Using Rewriting Rules to Compute Successful Modifications of an Argumentation System

Dionysios Kontarinis[1] **Alan Perotti**[2] **Elise Bonzon**[1] **Nicolas Maudet**[3]
Leon van der Torre[4] **Serena Villata**[5]

[1] LIPADE, Université Paris Descartes, {dionysios.kontarinis,elise.bonzon}@parisdescartes.fr
[2] Turin University, perotti@unito.it
[3] LIP6, Université Pierre et Marie Curie, nicolas.maudet@lip6.fr
[4] University of Luxembourg, leon.vandertorre@uni.lu
[5] INRIA, Sophia Antipolis, serena.villata@inria.fr

Résumé

Cet article s'intéresse à une approche par systèmes de règles de réécriture du calcul de modifications minimales à effectuer dans un système d'argumentation. Nous étudions tout d'abord les propriétés de ces modifications minimales, puis nous présentons une procédure spécifique dont nous détaillons les propriétés.

Abstract

In this article, we define a procedure which computes the minimal change of an argumentation system that achieves some goal. We first study the properties of such minimal changes. Then we present a procedure based on a set of rewriting rules, and we analyse its properties.

1 Introduction

Debates are pursued with the aim to obtain at the end a set of *accepted arguments*. During these debates, each agent tries to argue in such a way that his own *argumentative goals* belong to the final set of accepted arguments. Given the number of participants and of proposed arguments, it is a challenging task to identify the part of the debate to focus on, to compute how possible moves would affect the current state of the debate, and finally to understand which kind of move is better to play. A *move* modifies the current debate by either adding or removing attacks. A *successful move* brings about the acceptance or rejection of a particular argumentative goal, that is, ensuring that a designated argument belongs (or not) to some (all) extension(s). Specifically, we shall focus on (subset-)minimal successful moves, called *target sets* [3]. Our first contribution in this paper is to provide some general properties of such (minimal) successful moves. We then put forward a *rewriting procedure* and a set of dedicated rules which exploits the recent attack semantics of [10] to compute target sets. Then, we investigate the properties of this procedure.

Our work is inspired by proof theories for abstract argumentation frameworks, as in the work of [9], which treat the problem of how to prove the acceptance (or non-acceptance) of an argument under some semantics. The main new elements introduced here are the following. First, we consider dynamic systems where attacks can be added and removed. Second, we focus on minimal change required to achieve acceptance (or non-acceptance) of an argument and we analyze the properties of minimal change.

Recently, the question of the dynamics of argumentation systems has been studied by several authors [5, 2, 1]. Most relevant to our work is the recent work of Baumann [1]. He studies different types of *expansions*, that is, different ways to modify the current system. For instance the most general kind, *arbitrary expansions*, allow any modification of the system (addition/deletion of attacks). Formally, the problem studied is as follows : given a current argumentation system (AS), given a "goal set" E, find a minimal expansion such that E belongs to at least one extension of the modified AS. The notion of minimality differs from ours, since it relies on a pseudometric measuring the distance (in terms of the number of differences between AS). Another key difference is that our expansions are typically constrained. Most importantly, our work focuses on the design of a procedure which returns the target sets in a given situation.

The paper is organized as follows. Section 2 provides some basic background on abstract argumentation theory [7] and the notion of acceptability. Section 3 formalizes the

notion of *target sets*. In Section 4 we give a set of rewriting rules which are used to define a rewriting procedure and then we prove some of its properties. Finally, Section 5 concludes.

2 Background

In this section, we provide the basic concepts of abstract argumentation frameworks, as proposed by Dung [7], in which the exact content of arguments is left unspecified. In the definition of argumentation system we provide here, the difference compared to [7] is that, we do not only have the standard attack relation (here denoted $\rightarrow$), but we also have two more relations : the $\overset{+}{\rightsquigarrow}$ relation which denotes the attacks which can be added in the system, and the $\overset{-}{\rightsquigarrow}$ relation which denotes the attacks which can be removed from the system.

Definition 2.1 *We define an **argumentation system** as a tuple $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$, where A is a set of arguments, $\rightarrow \subseteq A \times A$ is a binary attack relation between arguments, $\overset{+}{\rightsquigarrow} \subseteq A \times A$ is a binary relation which contains the pairs of arguments which can be added in $\rightarrow$, and $\overset{-}{\rightsquigarrow} \subseteq A \times A$ is a binary relation which contains the pairs of arguments which can be removed from $\rightarrow$.*

Notice that $\overset{-}{\rightsquigarrow} \subseteq \rightarrow$ and that $\overset{+}{\rightsquigarrow} \cap \rightarrow = \{\}$. Moreover, from now on, we will suppose that the set of arguments $|A|$ is finite.

In Dung's framework, the *acceptability of an argument* depends on its membership to some sets, called extensions.

Definition 2.2 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system. Let $C \subseteq A$. The set C is **conflict-free** iff $\nexists a, b \in C$ such that $a \rightarrow b$. An argument $a \in A$ is **acceptable w.r.t.** C iff $\forall x \in A$: if $x \rightarrow a$, then $\exists y \in C$ such that $y \rightarrow x$.*

Several types of extensions (sets of arguments satisfying some properties) have been defined by Dung [7].

Definition 2.3 *A conflict-free set of arguments C is :*

- *An **admissible extension** iff each argument of C is acceptable w.r.t. C.*
- *A **preferred extension** iff it is a maximal (w.r.t. $\subseteq$) admissible extension.*
- *A **complete extension** iff $\forall x \in A$: if x is acceptable w.r.t. C, then $x \in C$.*
- *A **grounded extension** iff it is the minimal (w.r.t. $\subseteq$) complete extension.*

Admissible, preferred, complete and grounded semantics will be respectively denoted Adm, Pref, Comp and Gr in the rest of the paper.

The next question is to decide, given a semantics, which arguments are acceptable. In the case of the admissible, preferred and complete semantics, two notions of acceptability can be defined.

Definition 2.4 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, and let $a \in A$.*
*Argument a is said **credulously accepted** w.r.t. system AS under semantics $S \in \{Adm, Pref, Comp, Gr\}$, denoted $S_\exists(a, AS)$, iff a belongs to at least one extension of AS under the S semantics.*
*Argument a is said **sceptically accepted** w.r.t. AS under semantics $S \in \{Adm, Pref, Comp, Gr\}$, denoted $S_\forall(a, AS)$, iff a it belongs to all the extensions of AS under the S semantics.*

Notice that $\{\}$ is always an admissible extension, therefore it holds that $Adm_\forall(a, AS) = \{\}$. Thus, sceptical acceptability under admissible semantics is not an interesting notion, and we will not refer to it anymore.

Furthermore, as there always exists a unique grounded extension, there is no difference between credulous and sceptical acceptability for grounded semantics. So, if argument $a \in A$ is accepted under the grounded semantics, then we simply denote this by $Gr(a, AS)$. Moreover, as stated in [7], an argument $a \in A$ belongs to the grounded extension if and only if it is sceptically accepted under the complete semantics (thus, $Gr(a, AS) \Leftrightarrow Comp_\forall(a, AS)$). We will use this latter notation to refer to the grounded extension.

In the rest of the paper, we will denote by $Sem = \{Adm, Pref, Comp\}$ the set of admissible, preferred and complete semantics.

Finally, for the sake of readability, if there is no danger of confusing which argumentation system we refer to, we will simply write $\forall S \in Sem$, $S_\exists(a)$, or $S_\forall(a)$, without mentioning the AS.

The following property states that the set of arguments which are credulously accepted under the admissible semantics, are the same as those accepted under the preferred, or the complete semantics.

Property 1 [7] *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, where $|A|$ is finite, and $a \in A$. It holds that $Adm_\exists(a, AS) \Leftrightarrow Pref_\exists(a, AS) \Leftrightarrow Comp_\exists(a, AS)$.*

The case of sceptical acceptability is a bit different. Every argument sceptically accepted under complete semantics is also sceptically accepted under preferred semantics, but the inverse does not hold in the general case.

Property 2 [7] *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, where $|A|$ is finite, and $a \in A$. It holds that $Comp_\forall(a, AS) \Rightarrow Pref_\forall(a, AS)$. The inverse does not necessarily hold.*

As we have just seen, Dung's semantics [7] are stated in terms of sets of arguments, but it is also possible to express them using argument *labeling* [8, 4]. Roughly, an argument is labeled *in* if all its attackers are labeled *out*, it is labeled *out* if it has at least an attacker which is labeled *in* and *undec* otherwise. Villata et al. [10] introduce *attack semantics* where arguments are accepted when there are no *successful* attacks on them. An attack $a \rightarrow b$ is *in* (denoted $a \xrightarrow{1} b$) when its attacker is *in*, *undec* (denoted $a \xrightarrow{?} b$) when its attacker is *undec*, and *out* (denoted $a \xrightarrow{0} b$) when its attacker is *out*. Thus, an attack is successful when it is *in* or *undec*, and unsuccessful when it is *out*.

The difference among these two semantics may be explained by means of a simplified version of the example proposed in [10] and visualized in Figure 1. In argument semantics, an extension for a semantics $S \in Sem$ contains a and c. Thus, b is rejected whereas a and c are accepted. In attack semantics, the attacks $a \rightarrow b$ and $c \rightarrow b$ are successful, whereas $b \rightarrow c$ is unsuccessful.

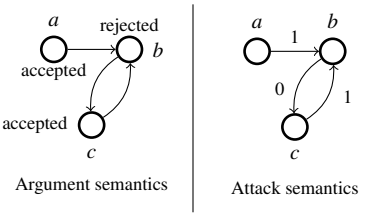


FIGURE 1 – Two argumentation semantics

Boella et al. [3] propose a new kind of labelling, called *conditional labelling*. The idea is to provide the agents with a way to discover the arguments they should attack to get a particular argument accepted or rejected. Conditional labelling extends the idea of argument labelling [4] by assigning a triple of propositional formulae, called *conditional labels*, to every argument in the framework. Given a conditional labelling, the agents have complete knowledge about the consequences of the attacks they may raise on the acceptability of each argument without having to recompute the labelling for each possible set of attacks they may raise.

3 Argumentative goals and target sets

In this work, we consider that attacks are the core components of an argumentation system (we note that the same choice is made in Baumann [1]) and thus prefer to commit to the attack semantics. So, we assume that the arguments of a system cannot change (neither new arguments can be added, nor arguments can be removed). Instead, what may happen, is the addition of new attacks and the removal of some attacks already in the system. A central notion, related to this type of change, is the following notion of atom.

Definition 3.1 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, and let $a, b \in A$ be two arguments. An* **atom** *of AS is defined as follows.*
$Atom(AS) ::= \top \mid \bot \mid\ a \xrightarrow{+} b \mid\ a \xrightarrow{-} b \mid\ a \xrightarrow{1} b \mid\ a \xrightarrow{?} b \mid\ a \xrightarrow{0} b \mid\ a \overset{1}{\twoheadrightarrow} b \mid\ a \overset{?}{\twoheadrightarrow} b \mid\ a \overset{0}{\twoheadrightarrow} b \mid\ PRO(a) \mid\ CON(a)$
The atoms $\top$, $\bot$, $a \xrightarrow{+} b$, $a \xrightarrow{-} b$, $a \xrightarrow{1} b$, $a \xrightarrow{?} b$, and $a \xrightarrow{0} b$ are called **closed atoms**, *whereas the atoms $a \overset{1}{\twoheadrightarrow} b$, $a \overset{?}{\twoheadrightarrow} b$, $a \overset{0}{\twoheadrightarrow} b$, $PRO(a)$ and $CON(a)$ are called* **open atoms**.
Let $Atoms(AS)$ denote the set of all the atoms constructable from AS.

The atom $a \xrightarrow{+} b$ (resp. $a \xrightarrow{-} b$) indicates the action of adding (resp. removing) the attack $a \rightarrow b$. The atoms with double arrow $a \overset{1}{\twoheadrightarrow} b$ (resp. $a \overset{?}{\twoheadrightarrow} b$, resp. $a \overset{0}{\twoheadrightarrow} b$) indicate that we must find a way for attack $a \rightarrow b$ to become '1' (resp. '?', resp. '0'). On the other hand, the atoms with simple arrow $a \xrightarrow{1} b$ (resp. $a \xrightarrow{?} b$, resp. $a \xrightarrow{0} b$), indicate that we have already found a way for attack $a \rightarrow b$ to become '1' (resp. '?', resp. '0'). Finally, the atom $\bot$ indicates failure, and the atom $\top$ indicates success. As it is evident, our focus is on attacks, so the use of attack semantics [10] is more convenient than the use of argument semantics.

Before continuing, let us provide a table which summarizes the different types of arrows we have introduced in our notation so far.

Notation	Type	Meaning
$a \rightarrow b$	Relation	a attacks b
$a \overset{+}{\rightsquigarrow} b$	Relation	$a \rightarrow b$ can be added
$a \overset{-}{\rightsquigarrow} b$	Relation	$a \rightarrow b$ can be removed
$a \xrightarrow{+} b$	Atom	add $a \rightarrow b$
$a \xrightarrow{-} b$	Atom	remove $a \rightarrow b$
$a \xrightarrow{1} b$	Atom (closed)	$a \rightarrow b$ has been set '1'
$a \xrightarrow{?} b$	Atom (closed)	$a \rightarrow b$ has been set '?'
$a \xrightarrow{0} b$	Atom (closed)	$a \rightarrow b$ has been set '0'
$a \overset{1}{\twoheadrightarrow} b$	Atom (open)	$a \rightarrow b$ must be set '1'
$a \overset{?}{\twoheadrightarrow} b$	Atom (open)	$a \rightarrow b$ must be set '?'
$a \overset{0}{\twoheadrightarrow} b$	Atom (open)	$a \rightarrow b$ must be set '0'

Using only the atoms $a \xrightarrow{+} b$ and $a \xrightarrow{-} b$, we define the notion of move on a system :

Definition 3.2 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system. Let $m = \{a \xrightarrow{x} b \mid x \in \{+, -\}\}$ be a set of atoms. Then, m is called* **move on** *AS iff $\forall (a \xrightarrow{+} b) \in m$, $(a,b) \in \overset{+}{\rightsquigarrow}$, and $\forall (a \xrightarrow{-} b) \in m$, $(a,b) \in \overset{-}{\rightsquigarrow}$.*

Definition 3.3 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, and let m be a move on AS. We define the* **resulting system of playing move** *m* **on** *AS as the argumentation system $\Delta(AS, m) = \langle A, \rightarrow_m, \overset{+}{\rightsquigarrow}_m, \overset{-}{\rightsquigarrow}_m \rangle$, such that :*

(1) $(a,b) \in \rightarrow_m$ *iff either* $(a,b) \in \rightarrow$ *and* $(a \xrightarrow{-} b) \notin m$*, or* $(a \xrightarrow{+} b) \in m$.

(2) $(a,b) \in \overset{+}{\rightsquigarrow}_m$ *iff either* $(a,b) \in \overset{+}{\rightsquigarrow}$ *and* $(a \xrightarrow{+} b) \notin m$*, or* $(a \xrightarrow{-} b) \in m$.

(3) $(a,b) \in \overset{-}{\rightsquigarrow}_m$ *iff either* $(a,b) \in \overset{-}{\rightsquigarrow}$ *and* $(a \xrightarrow{-} b) \notin m$*, or* $(a \xrightarrow{+} b) \in m$.

If we are able to play a move on $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$, we may be motivated to play it by the desire to satisfy a specific goal. Let us formally define this notion of goal.

Definition 3.4 *Let Systems denote a set of argumentation systems. Let Props denote a set of properties, such that each property can refer to any AS* $\in$ *Systems. We define the function* f *:* $Props \times Systems \rightarrow \{true, false\}$*, such that* $\forall P \in Props, \forall AS \in Systems$*, it holds that* $f(P,AS) = true$ *iff property P, when referring to system AS, holds; otherwise* $f(P,AS) = false$.
A property P may be chosen as a **positive goal**. *In that case, we say that goal P is satisfied in AS iff* $f(P,AS) = true$. *Also, a negated property* $\neg P$ *may be chosen as a* **negative goal**. *In that case, we say that goal* $\neg P$ *is satisfied in AS iff* $f(P,AS) = false$ *(that is iff* $f(\neg P,AS) = true$*).*

If we adopt a specific goal (either positive or negative) which is not satisfied in *AS*, then we search for possible moves on *AS*, leading to a modified system *AS*′ in which that goal is satisfied. Any move on *AS* which achieves this, is a successful move. That move is also a target set if the changes it makes on the initial system are minimal.

Definition 3.5 *Let* $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ *be an argumentation system, and Props be a set of properties. Let m be a move on AS, and* $P \in Props$ *be a property. Let g be a goal, such that g is P or* $\neg P$.
m is called **successful move for goal** *g iff goal g is satisfied in* $\Delta(AS,m)$*, that is if* $f(g, \Delta(AS,m)) = true$.
m is called **target set for goal** *g iff m is minimal w.r.t.* $\subseteq$ *among all the successful moves for g.*

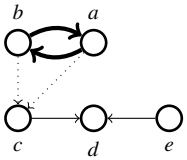
Let us now describe the types of goals (positive and negative) that we focus on. In the following, let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, and let *m* be a move on *AS*. We will focus on the acceptance of a single argument $d \in A$ called the *issue*. Also, let $X \in \{\exists, \forall\}$, and let $S \in Sem$ (we remind that $Sem = \{Adm, Pref, Comp\}$).

(1) Let $S_X(d)$ be a positive goal. We define :
- The set of successful moves $\mathbb{M}^S_X = \{m \mid S_X(d)$ is satisfied in $\Delta(AS,m)\}$.
- The set of target sets $\mathbb{T}^S_X = \{m \mid m \in \mathbb{M}^S_X$, and m is minimal w.r.t. $\subseteq$ among the elements of $\mathbb{M}^S_X\}$

(2) Let $\neg S_X(d)$ be a negative goal. We define :
- The set of successful moves $\mathbb{M}^S_{\neg X} = \{m \mid \neg S_X(d)$ is satisfied in $\Delta(AS,m)\}$.
- The set of target sets $\mathbb{T}^S_{\neg X} = \{m \mid m \in \mathbb{M}^S_{\neg X}$, and m is minimal w.r.t. $\subseteq$ among the elements of $\mathbb{M}^S_{\neg X}\}$

The following example aims at clarifying these notions.

Example 1 *Let* $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ *be an argumentation system such that* $A = \{a,b,c,d,e\}$, $\rightarrow = \{(a,b),(b,a),(c,d),(e,d)\}$, $\overset{+}{\rightsquigarrow} = \{(a,c),(b,c)\}$, $\overset{-}{\rightsquigarrow} = \{(c,d),(e,d)\}$*, and* $d \in A$ *is the issue.*



AS

Non-removable attacks are represented by thick arrows, removable attacks by simple arrows, and addable attacks by dotted arrows. Argument d does not belong to any admissible extension of AS. If our goal is to put d in some admissible (or preferred, or complete) extension, then there are three target sets for this goal : $\mathbb{T}^S_\exists = \{\{e \xrightarrow{-} d, c \xrightarrow{-} d\}, \{e \xrightarrow{-} d, a \xrightarrow{+} c\}, \{e \xrightarrow{-} d, b \xrightarrow{+} c\}\}$. *Also,* $\{e \xrightarrow{-} d, b \xrightarrow{+} c, a \xrightarrow{+} c\} \in \mathbb{M}^S_\exists$*, because this move is successful for the previous goal, though it is not a target set, because it is not minimal. Now, regarding sceptical preferred semantics, it holds that* $\mathbb{T}^{Pref}_\forall = \{\{e \xrightarrow{-} d, c \xrightarrow{-} d\}, \{e \xrightarrow{-} d, b \xrightarrow{+} c, a \xrightarrow{+} c\}\}$. *Finally, as far as grounded semantics is concerned,* $\mathbb{T}^{Comp}_\forall = \{\{e \xrightarrow{-} d, c \xrightarrow{-} d\}\}$.

We now provide some properties of the sets of succesful moves and of the sets of target sets.

Property 3 *It holds that*

$$\mathbb{M}^{Comp}_\forall \subseteq \mathbb{M}^{Pref}_\forall \subseteq \mathbb{M}^S_\exists$$

$$\mathbb{M}^S_{\neg\exists} \subseteq \mathbb{M}^{Pref}_{\neg\forall} \subseteq \mathbb{M}^{Comp}_{\neg\forall}$$

Proof 3 *Let us prove the relation regarding the positive goals. If move* $m \in \mathbb{M}^{Comp}_\forall$*, then d is accepted in* $AS' = \Delta(AS,m)$ *under complete semantics (using sceptical acceptability), so d belongs in all the complete extensions of AS*′*, therefore in all the preferred extensions of AS*′*. So, it holds that* $m \in \mathbb{M}^{Pref}_\forall$*. Thus, we have proved that* $\mathbb{M}^{Comp}_\forall \subseteq \mathbb{M}^{Pref}_\forall$*. Moreover, if* $m \in \mathbb{M}^{Pref}_\forall$*, then d belongs in all the preferred extensions of AS*′*, therefore d belongs in at least one preferred extension of AS*′ *(so, it also belongs in at least one admissible, and in at least one complete extension of AS*′*). Thus, it holds that* $m \in \mathbb{M}^S_\exists$*, and we have proved that*

$\mathbb{M}_{\forall}^{Pref} \subseteq \mathbb{M}_{\exists}^{S}$. *As a result,* $\mathbb{M}_{\forall}^{Comp} \subseteq \mathbb{M}_{\forall}^{Pref} \subseteq \mathbb{M}_{\exists}^{S}$. *Similarly, we can prove the second relation, regarding the negative goals.*

On the other hand, if we consider the corresponding target sets, the relations $\mathbb{T}_{\forall}^{Comp} \subseteq \mathbb{T}_{\forall}^{Pref}$, $\mathbb{T}_{\forall}^{Pref} \subseteq \mathbb{T}_{\exists}^{S}$ and $\mathbb{T}_{\forall}^{Comp} \subseteq \mathbb{T}_{\exists}^{S}$, do not hold in the general case.

Ex. 1, cont. *In this example, the relation* $\mathbb{T}_{\forall}^{Pref} \subseteq \mathbb{T}_{\exists}^{S}$ *does not hold. Small modifications of this example can show that, in the general case, neither* $\mathbb{T}_{\forall}^{Comp} \subseteq \mathbb{T}_{\forall}^{Pref}$ *nor* $\mathbb{T}_{\forall}^{Comp} \subseteq \mathbb{T}_{\exists}^{S}$ *hold.*

Similarly, for the negative goals, there are examples (omitted due to the lack of space) showing that the relations $\mathbb{T}_{\neg\exists}^{S} \subseteq \mathbb{T}_{\neg\forall}^{Pref}$, $\mathbb{T}_{\neg\forall}^{Pref} \subseteq \mathbb{T}_{\neg\forall}^{Comp}$ and $\mathbb{T}_{\neg\exists}^{S} \subseteq \mathbb{T}_{\neg\forall}^{Comp}$ do not hold in the general case.

Next, we highlight a link between the sets of target sets.

Property 4 *If m is a move such that* $m \in \mathbb{T}_{\forall}^{Comp}$ *and* $m \in \mathbb{T}_{\exists}^{S}$, *then* $m \in \mathbb{T}_{\forall}^{Pref}$ *(1)*
Moreover, if m is a move such that $m \in \mathbb{T}_{\neg\exists}^{S}$ *and* $m \in \mathbb{T}_{\neg\forall}^{Comp}$, *then* $m \in \mathbb{T}_{\neg\forall}^{Pref}$ *(2)*

Proof 4 *(1) By contradiction, let* $m \in \mathbb{T}_{\forall}^{Comp}$, $m \in \mathbb{T}_{\exists}^{S}$ *and assume that* $m \notin \mathbb{T}_{\forall}^{Pref}$. *Now,* $m \in \mathbb{T}_{\forall}^{Comp}$ *implies that* $m \in \mathbb{M}_{\forall}^{Comp}$ *(as m is minimal w.r.t.* $\subseteq$ *among the moves in* $\mathbb{M}_{\forall}^{Comp}$*). Then, from* $m \in \mathbb{M}_{\forall}^{Comp}$ *it follows that* $m \in \mathbb{M}_{\forall}^{Pref}$ *(from Property* 3*). Moreover, we assumed that* $m \notin \mathbb{T}_{\forall}^{Pref}$, *so there must exist another move* $m' \subset m$, *such that* $m' \in \mathbb{T}_{\forall}^{Pref}$ *(and, of course,* $m' \in \mathbb{M}_{\forall}^{Pref}$*). From* $m' \in \mathbb{M}_{\forall}^{Pref}$, *we get that* $m' \in \mathbb{M}_{\exists}^{S}$ *(from Property* 3*). Finally, from* $m' \in \mathbb{M}_{\exists}^{S}$ *and* $m \in \mathbb{T}_{\exists}^{S}$, *it follows that* $m \subseteq m'$. *Contradiction, since above we had* $m' \subset m$. *Therefore,* $m \in \mathbb{T}_{\forall}^{Pref}$.
We can prove (2) in a similar way.

Now that we have given some properties of the successful moves and of the target sets, we can define our rewriting procedure which shall allow us to compute all the target sets for some types of goals.

4 Rewriting Rules and Procedure

In this section we define a rewriting procedure which computes all the target sets for some types of goals. The procedure does this in a goal-driven manner, as it concentrates on the "relevant" attacks for a given goal. It starts from the issue and, going backwards, considers the attacks against the issue, then the attacks against the previous attacking arguments and so on.

The notion of atom is the basis of the procedure. The connectors $\wedge$ and $\vee$ are used in order to link atoms, forming conjuncts and formulas.

Definition 4.1
$Conjunct ::= Atom \mid (Conjunct \wedge Conjunct)$
$Formula ::= Conjunct \mid (Conjunct \vee Conjunct)$
Let Conjuncts denote the set of all possible conjuncts, and let Formulas denote the set of all possible formulas.
A conjunct which contains at least one open atom is called ***open conjunct****. Otherwise, it is called* ***closed conjunct****.*
A formula which contains at least one open conjunct is called ***open formula****. Otherwise, it is called* ***closed formula****.*

Having given the definitions of atom, conjunct and formula, we now proceed to the definition of some rewriting rules. There are two types of rewriting rules, *atom expansions* and *atom simplifications*. An atom expansion replaces a conjunct which features an open atom, by the disjunction of some new conjuncts. An atom simplification replaces two (or more) atoms in a conjunct by a single atom. Rewriting rules are indicated by the the ' :' sign.

Definition 4.2 *Let* $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ *be an argumentation system, with* $a, b \in A$. *Let p, q and r be conjunctions of (zero or more) atoms, and let* $x, y \in \{1, ?, 0\}$. *Finally, we denote by* $\mathcal{P}(X)$ *the powerset of a set X. The rewriting rules are presented in Figure 2.*

Let us briefly explain the intuition behind these rules. In the following, instead of saying that an attack is successful or unsuccessful, we say that it is '1', or '?', or '0'. The expansion rule for $PRO(a)$ states that, if there are no attacks against a, then we have no changes to make. But, if there are attacks against a, we must remove a subset of them, and make the rest of the attacks against it '0'. Notice that, if $(x,a) \in \overset{+}{\rightsquigarrow}$, then we introduce the atom $x \overset{0}{\rightarrow} a$ which is used as a type of marker. If we try to add this attack in the future, then this marker will cause the firing of a simplification rule (as we will see below). The expansion rule for $CON(a)$ states that there must be one attack against a (either already in the system, or added at this point) which is either '1' or '?'. The expansion rule for $a \overset{1}{\twoheadrightarrow} b$, states that in order for this attack to be '1', a subset of the attacks against it must be removed, and the rest must be '0'. The expansion rule for $a \overset{0}{\twoheadrightarrow} b$, states that in order for this attack to be '0', one attack (either already in the system, or added in it) must be '1'. Finally, the expansion rule for $a \overset{?}{\twoheadrightarrow} b$ is the longest, but the intuition behind it is straightforward. It states that in order for an attack to be '?', the following must hold : Firstly, a subset of its incoming attacks (either already in AS, or added at this point) must be '?' (notice that this set cannot be empty). Secondly, the rest of its incoming attacks must be either removed (the attacks in the set S') or they must be '0'. Thirdly, notice that if $(x,a) \in \overset{+}{\rightsquigarrow}$, then the atom $x \overset{0}{\rightarrow} a$ is used again as a marker, preventing the addition of a '1' attack against a in the future.

Atom expansions :

(1) $PRO(a) : \top$, if $\{(x,a)|(x,a) \in \rightarrow\} = \{\}$
$\bigvee_{S \in \mathcal{P}(\{(x,a)|(x,a) \in \overset{-}{\rightsquigarrow}\})} (\bigwedge_{(x,a) \in S}((x \overset{-}{\rightarrow} a) \wedge (x \overset{0}{\rightarrow} a)) \wedge \bigwedge_{(x,a) \in (\rightarrow \setminus S)} (x \overset{0}{\twoheadrightarrow} a) \wedge \bigwedge_{(x,a) \in \overset{+}{\rightsquigarrow}} (x \overset{0}{\rightarrow} a))$, otherwise.

(2) $CON(a) : \bot$, if $\{(x,a)|(x,a) \in (\rightarrow \cup \overset{+}{\rightsquigarrow})\} = \{\}$
$\bigvee_{(x,a) \in \rightarrow} ((x \overset{1}{\twoheadrightarrow} a) \vee (x \overset{?}{\twoheadrightarrow} a)) \vee \bigvee_{(x,a) \in \overset{+}{\rightsquigarrow}} (((x \overset{+}{\rightarrow} a) \wedge (x \overset{1}{\twoheadrightarrow} a)) \vee ((x \overset{+}{\rightarrow} a) \wedge (x \overset{?}{\twoheadrightarrow} a)))$, otherwise.

(3) $p \wedge (a \overset{1}{\twoheadrightarrow} b) \wedge q : \bigvee_{S \in \mathcal{P}(\{(x,a)|(x,a) \in \overset{-}{\rightsquigarrow}\})} (p \wedge (a \overset{1}{\rightarrow} b) \wedge q \wedge \bigwedge_{(x,a) \in S}((x \overset{-}{\rightarrow} a) \wedge (x \overset{0}{\rightarrow} a)) \wedge \bigwedge_{(x,a) \in (\rightarrow \setminus S)} (x \overset{0}{\twoheadrightarrow} a) \wedge \bigwedge_{(x,a) \in \overset{+}{\rightsquigarrow}} (x \overset{0}{\rightarrow} a))$

(4) $p \wedge (a \overset{0}{\twoheadrightarrow} b) \wedge q : \bot$, if $\{(x,a)|(x,a) \in (\rightarrow \cup \overset{+}{\rightsquigarrow})\} = \{\}$
$\bigvee_{(x,a) \in \rightarrow} (p \wedge (a \overset{0}{\rightarrow} b) \wedge q \wedge (x \overset{1}{\twoheadrightarrow} a)) \vee \bigvee_{(x,a) \in \overset{+}{\rightsquigarrow}} (p \wedge (a \overset{0}{\rightarrow} b) \wedge q \wedge (x \overset{+}{\rightarrow} a) \wedge (x \overset{1}{\twoheadrightarrow} a))$, otherwise.

(5) $p \wedge (a \overset{?}{\twoheadrightarrow} b) \wedge q : \bot$, if $\{(x,a)|(x,a) \in (\rightarrow \cup \overset{+}{\rightsquigarrow})\} = \{\}$
$\bigvee_{S \in \mathcal{P}(\{(x,a)|(x,a) \in (\rightarrow \cup \overset{+}{\rightsquigarrow})\}), S \neq \{\}} \bigvee_{S' \in \mathcal{P}(\{(x,a)|(x,a) \in (\overset{-}{\rightsquigarrow} \setminus S)\})} (p \wedge (a \overset{?}{\rightarrow} b) \wedge q \wedge \bigwedge_{(x,a) \in (\rightarrow \cap S)} (x \overset{?}{\twoheadrightarrow} a) \wedge \bigwedge_{(x,a) \in (\overset{+}{\rightsquigarrow} \cap S)} ((x \overset{+}{\rightarrow} a) \wedge (x \overset{?}{\twoheadrightarrow} a)) \wedge \bigwedge_{(x,a) \in S'} ((x \overset{-}{\rightarrow} a) \wedge (x \overset{0}{\rightarrow} a)) \wedge \bigwedge_{(x,a) \in (\rightarrow \setminus (S \cup S'))} (x \overset{0}{\twoheadrightarrow} a) \wedge \bigwedge_{(x,a) \in (\overset{+}{\rightsquigarrow} \setminus S)} (x \overset{0}{\rightarrow} a))$, otherwise.

Atom simplifications :

(1)	(a) $p \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge q \wedge (a \overset{x}{\rightarrow} b) \wedge r : p \wedge q \wedge r \wedge (a \overset{x}{\rightarrow} b)$	(2)	$p \wedge (a \overset{x}{\rightarrow} b) \wedge q \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge r : p \wedge q \wedge r \wedge (a \overset{x}{\twoheadrightarrow} b)$
	(b) $p \wedge (a \overset{x}{\rightarrow} b) \wedge q \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge r : p \wedge q \wedge r \wedge (a \overset{x}{\rightarrow} b)$	(4)	$p \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge q \wedge (a \overset{y}{\twoheadrightarrow} b) \wedge r : \bot$, if $x \neq y$
(3)	(a) $p \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge q \wedge (a \overset{y}{\rightarrow} b) \wedge r : \bot$, if $x \neq y$	(5)	$p \wedge (a \overset{x}{\rightarrow} b) \wedge q \wedge (a \overset{y}{\rightarrow} b) \wedge r : \bot$, if $x \neq y$
	(b) $p \wedge (a \overset{y}{\rightarrow} b) \wedge q \wedge (a \overset{x}{\twoheadrightarrow} b) \wedge r : \bot$, if $x \neq y$	(6)	$p \wedge \bot \wedge r : \bot$

FIGURE 2 – Rewriting rules

Now, as far as the simplification rules are concerned : Rules 1 and 2 are used to ensure that each attack is "treated" only once (and they ensure the termination of the procedure). Rules 3, 4 and 5 are fired when some attack is set to be both '1' and '0' (or '?' and '1', or finally '?' and '0'), at the same time. In that case, the $\bot$ atom is introduced. Rule 6 states that the $\bot$ atom removes all other atoms found in the same conjunct with it.

Let us now informally describe the rewriting procedure. It starts with a formula of the type $PRO(d)$, or $CON(d)$ and then uses the expansion rules to rewrite it, thus obtaining another formula with more (or at least the same number of) conjuncts. After the application of an expansion rule, there follows (possibly) a series of simplifications. The expansion and simplification steps are repeated, until the procedure computes a closed formula. Then, it returns a set of moves, one move for each conjunct of the closed formula.

Note that this kind of procedure is classical in rewriting systems, as for example in the system Maude [6].

Definition 4.3 *Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system, and let $d \in A$. The* **rewriting procedure** *RP takes as input a formula denoted $initF$, such that either $initF = PRO(d)$, or $initF = CON(d)$, and returns a set of moves denoted $\mathcal{M}_d$. More specifically, the set of returned moves is denoted $\mathcal{M}_d^{PRO}$ if $initF = PRO(d)$, and it is denoted $\mathcal{M}_d^{CON}$ if $initF = CON(d)$. The procedure's algorithm is Algorithm 1.*

Ex. 1, cont. *Let the input formula to the RP procedure be $initF = PRO(d)$. Let f_0 denote the initial formula $PRO(d)$, and f_{i+1} denote the formula we obtain after the application of an expansion rule on f_i. The procedure begins as follows :*
$f_0 = PRO(d)$, $f_1 = (e \overset{-}{\rightarrow} d \wedge e \overset{0}{\rightarrow} d \wedge c \overset{-}{\rightarrow} d \wedge c \overset{0}{\rightarrow} d) \vee$
$(e \overset{-}{\rightarrow} d \wedge e \overset{0}{\rightarrow} d \wedge c \overset{0}{\twoheadrightarrow} d) \vee (e \overset{0}{\twoheadrightarrow} d \wedge c \overset{-}{\rightarrow} d \wedge c \overset{0}{\rightarrow} d) \vee$
$(e \overset{0}{\twoheadrightarrow} d \wedge c \overset{0}{\twoheadrightarrow} d)$, $f_2 = \ldots$
Once RP computes a closed formula f_n, there are no more expansion rules to apply. Then, the procedure returns a move for each conjunct of f_n (unless that conjunct is $\bot$). In this example, RP returns the following set of three moves : $\mathcal{M}_d^{PRO} = \{\{e \overset{-}{\rightarrow} d, c \overset{-}{\rightarrow} d\}, \{e \overset{-}{\rightarrow} d, b \overset{+}{\rightarrow} c\}, \{e \overset{-}{\rightarrow} d, a \overset{+}{\rightarrow} c\}\}$.

A series of applications of expansion rules leading to a closed formula, can be illustrated as the gradual expansion of a tree, called "expansion-tree". The nodes of an expansion-tree are conjuncts. Initially, an expansion-tree

Data : An argumentation system $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$, $initF = PRO(d)$ or $initF = CON(d)$, $d \in A$.
Result : A set of moves $\mathcal{M}_d$.
Initialise formula $currF := initF$;
while *currF is an open formula* **do**
  Arbitrarily choose a conjunct $(p \wedge atExp \wedge q)$ of $currF$, which contains the open atom $atExp$;
  if $p \wedge atExp \wedge q : (p \wedge q \wedge r_1) \vee \cdots \vee (p \wedge q \wedge r_k)$ *is an atom expansion rule* **then**
    Update $currF$, by replacing $(p \wedge atExp \wedge q)$ with $(p \wedge q \wedge r_1) \vee \cdots \vee (p \wedge q \wedge r_k)$;
  end
  while *a simplification can be applied on some conjunct of currF* **do**
    Arbitrarily choose such a simplification, and apply it ;
  end
end
Initialise the set of moves $\mathcal{M}_d := \{\}$;
foreach *conjunct C of currF* **do**
  if $C \neq \perp$ **then**
    $m := \{a \overset{x}{\rightarrow} b \mid a \overset{x}{\rightarrow} b$ appears in C, and $x \in \{+, -\}\}$;
    Add m into the set $\mathcal{M}_d$;
  end
end
return $\mathcal{M}_d$;

Algorithm 1 : The rewriting procedure RP

has only one node-conjunct (which is $initF$). When an expansion rule is applied in a node-conjunct (the application is simply called "expansion") a number of child nodes are created for that node. This way the expansion-tree gets bigger after each expansion, until at some point all its leaves are closed conjuncts. Then, the procedure computes $\mathcal{M}_d$ from the leaves of the expansion-tree.

Now, let us highlight some properties that the *RP* procedure has, when it is coupled with the above expansion and simplification rules. First, *RP* always terminates (provided a finite number of arguments in the system we refer to). Second, for given input formula $initF$, *RP* always returns the same set of moves $\mathcal{M}_d$.

Property 5 *Termination of RP*
Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system. Then, the procedure RP always terminates.

Proof 5 *Let $f_{cl}, f_{op}, f_{at} : Conjuncts \rightarrow \mathbb{N}$, which, for every conjunct c, return the number of its a $\overset{x}{\rightarrow}$ b atoms (f_{cl}), the number of its a $\overset{x}{\twoheadrightarrow}$ b atoms (f_{op}), and such that $f_{at}(c) = f_{cl}(c) + f_{op}(c)$. In order to prove the termination of RP, we must prove that, after a finite number of expansions, a closed formula (without open atoms) is computed by RP. First, notice that for any conjunct c of any formula computed by RP, and $\forall (a,b) \in (\rightarrow \cup \overset{+}{\rightsquigarrow})$: If atom a $\overset{x}{\twoheadrightarrow}$ b is in c, then it is there only once (simplification rule 2); there is no atom a $\overset{y}{\twoheadrightarrow}$ b with $x \neq y$ in c (simplification rule 4); there is no atom a $\overset{x}{\rightarrow}$ b in c (simplification rule 1); and finally, there is no atom a $\overset{y}{\rightarrow}$ b with $x \neq y$ in c (simplification rule 3). Similarly, notice that if atom a $\overset{x}{\rightarrow}$ b is in c, then it is there only once (because simplification rule 1 would have removed any atom a $\overset{x}{\twoheadrightarrow}$ b, therefore no second atom a $\overset{x}{\rightarrow}$ b can appear in c); there is no atom a $\overset{y}{\rightarrow}$ b with $x \neq y$ in c (simplification rule 5); there is no atom a $\overset{x}{\twoheadrightarrow}$ b in c (simplification rule 1); and finally, there is no atom a $\overset{y}{\twoheadrightarrow}$ b with $x \neq y$ in c (simplification rule 3). Therefore, it holds that $f_{at}(c) \leq |(\rightarrow \cup \overset{+}{\rightsquigarrow})|$. From this inequality and from $f_{at}(c) = f_{cl}(c) + f_{op}(c)$, it follows that $f_{op}(c) \leq |(\rightarrow \cup \overset{+}{\rightsquigarrow})| - f_{cl}(c)$. Now, let $form_i$ be an open formula computed by RP and let c_i be an open conjunct of $form_i$ which appears after p expansions.* [1] *From the definition of the expansion and simplification rules, it follows that when an open atom of c_i is expanded, then each of the conjuncts which replace c_i, contains one more a $\overset{x}{\rightarrow}$ b atom than c_i (the only exception is the initial formula $PRO(d)$ or $CON(d)$, which is expanded without producing any a $\overset{x}{\rightarrow}$ b atoms). Therefore, it holds that $f_{cl}(c_i) = p - 1$. As a result, $f_{op}(c_i) \leq |(\rightarrow \cup \overset{+}{\rightsquigarrow})| - (p-1)$. Given that, in every expansion, a conjunct is replaced by a finite number of new conjuncts (this follows directly from the atom expansion rules and from the fact that $|A|$ is finite), by continuing the expansions, RP will eventually compute a formula, such that every conjunct it has is either closed, or it has been computed after $p = |(\rightarrow \cup \overset{+}{\rightsquigarrow})| + 1$ expansions. In the latter case, the conjunct is necessarily closed, because $f_{op}(c_i) \leq 0$, so c_i contains no open atoms. Once a closed formula is computed, the procedure RP returns one move for every conjunct of that formula (except for the $\perp$ conjuncts). Therefore, RP always terminates.*

Property 6 *Determinism of RP*
Let $AS = \langle A, \rightarrow, \overset{+}{\rightsquigarrow}, \overset{-}{\rightsquigarrow} \rangle$ be an argumentation system. For every input formula $initF$ (such that $initF = PRO(d)$ or $initF = CON(d)$ with $d \in A$), the RP procedure always returns the same set of moves $\mathcal{M}_d$.

Proof 6 *We shall just provide the sketch of this proof. In order to prove that RP is deterministic, it suffices to prove that if an arbitrary move can be computed by RP, then it is always computed by RP. More specifically, we prove that for any pair of expansion-trees t_1 and t_2 which are constructible by RP, for every leaf node (closed conjunct) of t_1, which is not $\perp$, there exists a leaf node of t_2 which has exactly the same a $\overset{+}{\rightarrow}$ b and a $\overset{-}{\rightarrow}$ b atoms. Now we explain how we*

1. It is helpful here to consider the corresponding expansion-tree, where the node-conjunct c_i is found in depth p.

proceed in order to prove the above. Given an arbitrary leaf of t_1, we denote p_1 the path which connects the root of t_1 to that leaf. By checking the expansion rules which were applied along p_1, we calculate as follows a second path p_2 connecting the root of t_2 to some node : we begin at the root of t_2, and we denote $atExp$ the atom expanded at this point. Next, we check if $atExp$ was expanded in p_1. If it was not, then we do nothing. But, if it was expanded, then we check the specific expansion rule applied in p_1 for $atExp$. Then, in t_2, we choose the edge corresponding to the same expansion rule which was applied in p_1, and we add that edge at the end of p_2. So, p_2 gets extended by one edge, and we end up to a child node of the node we expanded. As long as a new edge is added to p_2, we repeat the previous steps. Otherwise, we stop. At this point, we first prove that the last node of p_2 is necessarily a leaf of t_2. Then, we prove that every $a \xrightarrow{+} b$ and $a \xrightarrow{-} b$ atom appearing in the first leaf necessarily appears in the second leaf, and vice-versa. This proves that RP is determinist.

We now analyze the usefulness of the rewriting procedure wrt. the different argumentative goals. We shall say that :

- the procedure is *correct* for goal g if every move it returns is successful for g.
- the procedure is *complete* for goal g if it returns *all* the target sets for g.

Property 7 *Correctness and Completeness of RP*
The following table illustrates for which goals the rewriting procedure is correct and / or complete.

Goal	Correctness	Completeness
$S_\exists(d)$	*Yes*	*Yes*
$Pref_\forall(d)$	*No*	*No*
$Comp_\forall(d)$	*No*	*Yes*
$\neg S_\exists(d)$	*No*	*No*
$\neg Pref_\forall(d)$	*No*	*?*
$\neg Comp_\forall(d)$	*Yes*	*Yes*

Proof 7 Correctness of *RP* for $S_\exists(d)$, *that is $\mathcal{M}_d^{PRO} \subseteq \mathbb{M}_\exists^S$. Let $m \in \mathcal{M}_d^{PRO}$. We must prove that $m \in \mathbb{M}_\exists^S$. The move $m \in \mathcal{M}_d^{PRO}$ correponds to some conjunct, denoted c_m, of the closed formula computed by RP. From c_m we can construct the set of arguments $D = \{x \mid x \xrightarrow{1} y \text{ is an atom of } c_m\}$. We will now prove that in $\Delta(AS,m) = \langle A, \rightarrow_m, \overset{+}{\rightsquigarrow}_m, \overset{-}{\rightsquigarrow}_m \rangle$, it holds that D is an admissible set of arguments which defends argument d. First, let us assume that in $\Delta(AS,m)$ the set D is not conflict-free. In that case there exist two arguments $x_1, x_2 \in D$, such that $(x_1,x_2) \in \rightarrow_m$. Now, $x_1, x_2 \in D$ implies that $\exists x_3, x_4 \in A$ such that $x_1 \xrightarrow{1} x_3$ and $x_2 \xrightarrow{1} x_4$ are atoms of c_m. Given that $x_2 \xrightarrow{1} x_4$ appears in c_m, and that $(x_1,x_2) \in \rightarrow_m$, it follows that atom $x_1 \xrightarrow{0} x_2$ must also appear in c_m (because of expansion rule 3). In turn, this means that $\exists x_5 \in A$ such that $x_5 \xrightarrow{1} x_1$ also appears in c_m (because of expansion rule 4). Similarly, given that $x_1 \xrightarrow{1} x_3$ appears in c_m, it holds that $x_5 \xrightarrow{0} x_1$ also appears in c_m. But, it is impossible for both $x_5 \xrightarrow{1} x_1$ and $x_5 \xrightarrow{0} x_1$ to appear in the same conjunct (because simplification rules 3, 4 and 5 prevent this, by introducing the $\bot$ atom). Therefore, we have proved that D is conflict-free. Second, let us assume that in the system $\Delta(AS,m)$, the set D does not defend all its elements. In that case $\exists x_1 \in D$ and $\exists x_2 \notin D$ such that $(x_2,x_1) \in \rightarrow_m$, and no argument of D attacks x_2. The fact that $x_1 \in D$ implies that $\exists x_0 \in A$ such that atom $x_1 \xrightarrow{1} x_0$ appears in c_m. So, from expansion rule 3, atom $x_2 \xrightarrow{0} x_1$ also appears in c_m, and as a result, $\exists x_3 \in A$ such that atom $x_3 \xrightarrow{1} x_2$ also appears in c_m (otherwise, expansion rule 4 would have introduced atom $\bot$ in c_m). By definition of the set D, notice that $x_3 \in D$. Impossible, since we assumed that no argument of D attacks x_2 in $\Delta(AS,m)$. Therefore, we have proved that D defends all its elements. Given that D is conflict-free and it defends all its elements, it follows that D is an admissible set of arguments. Finally, since for every attack $(x,d) \in \rightarrow_m$ against the issue d, atom $x \xrightarrow{0} d$ appears in c_m (because of expansion rule 1), it holds that argument d is defended by the set D. From this, and from the fact that D is admissible in $\Delta(AS,m)$, it follows that also $D \cup \{d\}$ is admissible in $\Delta(AS,m)$. So, it holds that $m \in \mathbb{M}_\exists^S$, and we have finally proved that $\mathcal{M}_d^{PRO} \subseteq \mathbb{M}_\exists^S$.*

Correctness of *RP* for $\neg Comp_\forall(d)$ (sketch of proof). *Let $m \in \mathcal{M}_d^{CON}$. We must prove that $m \in \mathbb{M}_{\neg\forall}^{Comp}$. The move m correponds to some conjunct, denoted c_m, of the closed formula computed by RP. From the expansion rule for $CON(d)$, it follows that c_m either contains an atom $x \xrightarrow{1} d$, or it contains an atom $x \xrightarrow{?} d$. First, if c_m contains an atom $x \xrightarrow{1} d$, then there exists some admissible extension of $\Delta(AS,m)$ which contains argument x. So, d does not belong to the grounded extension of $\Delta(AS,m)$, in other words $m \in \mathbb{M}_{\neg\forall}^{Comp}$. Second, if c_m contains an atom $x \xrightarrow{?} d$, then in the system $\Delta(AS,m)$ there exists a cycle of attacks and a path of attacks coming from the cycle and leading to d. This follows from the expansion rules 4 and 5, because when an atom $x \overset{?}{\twoheadrightarrow} y$ is expanded, then an atom $z \overset{?}{\twoheadrightarrow} x$ is introduced, and the only way to finally obtain a conjunct other than $\bot$, is to create a cycle of '?' attacks. Additionnally, from expansion rule 5, it follows that every attack against an argument of that cycle (or against an argument of the path connected to that cycle) is set to be either '?' or '0'. In this case, it is straightforward to prove that there exists no argument of the grounded extension of $\Delta(AS,m)$ which attacks an argument of the cycle (or of the path). The fact that argument d is "connected" to such a cycle, implies that d does not belong to the grounded extension*

of $\Delta(AS,m)$. [2] *Therefore,* $m \in \mathbb{M}^{Comp}_{\neg\forall}$*, and we have proved that* $\mathcal{M}^{CON}_d \subseteq \mathbb{M}^{Comp}_{\neg\forall}$.

Completeness of *RP* for $S_\exists(d)$ (sketch of proof). *Note that the proof for completeness for* $Comp_\forall(d)$ *is almost identical. The proof of completeness for* $\neg Comp_\forall(d)$ *is omitted. So, we want to prove that* $\mathbb{T}^S_\exists \subseteq \mathcal{M}^{PRO}_d$*. Let* t *be a target set such that* $t \in \mathbb{T}^S_\exists$*. We shall prove that there exists an expansion-tree, constructible by the RP procedure, which has a leaf node containing all the atoms of* t*, and no additional* $x \xrightarrow{+} y$*, or* $x \xrightarrow{-} y$ *atoms. If this is the case, and given that RP is deterministic, it follows that RP will always compute* t*. Let* $X = \{x_1 \xrightarrow{-} d, \ldots, x_n \xrightarrow{-} d\}$ *be a set of atoms, such that* $\{x_1, \ldots, x_n\}$ *are all the arguments attacking* d *in the initial system. Naturally,* t *contains a subset of the atoms in* X*, denoted* $X' \subseteq X$*. On the other hand, we can prove that* t *cannot contain any atoms of the form* $x_i \xrightarrow{+} d$*. At this point, notice that the RP procedure expands the root node (the conjunct* $PRO(d)$*) and creates a number of child nodes. Each child node has a different subset of* X *(because of the expansion rule for* $PRO(d)$*). Therefore, there exists exactly one child node of the root which contains exactly the atoms of* X'*. Let us denote that node n. Let* $\{x_k, \ldots, x_l\}$ *denote the subset of arguments whose attacks against* d *remain in* $\Delta(AS,t)$*. Then, according to the expansion rule for* $PRO(d)$*, the node n also contains the atoms* $x_k \overset{0}{\twoheadrightarrow} d$*, ...,* $x_l \overset{0}{\twoheadrightarrow} d$*. Since* t *contains the atoms of* X'*, plus some additional atoms (resulting from further expansions of* $x_k \overset{0}{\twoheadrightarrow} d$*, ...,* $x_l \overset{0}{\twoheadrightarrow} d$*), it can be denoted* $t = X' \cup Z$*. Notice that every atom of* Z *refers to an attack necessarily "connected" to some argument of the set* $\{x_k, \ldots, x_l\}$*. Lets focus on the attacks against* d *which are not removed. Those attacking arguments must get attacked back, in order for* d *to be reinstated. At this point, it is not difficult to prove that : (A) It is impossible for any* $y \xrightarrow{-} x_i$ *atom to appear in* t*. (B) It is impossible for 2 or more atoms* $y_1 \xrightarrow{+} x_i$*,* $y_2 \xrightarrow{+} x_i$ *to appear in* t*. As a result, for every argument* $x_i \in \{x_k, \ldots, x_l\}$ *which atacks* d*,* t *can only contain 0 or 1 atoms of the type* $y \xrightarrow{+} x_i$*. We must make sure that RP computes all the possible combinations of attacks reinstating* d*. Consider that the RP expands node n, by choosing and expanding first* $x_k \overset{0}{\twoheadrightarrow} d$*, and then, in every child node of n, it chooses and expands the next atom* $x_{k+1} \overset{0}{\twoheadrightarrow} d$*. It continues this way, until it finally expands atom* $x_l \overset{0}{\twoheadrightarrow} d$ *in every leaf node below n. After all these expansions, there is a subtree created below node n, whose every leaf (these leaves are not necessarily closed), contains either 0 or 1 atoms of the type* $y \xrightarrow{+} x_i$ *for every* $x_i \in \{x_k, \ldots, x_l\}$*. In other words, the procedure has created a node for every possible combination of attacks which can reinstate* d*. Therefore, there exists (exactly one) leaf-node of that subtree, which contains exactly the atoms of* t *which add attacks against the arguments of* $\{x_k, \ldots, x_l\}$*. Moreover, every such node, if it contains atom* $y \xrightarrow{+} x_i$*, then it also contains atom* $y \overset{1}{\twoheadrightarrow} x_i$ *(because the expansion rules demand that the added attacks must be '*1*'). By continuing to search the graph backwards, we consider the indirect attackers (and defenders) of* d*. The procedure continues, similarly to the expansion of atom* $PRO(d)$*, and uses the expansion rules for the* $y \overset{1}{\twoheadrightarrow} x_i$ *atoms. Therefore, after a finite number of expansions, the procedure eventually computes a node which contains exacly the combination of atoms in* t*. The last thing we must verify, is that the simplification and expansion rules producing* $\bot$*, cannot lead to the "loss of a target set". This means that if a node is simplified into* $\bot$*, then we must prove that it would have been impossible for a target set to appear in the subtree below that node. We begin with the only expansion rule which can add the* $\bot$ *atom : If there is a node n which has the atom* $x \overset{0}{\twoheadrightarrow} y$*, and there is no potential attacker of* x *in the system, then the expansion rule for* $x \overset{0}{\twoheadrightarrow} y$ *adds the* $\bot$ *atom. Having the* $x \overset{0}{\twoheadrightarrow} y$ *atom in node n, means that all the target sets found in the subtree below node n, must lead to a modified system where there exists an argument attacking* x*. Since* x *has no potential attackers, this can never happen, therefore a* $\bot$ *leaf can be added, without any loss of target sets. And now the simplification rules which may give a* $\bot$ *leaf : If there is a node n containing an atom of the type* $x \overset{0}{\twoheadrightarrow} y$*, or* $x \xrightarrow{0} y$*, or* $x \overset{1}{\twoheadrightarrow} y$*, or* $x \xrightarrow{1} y$*, as well as an atom with the opposite number (0/1), then a simplification rule is fired and adds the* $\bot$ *atom. Let us see why we can never "lose" a target set by this type of simplification. We consider the case where we have a node with both* $x \overset{0}{\twoheadrightarrow} y$ *and* $x \overset{1}{\twoheadrightarrow} y$ *(the other cases are similar). The existence of both* $x \overset{0}{\twoheadrightarrow} y$ *and* $x \overset{1}{\twoheadrightarrow} y$ *in the same node n, means that every eventual target set found in the subtree below node n, leads to a modified system in which some admissibe extension : (a) attacks argument* x *(because of atom* $x \overset{0}{\twoheadrightarrow} y$*), and (b) contains argument* x *(because of atom* $x \overset{1}{\twoheadrightarrow} y$*). Of course this is impossible, as every admissible extension is conflict-free. Therefore, no target set could be found in that subtree, thus we can add the atom* $\bot$*, without any losing of target sets.*

The fact that *RP* is complete for the goals $S_\exists(d)$, $Comp_\forall(d)$ and $\neg Comp_\forall(d)$ is arguably one of its main advantages. Here we note that for all the above goals where *RP* is not correct (or not complete) there exist counter-examples, which were omitted due to the lack of space. The sole exception is the completeness for goal $\neg Pref_\forall(d)$, which remains open so far. We will just provide a counter-example showing

2. In other words, if we used argument labelling, we would say that attributing the *undec* label to all the arguments of that cycle and of that path, is part of a valid labelling.

that the relation $\mathbb{T}_{\forall}^{Pref} \subseteq \mathcal{M}_d^{PRO}$ (completeness of *RP* for goal $Pref_{\forall}(d)$) does not hold in the general case.

Ex. 1, cont. *In this example, as we have already seen, it holds that* $\{e \xrightarrow{-} d, b \xrightarrow{+} c, a \xrightarrow{+} c\} \in \mathbb{T}_{\forall}^{Pref}$*, but* $\{e \xrightarrow{-} d, b \xrightarrow{+} c, a \xrightarrow{+} c\} \notin \mathcal{M}_d^{PRO}$*. Therefore,* $\mathbb{T}_{\forall}^{Pref} \subseteq \mathcal{M}_d^{PRO}$ *does not hold in the general case.*

5 Conclusion

The dynamics of argumentation systems is a central and compelling notion to address when debates are to be considered among users or agents. However, the task of computing which move to make in order to reach a given argumentative goal is difficult. In this paper we focus on complex simultaneous moves involving addition and retraction of attacks. We first proved a number of results related to the relation which holds among sets of successful moves. Then we described an approach based on a dedicated rewriting procedure, and proposed rules inspired from the attack semantics [10]. This approach has the advantage of being relatively easy to design and interpret. This is an important feature if we consider a context where such moves are suggested to a user, since for instance traces can provide human-readable explanations of the result of the procedure. We then presented a number of results regarding the correctness and completeness of the procedure : regarding positive goals, it is correct and complete for the credulous version of the semantics discussed in this paper ; while it is complete for the sceptical version of the complete semantics, regardless of the type of goal considered.

In order to implement the rewriting rules, we used the Maude programming language [3] [6]. Maude is a high-level, based on rewriting logic, declarative language which can model systems. In fact a program in Maude is a logical theory, and a computation made by that program is a logical deduction using the axioms of the theory. Given its declarative nature, our program was compact and relatively easy to understand. Also, the way the Maude system operates and performs the rewritings corresponds to the way that the *RP* procedure works. These advantages were essential for choosing Maude for the implementation.

The efficiency of our procedure requires further investigation. Interestingly, the work of [1] already mentioned in the introduction may prove useful in that respect : first note that a distance minimal expansion must be $\subseteq$-minimal. Furthermore, as our expansions are more constrained (not all attacks can be added or removed), a minimal value for their problem remains a minimal value for ours. Taken together, this means that their lower bounds hold in our setting. Thus, if the value function of [1] (which provides means to compute simply such value) gives a value k, then we know for sure that no target set of size $< k$ can be found. This could be exploited in further developments of the rewriting procedure to make it more efficient.

3. http ://maude.cs.uiuc.edu

Références

[1] Ringo Baumann. What does it take to enforce an argument ? minimal change in abstract argumentation. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 127–132. IOS Press, 2012.

[2] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks : Enforcing and monotonicity results. In Pietro Baroni, Federico Cerutti, Massimiliano Giacomin, and Guillermo Ricardo Simari, editors, *COMMA*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, pages 75–86. IOS Press, 2010.

[3] Guido Boella, Dov Gabbay, Alan Perotti, Leendert van der Torre, and Serena Villata. Conditional labelling for abstract argumentation. In *Proc. of TAFA'11*, 2011.

[4] Martin Caminada. On the issue of reinstatement in argumentation. In *JELIA'06*, pages 111–123, 2006.

[5] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in abstract argumentation frameworks : Adding an argument. *J. Artif. Intell. Res. (JAIR)*, 38 :49–84, 2010.

[6] Manuel Clavel, Fransisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Jose F. Quesada. The maude system. In *Proceedings of the 10th International Conference on Rewriting Techniques and Applications*, RtA '99, pages 240–243, London, UK, UK, 1999. Springer-Verlag.

[7] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-persons games. *Artificial Intelligence*, 77 :321–357, 1995.

[8] Hadassa Jakobovits and Dirk Vermeir. Robust semantics for argumentation frameworks. *J. Log. Comput.*, 9(2) :215–261, 1999.

[9] Sanjay Modgil and Martin Caminada. Proof theories and algorithms for abstract argumentation frameworks. In Guillermo Simari and Iyad Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 105–129. Springer US, 2009.

[10] Serena Villata, Guido Boella, and Leendert van der Torre. Attack semantics for abstract argumentation. In Toby Walsh, editor, *IJCAI*, pages 406–413. IJCAI/AAAI, 2011.