangle\varphi$ means "before some execution of the program π formula φ was true". This indeed corresponds to the update of $\mathsf{Th}_{\mathcal{F}}$ by $\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}}$.

4 Updating the Argument System Throughout the Dialogue

With the extended framework, we are now able to model changes that may happen in a dialogue, that is, addition, and possibly, removal of arguments. We update the theory corresponding to an argument system to this end:

$$\mathsf{Th}_{\mathcal{F}} \diamond \mathsf{En}_a = \langle (\mathsf{En}_a \leftarrow \top)^- \rangle \mathsf{Th}_{\mathcal{F}}$$
$$\mathsf{Th}_{\mathcal{F}} \diamond \neg\mathsf{En}_a = \langle (\mathsf{En}_a \leftarrow \bot)^- \rangle \mathsf{Th}_{\mathcal{F}}$$

where $\mathsf{Th}_{\mathcal{F}} \diamond \mathsf{En}_a$ refers to updating the system to add (enable) argument a and $\mathsf{Th}_{\mathcal{F}} \diamond \neg\mathsf{En}_a$ to updating the system to remove (disable) argument a. 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. The framework, being an extension of [6], however allows us to remove some of these attacks, or add extra ones, if necessary.

Example 5. In our running example, at step 0, $\mathcal{A}_0^{En} = \emptyset$, and $\mathsf{Th}_{\mathcal{F}_0}$ is as described in Example 2. At step 1, argument a_1 is enabled. The theory is thus updated as follows:

$$\mathsf{Th}_{\mathcal{F}_0} \diamond \mathsf{En}_{a_1} = \langle (\mathsf{En}_{a_1} \leftarrow \top)^- \rangle \mathsf{Th}_{\mathcal{F}_0}$$

Using properties of DL-PA, we explain in details this update. First, it is shown in [6] that $(p \leftarrow \top)^-$ is equivalent to $p? \cup (p?; p \leftarrow \bot)$ (p is now true and was either already true or was false). Hence:

$$\mathsf{Th}_{\mathcal{F}_0} \diamond \mathsf{En}_{a_1} \equiv \langle \mathsf{En}_{a_1}? \cup (\mathsf{En}_{a_1}?; \mathsf{En}_{a_1} \leftarrow \bot) \rangle \mathsf{Th}_{\mathcal{F}_0}$$

DL-PA shares most properties about program operators with PDL [1], such as:

$$\langle \pi \cup \pi' \rangle \varphi \leftrightarrow \langle \pi \rangle \varphi \vee \langle \pi' \rangle \varphi \qquad \langle \pi; \pi' \rangle \varphi \leftrightarrow \langle \pi \rangle \langle \pi' \rangle \varphi \qquad \langle \chi? \rangle \varphi \leftrightarrow \chi \wedge \varphi$$

Using these, we can transform our updated theory:

$$\mathsf{Th}_{\mathcal{F}_0} \diamond \mathsf{En}_{a_1} \equiv (\mathsf{En}_{a_1} \wedge \mathsf{Th}_{\mathcal{F}_0}) \vee (\mathsf{En}_{a_1} \wedge \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{Th}_{\mathcal{F}_0})$$

The first part of the disjunction is equivalent to $\bot$ since $\mathsf{Th}_{\mathcal{F}_0}$ is a conjunction of literals and one is $\neg\mathsf{En}_{a_1}$. For the second part, we consider two properties of DL-PA [1]:

$$\langle p \leftarrow \top \rangle (\varphi \wedge \varphi') \leftrightarrow \langle p \leftarrow \top \rangle \varphi \wedge \langle p \leftarrow \top \rangle \varphi' \qquad \langle p \leftarrow \top \rangle \neg\varphi \leftrightarrow \neg \langle p \leftarrow \top \rangle \varphi$$

With these equivalences, we know that in $\langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{Th}_{\mathcal{F}_0}$, the operator $\langle \mathsf{En}_{a_1} \leftarrow \bot \rangle$ can distribute over the conjunction and be placed before every literal of $\mathsf{Th}_{\mathcal{F}_0}$, and that for negative literals, $\langle \mathsf{En}_{a_1} \leftarrow \bot \rangle$ can be "pushed" against the variable. We obtain:

$$\begin{aligned}\mathsf{Th}_{\mathcal{F}_0} \diamond \mathsf{En}_{a_1} \equiv \mathsf{En}_{a_1} \wedge \Big(\bigwedge_{(a,b) \in R} \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{(a,b) \notin R} \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{Att}_{a,b} \Big) \\ \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_1} \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_2} \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_3} \\ \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_4} \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_5} \wedge \neg \langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_6}\end{aligned}$$

We finally use a last DL-PA property [1]:

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

Most of the conjuncts fall in the second category and thus are not affected by the program, except $\neg\langle \mathsf{En}_{a_1} \leftarrow \bot \rangle \mathsf{En}_{a_1}$ which is equivalent to $\neg\bot$, that is, to $\top$, and thus will disappear from the conjunction. We end with:

$$\mathsf{Th}_{\mathcal{F}_0} \diamond \mathsf{En}_{a_1} \equiv \mathsf{En}_{a_1} \wedge \Big(\bigwedge_{(a,b)\in R} \mathsf{Att}_{a,b} \Big) \wedge \Big(\bigwedge_{(a,b)\notin R} \neg\mathsf{Att}_{a,b} \Big)$$
$$\wedge \neg\mathsf{En}_{a_2} \wedge \neg\mathsf{En}_{a_3} \wedge \neg\mathsf{En}_{a_4} \wedge \neg\mathsf{En}_{a_5} \wedge \neg\mathsf{En}_{a_6}$$

which is the theory $\mathsf{Th}_{\mathcal{F}_1}$, i.e., for $\mathcal{A}_1^{En} = \{a_1\}$.

Example 6. We can finally fully run through our main example. $\mathcal{A}$ and R remain constant and are as described in Example 1. We are going to run the example from step 0 to step 6, hence showing the addition/enablement of arguments step after step. Notice that it may be run the other way round, from step 6 to step 0; the removal/disablement of arguments would then be illustrated.

At step 0, $\mathcal{A}_0^{En} = \emptyset$, and $\mathsf{Th}_{\mathcal{F}_0}$ is as in Example 2. The execution of

$$\langle (\mathsf{makeExt}_{\mathcal{A}}^{\mathsf{Stable}})^{-} \rangle \mathsf{Th}_{\mathcal{F}_0}$$

allows one to get the only stable extension of $\mathcal{F}_0$: $\emptyset$.

When a_1 is uttered, the set of enabled arguments becomes $\mathcal{A}_1^{En} = \{a_1\}$. As we have seen in Example 5:

$$\mathsf{Th}_{\mathcal{F}_1} = \langle (\mathsf{En}_{a_1} \leftarrow \top)^{-} \rangle \mathsf{Th}_{\mathcal{F}_0}$$

Executing $\mathsf{makeExt}_{\mathcal{A}}^{\mathsf{Stable}}$ from any valuation satisfying $\mathsf{Th}_{\mathcal{F}_1}$ will lead to one valuation, where In_{a_1} is true and every In_{a_i} for $i \in \{2, \ldots, 6\}$ is false; this means we obtain one stable extension: $\{a_1\}$.

We summarize the results at each step of the dialogue in Table 2. Numbers in the first column are steps. In the second column, we write, first, the DL-PA formula describing the updated theory, and second, the DL-PA formula true in valuations corresponding to (stable) extensions. Then in the third column we give the current graph representing the argument system, and the set of extensions.

After the server (London agent) has presented all his arguments, we end up with one extension: $\{a_2, a_5, a_6\}$. In this setting, he accepts to provide the information as argument a_6, which directly supports the permission, belongs to at least one extension (following the principle of trustfulness of [14]).

5 Conclusion

This paper presents an extension of the formal framework of [6], which allows us to update the argument system by adding or removing an argument. We achieve this by

Table 2. Evolution of the argument system and of the set of extensions during the dialogue

Step	DL-PA formulas	Graph and extensions
0	$\mathsf{Th}_{\mathcal{F}_0}$	
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_0}$	$\{\emptyset\}$
1	$\langle(\mathsf{En}_{a_1}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_0} = \mathsf{Th}_{\mathcal{F}_1}$	a_1
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_1}$	$\{\{a_1\}\}$
2	$\langle(\mathsf{En}_{a_2}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_1} = \mathsf{Th}_{\mathcal{F}_2}$	$a_1 \longrightarrow a_2$
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_2}$	$\{\{a_1\}\}$
3	$\langle(\mathsf{En}_{a_3}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_2} = \mathsf{Th}_{\mathcal{F}_3}$	$a_1 \longrightarrow a_2 \longleftarrow a_3$
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_3}$	$\{\{a_1, a_3\}\}$
4	$\langle(\mathsf{En}_{a_4}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_3} = \mathsf{Th}_{\mathcal{F}_4}$	$a_1 \longrightarrow a_2 \longleftarrow a_3 \longleftarrow a_4$
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_4}$	$\{\{a_1, a_4\}\}$
5	$\langle(\mathsf{En}_{a_5}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_4} = \mathsf{Th}_{\mathcal{F}_5}$	$a_1 \longrightarrow a_2 \longleftarrow a_3 \longleftarrow a_4 \longleftarrow a_5$
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_5}$	$\{\{a_1, a_3, a_5\}\}$
6	$\langle(\mathsf{En}_{a_6}\leftarrow\top)^-\rangle\mathsf{Th}_{\mathcal{F}_5} = \mathsf{Th}_{\mathcal{F}_6}$	$a_1 \longrightarrow a_2 \longleftarrow a_3 \longleftarrow a_4 \longleftarrow a_5$, a_6
	$\langle(\mathsf{makeExt}^{\mathsf{Stable}}_{\mathcal{A}})^-\rangle\mathsf{Th}_{\mathcal{F}_6}$	$\{\{a_2, a_5, a_6\}\}$

including a set of currently enabled arguments to the argument system and a new set of propositional variables reflecting this set in the logical representation. With new appropriate formulas for the argument system and semantics, the formalization of [6] is general enough to transpose well to this extension. Our new framework allows us to easily model the evolution of the argument graph during dialogue, as shown with the example of [14]. It is interesting to note that while we focused on the stable semantics, any semantics that can be described by a logical formula can be encoded in our framework.

Implemented theorem proving methods for DL-PA have not yet been developed, DL-PA being quite recent, but a paper such as the present on applications motivates implementation work.

The evolution of the argument system and the set of extensions during the dialogue, as illustrated in Table 2, is similar to the key idea of Discourse Representation Theory (DRT) [11] in linguistics. This idea is that the semantics of a dialogue may be constructed by the participants, jointly and incrementally as the dialogue proceeds. Other work in multi-agent systems has also drawn on these ideas, for example, [4, 13].

The dialogue analysis has been made from the point of view of one agent in this paper, and a posteriori. A perspective is to extend it to several agents, and to capture some ongoing strategy each one of them may have to fulfill own goals. In this case, agents may or may not agree on the set of arguments and on the set of attacks. For every of his arguments, an agent could compute the set of extensions for the current graph plus this argument, and thereby identify the argument that brings him closer to his personal goal. This is similar to the update of extensions in [6] but with a different approach (modifying arguments rather than attacks). We leave this to future work.

Acknowledgements. This work benefited from the support of the AMANDE project (ANR-13-BS02-0004) of the French National Research Agency (ANR).

References

1. 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)
2. Baroni, P., Giacomin, M.: Semantics of abstract argument systems. In: Simari, G., Rahwan, I. (eds.) Argumentation in Artificial Intelligence, pp. 25–44. Springer, US (2009)
3. Besnard, P., Doutre, S.: Checking the acceptability of a set of arguments. In: 10th International Workshop on Non-Monotonic Reasoning (NMR 2004), pp. 59–64 (2004)
4. Bratu, M., Andreoli, J.-M., Boissier, O., Castellani, S.: A software infrastructure for negotiation within inter-organisational alliances. In: Padget, J., Shehory, O., Parkes, D., Sadeh, N., Walsh, W.E. (eds.) AMEC 2002. LNCS, vol. 2531, pp. 161–179. Springer, Heidelberg (2002). doi:10.1007/3-540-36378-5_10
5. Coste-Marquis, S., Konieczny, S., Mailly, J.G., Marquis, P.: On the revision of argumentation systems: minimal change of arguments statuses. KR **14**, 52–61 (2014)
6. Doutre, S., Herzig, A., Perrussel, L.: A dynamic logic framework for abstract argumentation. In: KR 2014, pp. 62–71. AAAI Press (2014)
7. 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)
8. Gaudou, B., Herzig, A., Lorini, E., Sibertin-Blanc, C.: How to do social simulation in logic: modelling the segregation game in a dynamic logic of assignments. In: Villatoro, D., Sabater-Mir, J., Sichman, J.S. (eds.) MABS 2011. LNCS, vol. 7124, pp. 59–73. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28400-7_5
9. Harel, D.: Dynamic logic. In: Gabbay, D.M., Günthner, F. (eds.) Handbook of Philosophical Logic, vol. II, pp. 497–604. D. Reidel, Dordrecht (1984)
10. Herzig, A., Lorini, E., Moisan, F., Troquard, N.: A dynamic logic of normative systems. In: IJCAI 2011, pp. 228–233 (2011). www.irit.fr/~Andreas.Herzig/P/Ijcai11.html
11. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory, Studies in Linguistics and Philosophy, vol. 42. Kluwer Academic, Dordrecht, The Netherlands (1993)
12. Maffre, F.: Ignorance is bliss: observability-based dynamic epistemic logics and their applications. Ph.D. thesis, University of Toulouse (2016)
13. McBurney, P., Parsons, S.: Posit spaces: a performative theory of e-commerce. In: AAMAS 2003, pp. 624–631. ACM Press (2003)
14. Perrussel, L., Doutre, S., Thévenin, J.-M., McBurney, P.: A persuasion dialog for gaining access to information. In: Rahwan, I., Parsons, S., Reed, C. (eds.) ArgMAS 2007. LNCS, vol. 4946, pp. 63–79. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78915-4_5
15. de Saint-Cyr, F.D., Bisquert, P., Cayrol, C., Lagasquie-Schiex, M.C.: Argumentation update in YALLA (yet another logic language for argumentation). Int. J. Approximate Reasoning **75**, 57–92 (2016)