Guillermo Ricardo Simari Paolo Torroni (eds.)

Argumentation and Non-Monotonic Reasoning

An LPNMR Workshop

Tempe, Arizona, US, May 14, 2007
Proceedings

ArgNMR Home Page:
http://lia.deis.unibo.it/confs/argnmr/

Preface

Research on argumentation and non-monotonic reasoning began in full force in the early eighties. The first attempts showed how argumentation results in a very natural way of conceptualizing commonsense reasoning, appropriately reflecting its defeasible nature. Further work in the knowledge representation and reasoning community has shown that argumentation provides a useful perspective for relating different non-monotonic formalisms. More recently, argumentation has been revealed as a powerful conceptual tool for exploring the theoretical foundations of reasoning and interaction in autonomous agents and multiagent systems.

This volume contains the papers that will be presented at the First International Workshop on Argumentation and Non-Monotonic Reasoning (ArgNMR 2007) on May 14, 2007 in Tempe, Arizona, US. Each submission was reviewed by at least 3 programme committee members.

ArgNMR will consist of 9 presentations and 2 discussion sessions. Our intention is to propose this event as an opportunity for exchanging ideas on the fundamental theoretical basis and the design and implementation of argument-based systems including semantics, proof theory, applications and the comparison of those systems with other types of non-monotonic reasoning.

We wish to thank the authors of this volume, the ArgNMR Programme Committee, the delegates who will attend, Chitta Baral for providing logistic support, Gerd Brewka and John Schlipf for inviting us to propose such an event in co-location with LPNMR, and Andrei Voronkov for his contribution to ArgNMR through EasyChair.

April 2007

Guillermo Ricardo Simari
Paolo Torroni

Workshop Organization

Programme Chairs

Guillermo Ricardo Simari, *U Nacional del Sur, Bahía Blanca, Argentina*
Paolo Torroni, *U Bologna, Italy*

Programme Committee

Leila Amgoud, *IRIT-CNRS Toulouse, France*
Grigoris Antoniou, *FORTH-ICS, Greece*
Pietro Baroni, *U Brescia, Italy*
Trevor J.M. Bench-Capon, *U Liverpool, United Kingdom*
Carlos Iván Chesñevar, *U Nacional del Sur, Bahía Blanca, Argentina*
Jürgen Dix, *TU Clausthal, Germany*
Phan Minh Dung, *Asian Institute of Technology, Thailand*
Lluis Godo, *IIIA-CSIC, Spain*
Anthony Hunter, *U College London, United Kingdom*
Antonis C. Kakas, *U Cyprus*
Gabriele Kern-Isberner, *U Dortmund, Germany*
Nicolas Maudet, *U Paris-Dauphine, France*
Peter J. McBurney, *U Liverpool, United Kingdom*
Donald Nute, *U Georgia, Athens, GE, United States*
Henry Prakken, *U Utrecht, The Netherlands & U Groningen, The Netherlands*
Iyad Rahwan, British *U Dubai, UAE & U Edinburgh, United Kingdom*
Tran Cao Son, *New Mexico State U, NM, United States*
Francesca Toni, *Imperial College London, United Kingdom*

Local Organization

Chitta Baral, *Arizona State University, Arizona, United States*

Table of Contents

Three senses of "Argument" 1
Adam Wyner, Trevor J.M. Bench-Capon, Katie Atkinson

An Abstract Presentation of Dialectical Explanations in Defeasible Argumentation 17
Alejandro García, Carlos Iván Chesñevar, Nicolás Daniel Rotstein, Guillermo R. Simari

Characterizing Defeat Graphs where Argumentation Semantics Agree 33
Pietro Baroni, Massimiliano Giacomin

A Sound and Complete Dialectical Proof Procedure for Sceptical Preferred Argumentation 49
Phan Minh Dung, Phan Minh Thang

Argumentation-based Proof for an Argument in a Paraconsistent Setting . 64
Iara Almeida, José Júlio Alferes

CaSAPI: A System for Credulous and Sceptical Argumentation 80
Dorian Gaertner, Francesca Toni

Reductio ad Absurdum Argumentation in Normal Logic Programs 96
Luís Moniz Pereira, Alexandre Miguel Pinto

Inferring Preferred Extensions by Minimal Models 114
Juan Carlos Nieves, Mauricio Osorio Galindo, Ulises Cortés

Formal Properties of the SCIFF-AF Multiagent Argumentation Framework 125
Paolo Torroni

Author Index 141

Three Senses of "Argument"

Adam Wyner[1], Trevor Bench-Capon[1], and Katie Atkinson[1]

[1] Department of Computer Science, Ashton Building
University of Liverpool
Liverpool, United Kingdom, L693BX
{azwyner, tbc, katie}@csc.liv.ac.uk

Abstract. In AI approaches to argumentation, different *senses* of argument are often conflated. We propose a *three-level* distinction between arguments, cases, and debates. This allows for modularising issues within levels and identifying systematic relations between levels. Arguments, comprised of rules, facts, and a claim, are the basic units; they instantiate argument schemes; they have no sub-arguments. Cases are sets of arguments supporting a claim. Debates are a set of arguments in an attack relation; they include cases for and against a particular claim. Critical questions, which depend on the argument schemes, are used to determine the attack relation between arguments. In a debate, rankings on arguments or argument relations are given as components based on features of argument schemes. Our analysis clarifies the role and contribution of distinct approaches in the construction of rational debate. It identifies the source of properties used for evaluating the status of arguments in Argumentation Frameworks.

Keywords. Argumentation, argument, case, debate.

1 Introduction

In AI we find a number of approaches to argumentation and argument. Some approaches represent arguments as trees or graphs (e.g. Reed and Rowe 2005), some are highly concerned with the structure of arguments (e.g. Caminada and Amgoud 2005) and the way arguments support one another (e.g. Cayrol and Lagasquie-Schiex 2005). From informal logic we have the notion of argument schemes (e.g. Walton 1996), while much of the more formal work has taken place in the context of abstract argumentation frameworks (e.g. Dung 1995). With this variety of approaches it is important to determine the relations between them, and in particular to avoid conflation of distinct ideas. To this end we will, in this paper, explore three different senses of the word "argument", all of which are represented in the previous work mentioned above, in order to give a clear characterisation of what may be intended by argument, and to identify the appropriate role of various senses in argumentation as a whole.

The Oxford English Dictionary lists seven senses of the word "argument", of which three will concern us in this paper. We begin by giving the definitions below: although these are senses 3a, 4 and 5 in the OED, we will introduce our own numbering for clarity. In Sense 1 an argument is a self-contained entity, a reason for a conclusion.

> **Sense 1: "3. a.** A statement or fact advanced for the purpose of influencing the mind; a reason urged in support of a proposition.*"*

Thus we can see an argument in Sense 1 as a pair <reason, conclusion>, which makes no reference to any other arguments. This is quite a common use in AI and elsewhere: Toulmin's scheme (Toulmin 1958), as originally presented, was "stand alone" in the sense that it made no reference to the grounds on which the reasons were believed, nor the uses to which the claim might be put. The arguments based on the many schemes found in (Walton 1996) share this feature. Most common of all in AI are arguments of the form "Q because P" representing the application of a single (defeasible) rule. In law this is akin to a single point made within a case.

In the second sense, reference is made to where the reasons come from:

> **Sense 2: "4.** A connected series of statements or reasons intended to establish a position (and, *hence*, to refute the opposite); a process of reasoning; argumentation."

In Sense 2 we move beyond a single step of reasoning, giving grounds for the reasons advanced for the conclusion. An argument in Sense 2 may be seen as a chain of reasons, reasons for reasons. In AI this can appear as a proof tree, as with the typical "how" explanation of a rule based expert system, and is a commonly used notion of argument in work such as (Pollock 2001) when an "argument" has sub-arguments: e.g. "P → Q" and "Q → R" are sub-arguments of the argument "P, P → Q, Q → R, so R" where "→" is some kind of, possibly defeasible, implication. In law this may be seen as the whole case to be presented for a particular party.

The third sense relates arguments in the previous senses:

> **Sense 3: "5. a.** Statement of the reasons for and against a proposition; discussion of a question; debate."

In Sense 3 we have the possibility of conflict: we have reasons *against* as well as *for*, the proposition, and we may have multiple arguments in the preceding two senses on both sides. In AI this corresponds more to an argumentation framework in the sense introduced by Dung (1995). In law it corresponds to the whole of a case with all the arguments for both parties and perhaps also the adjudication of a judge.[1]

In this paper we shall distinguish between these three senses of argument. In the following we will refer to Sense 1, as an *argument*: we shall always here mean an argument which cannot be divided into sub-arguments. For Sense 2, a collection of arguments advocating a particular point of view, we shall use the term *case*. This

[1] In AI sometimes "argumentation" is used rather than "argument": in fact no distinction between these terms is reflected in the definitions given in the OED. There are senses of "argumentation" corresponding to each of the senses of "argument" discussed above. Differences seem to be in connotation: "argumentation" is typically used pejoratively, and sometimes carries a sense of process, the putting forward of arguments.

picks up on phrases such as "the case for the prosecution", but should not be confused with the whole of a case as mentioned above. Rather, for a collection of arguments for and against a point of view, we shall use the term *debate*.

In distinguishing the three senses, we also relate them. Arguments are *parts* of cases, and a case is *part* of a debate. Furthermore, changes in one of the parts may induce a change in another, as we shall see.

Before proceeding further, we should mention, for purposes of comparison, Prakken's well known four layer model of argumentation (Prakken 1997). He distinguishes a *logic* layer, which is concerned with arguments and is where questions such as whether the argument is sound can be posed. Prakken, however, does not distinguish between Senses 1 and 2, and so both arguments and cases may emerge from the logic layer. Next there is a *dialectical* layer, which examines conflicts between the arguments/cases identified in the logic layer. This layer corresponds to what we are terming debate, and it is intended to resolve conflicts between the arguments/cases identified. Next there is a *procedural* layer, which controls the conduct of the dispute, how arguments can be introduced and challenged. Finally, there is a *strategic* layer: while the procedural layer controls what it is possible or legal to do, the strategic layer determines what it is advisable to do. In what follows we will be concerned only with the logical and dialectical layers.

In Section 2, we present arguments as the basic unit. However, arguments have parts, which are specified by the argument schemes which they instantiate; for instance, arguments have claims, which is the proposition that holds if the argument succeeds. A key notion is that arguments do not have other arguments as parts. In Section 3, critical questions are presented as a means to establish attack relations between arguments; given an argument and a critical question associated with it, an affirmative answer to the question implies that another argument attacks the argument and in what way. Given arguments and attack relations, we move to the level of debates in Section 4, where sets of arguments are provided for and against a particular claim. Different sets of arguments are derived from different attack relations; in turn, the attack relations depend on the critical questions and the argument schemes that have been instantiated. In Section 5, we discuss abduction in Argumentation Frameworks. We present cases in Section 6 in terms of admissible sets in an Argumentation Framework, for a case is a set of arguments that support a particular claim. We discuss the role of evaluation metrics such as preference or value rankings in Section 7; the rankings use properties that come from particular argument schemes, and have consequences for properties of sets of arguments at the level of the Argumentation Framework.

2 Arguments

In order to generate some arguments, we will need some facts and some means of inferring conclusions from those facts. We will use as a starting point a very simple knowledge base, KB1, comprising four defeasible rules and three facts, from which we can generate a standard form of argument: *P and if P then Q, so Q* . The facts and rules of KB1 are:

R1 $P \rightarrow Q$
R2 $Q \rightarrow R$
R3 $S \rightarrow \neg Q$
R4 $T \rightarrow \neg R$
F1 P
F2 S
F3 T

We begin by forming arguments by applying the available rules to the available facts. Each of the facts is the antecedent of a rule, and so we get three arguments:

A1 F1, R1 so Q
A2 F2, R3 so $\neg Q$
A3 F3, R4 so $\neg R$

Note that A1 and A2 have conflicting claims. This is not unusual: it simply means that we have a reason to believe Q, and a reason to disbelieve Q: we are not saying that the claims of all the arguments are true, only that we have a reason to think they may be. We expect such conflicts to appear in the logic level of argumentation: it is the role of the dialectical layer to resolve them. In our terms, such conflicts open up the possibility of debate. Of course, it needs to be ensured at that level that arguments with conflicting claims are not co-tenable.

But now we have obtained Q using A1 and Q is itself the antecedent of a rule, so we can perhaps add:

A4 Q, R2, so R

Alternatively we might want to reflect that Q was derived as the conclusion of A1 and so include A1 as a sub-argument.

C1 A1, R2, so R.

Note that C1 is, in our terms a case and not an argument: it contains A1 as a sub-argument. It is a chain of arguments for R, and so what we call a case. A difference between these approaches emerges if we add another rule and fact to KB1 to get KB2:

R5 $U \rightarrow Q$
F4 U

Now we have a second argument for Q:

A5 F4, R5, so Q

Now A4 still applies, so we get no extra argument for Q, but using the approach with sub-arguments we would get a second *case* for R:

C2 A5, R2, so R

Although the production of such cases is very natural in AI, in which the chaining of rules is standard practice, and although these cases (i.e. arguments with sub-arguments) have been termed arguments in a number of common approaches (Caminda and Amgoud 2005, and Pollock 2001), we will restrict ourselves for the time being to strict arguments in Sense 1.

We see arguments in Sense 1 as *the instantiation of an argument scheme*. In relation to KB1 we will use two argument schemes:

AS1 Defeasible Modus Ponens
Data: Type: Fact | Conjunction of Facts
Warrant: Type: Rule with Data as antecedent
Claim: Type Fact: the consequent of Warrant.

AS2 Argument by Assertion
Data: Type: Fact
Claim: Type: Fact, namely Data

Now A1-5 are all instantiations of AS1: instantiating AS2 gives us four more arguments:

A6: P, so P
A7: S, so S
A8: T, so T
A9: U, so U

While in this sense, arguments do not have sub-arguments, arguments nonetheless have *parts*, as indicated by the argument schemes. Among the parts of an argument we have Data, Warrant, and Claim, and other argument schemes may have other parts.

We have now identified all the arguments that can be generated from KB2. All these arguments are sound in that they are instantiations of our permitted argument schemes. Our argument schemes do not allow the production of cases such as C1 and C2: that would require a scheme which allowed an argument to act as Data like a Fact. We do not want to allow this, since our conception of argument (Sense 1) does not permit arguments to be related to one other. As we consider later, there are relations between arguments, where the term is used in its other senses.

3 Critical Questions

Having identified the arguments, we will now wish to identify relations between them. In particular we need to identify which arguments attack one another. As noted above, A1 and A2 are in mutual conflict because the claim of one negates the claim of the other. In order to make our identification of attacks systematic, we will draw on

the notion of *critical questions*, taken from informal logic. In Walton (1996) each argument scheme is associated with a characteristic set of critical questions. Argument schemes are instantiated. Let us suppose an argument A which instantiates a scheme and with respect to which we ask a critical question. An affirmative answer to the question implies an argument which is the instantiation of some scheme and which is in some conflict with our initial argument A. As we remark below, there are several ways the conflict can arise.

So what are the critical questions in our example?

For AS2, the only possibility is that we deny the premise and conclusion, which are of course, the same for this scheme. Thus:

AS2CQ1 Have we reason to believe the premise/claim is false?

If there is an argument A which instantiates AS2 and the answer to this question is *yes*, then there will be another argument B which instantiates AS2 and which is in conflict with A. Thus, we have two arguments A and B which we say *attack* one another, for they make claims which are in conflict.

For AS1 we would expect to have three critical questions corresponding to the standard kinds of attack found in the literature, namely premise defeat, undercut and rebuttal. AS1, however, cannot be undercut, since the claim of AS1 is always a fact, not a rule, and so we cannot infer that a rule is inapplicable. Accordingly we modify AS1 to AS3:

AS3 Defeasible Modus Ponens with undercut
Data: Type: Fact | Conjunction of Facts
Warrant: Type: Rule with Data as antecedent
Claim: Type Fact | Rule: the consequent of Warrant

This gives the following three critical questions.

AS3CQ1: Have we reason to believe the data is false?
AS3CQ2: Have we reason to believe the warrant does not apply?
AS3CQ3: Have we reason to believe the claim is false?

Thus an argument whose claim is the negation of the data, or the warrant, or the claim of an instantiation of AS3 will, in their corresponding ways, attack that instantiation. Note that AS3CQ3 gives rise to a symmetric attack, the others to asymmetric attacks.

The use of these critical questions thus allows us to determine which of our arguments are in conflict.

We might also consider whether we have additional critical questions. For example, if we have used as data the claim of a defeasible argument, we will need to be wary of conclusions we draw on the basis of it, since we cannot rely on such rules to be transitive. So we might add a critical question to AS3:

AS3CQ4: Are we sure the data is true?

Such a critical question instantiates the following argument scheme:

AS4 Argument from Defeasibility:
Data: Type: Fact: where Fact is the claim of an instantiation of AS3
Claim: Type: Fact: negation of Data.

This raises doubt, but does not substantiate the doubt.

The associated critical question is:

AS5CQ1: Do we have an independent reason to believe Data?

Having discussed arguments and their relationships, we can move the discussion to the level of debates, for which we will use argumentation frameworks. There we consider the arguments only in terms of the relationships we have determined hold between them, namely *attack*. After having discussed debates, we return to discuss the cases, which we define as part of a debate.

4 Argumentation Frameworks and Debates

For our dialectical layer we will use Dung's Argumentation Framework (AF), introduced in Dung (1995). In an AF, we have arguments in attack relations. We recall some key notions of that framework.

Definition 1 *An* argument system *is a pair AF = <X,A> in which X is a set of* arguments *and A is the* attack relationship *for AF. Unless otherwise stated, X is assumed to be* finite*, and A comprises a set of ordered pairs of* distinct *arguments. A pair <x, y> is referred to as 'x* attacks *(or* is an attacker of *) y' or 'y* is attacked by *x'.*

For R, S subsets of arguments in the system AF we say that:
a) s ∈ S is attacked by *R if there is some* r∈ R such that <r, s> ∈ A.
b) x ∈ X s acceptable with respect to S *if for every y ∈ X that attacks x there is some z ∈ S that attacks y.*
c) S is conflict-free if *no argument in S is attacked by any other argument in S.*
d) *A conflict-free set S is* admissible *if every argument in S is acceptable with respect to S.*
e) *S is a* preferred extension *if it is a maximal (with respect to set inclusion) admissible set.*
f) *S is a* stable extension *if S is conflict free and every argument y, ¬ (y ∈ S), is attacked by S.*
g) *S is a* complete extension *if S is a subset of A, S is admissible, and each argument which is defended by S is in S.*

h) *S* is a grounded extension *if it is the least (wrt set inclusion) complete extension*.
i) *An argument x is* credulously accepted *if there is* some *preferred extension containing it; x is* skeptically accepted *if it is a member of* every *preferred extension.*

Dung specifically states that arguments are abstract, and that attack is the only relation between them. This in part motivates our desire to exclude cases, arguments related to other arguments which form their parts, from the dialectical layer. As discussed above, we can use our argument schemes and critical questions to identify the sets X and A. So, what is the argumentation framework, AF2, corresponding to KB2?

X is the set of all arguments generated in the previous section: {A1, A2, A3, A4, A5, A6, A7, A8, A9}.

Using AS3CQ3, we can see A1 and A2 are in conflict, since the claim of one is the negation of the claim of the other. Next AS3CQ1 shows that A2 must attack A4, since the claim of A2 negates a premise of A4. Applying these two principles gives us the attack relation: {<A1,A2>, <A2,A1>, <A2,A4>, <A3,A4>, <A4,A3>, <A2,A5>, <A5,A2>}. A graphical representation of AF2 is given in Figure 1: here, to help understanding of the diagram, we label arguments with their claim as well as their name, even though strictly these claims are abstracted away with the rest of the structure when we form an AF.

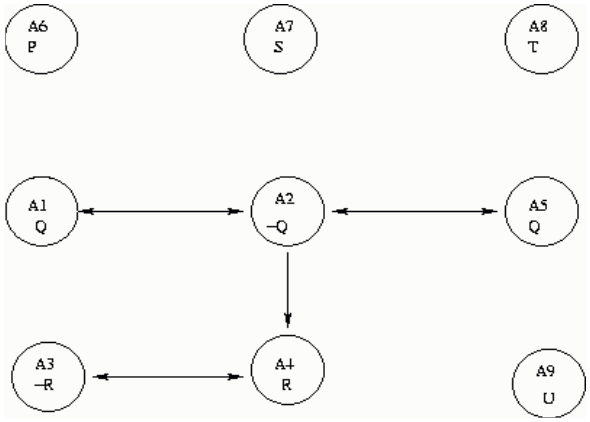

Figure 1: AF2

The grounded extension is the rather disappointing {A6,A7,A8,A9}. We have a number of preferred extensions:

{ A1, A3 A5,A6,A7,A8,A9}
{ A1, A4 A5,A6,A7,A8,A9}
{ A2,A3,A6,A7,A8,A9}

These extensions allow us, therefore, to accept any of the arguments credulously, but only the arguments from assertion sceptically. This is, of course, not very useful,

and so we often find some notion of priority between arguments. This is often based on a notion of priority between the rules on which they are based. For example we might say R5 > R3 > R1. The effect of this is to break the symmetry of the attack relation between arguments with the same conclusion: thus from KB1, A2 would now defeat A1, but the additional rule, R5, in KB2 means that in AF2 the attacks <A1, A2> and <A2, A5> are both removed, so that A2 is defeated. We would still then need to decide the priority between A3 and A4. Note again that we have to resort back to the logical level to identify the rules and their priorities.

To illustrate undercutting, suppose we extend KB2 to KB3 by adding :

R6: U → ¬R2 (i.e. U → ¬ (Q → R))

Now we can extend AF2 to AF3 by adding an extra argument which instantiates AS3:

A10 F4, R6, so ¬R2

A10 attacks A4 (by undercut), but not vice versa, so <A10,A4> is added to the attack relation of AF3.

5 Another Argument Scheme

The above discussion used two argument schemes. There is, however, no reason to limit ourselves to the sorts of arguments we can generate. For example, let us consider KB4, which is KB2 but with F1 and F4 replaced by F5, namely R. Using the argument schemes AS1-3, we can show arguments A2, A3, A7, A8 and A9 and, using argument by assertion,

A11: R, so R.

Suppose, we now introduce an additional argument scheme:

AS5 Argument from Abduction
Data: Type: Fact
Warrant: Type: Rule with Data as consequent
Claim: Type Fact: the antecedent of Warrant

This enables us to produce the following arguments[2]:

A12 F5, R2, so Q
A13 Q, R1, so P
A14 Q, R5, so U

[2] Here we do not consider arguments based on the contraposition of defeasible rules.

Like any argument scheme, AS5 will need its characteristic critical questions. For this scheme we need to consider not only the usual notions of premise defeat, undercut and explanation, but also the possibility of their being a competing, perhaps better, explanation of the claim. It is part of the notion of arguing by abduction that the justification for abducing the antecedent is that it represents the *best* explanation of the consequent. Here P and U are competing explanations for Q. We assume that two abductive arguments conflict when they have the *same* data, since we cannot reuse the explanation. This is an important point: determining whether arguments attack one another depends crucially on the argument scheme which they instantiate.

We therefore have four critical questions:

AS4CQ1: Have we reason to believe the data is false?
AS4CQ2: Have we reason to believe the warrant does not apply?
AS4CQ3: Have we reason to believe the claim is false?
AS4CQ4: Is there another explanation of the data?

Thus, instantiations of AS4 are attacked by arguments with the same data as well as the attacks applicable to AS3.

Now we can organize this into an argument framework AF4.

The set of arguments is now {A2, A3, A7, A8, A11, A12, A13, A14}.

What of the attacks? A3 and A11 are in mutual conflict, as are A2 and A12. But now using AS4CQ4 we can see that A13 and A14 are in conflict. Additionally if A3 is accepted, by AS4CQ1 A12 must fail, since the abductive premise fails. Similarly A2 attacks A13 and A14, using AS4CQ1.

Thus attacks = {<A2, A12>, <A12, A2>, <A3, A11>, <A11, A3>, <A13, A14>, <A14, A13>, <A3, A12>, <A2, A13>, <A2, A14>}

We can show the resulting AF4 in Figure 2.

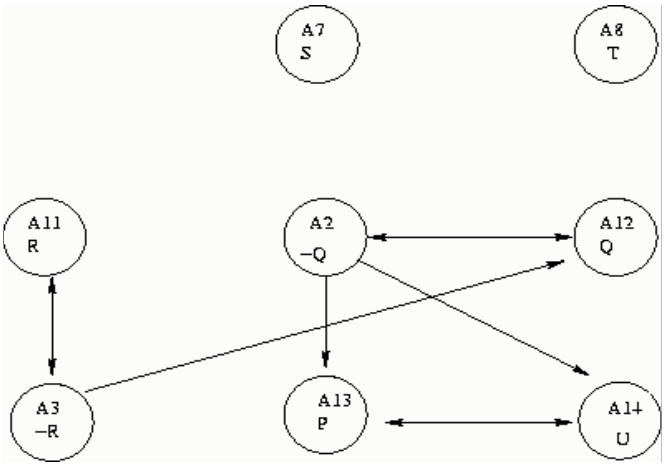

Figure 2: AF4

Preferred extensions of AF4 are:

{A7, A8, A11, A12, A13}
{A7, A8, A11, A12, A14}
{A7, A8, A11, A2}
{A7, A8, A3, A2}

We will leave for later consideration how we might choose between these preferred extensions.

A further possibility is that we might think that there may be another explanation of the claim of an instantiation of AS4, even if we don't know what it is:

AS4CQ5: Might there be another explanation?

A positive answer to this critical question instantiates AS6:

AS6: Argument from Unknown Explanation
Data: Type: Fact: where Fact is the claim of an instantiation of AS4
Claim: Type: Fact: Claim.

Note that AS6 is not legitimate if we believe that our knowledge of possible explanations is complete. This gives two critical questions:

AS6CQ1 Do we have an independent reason to believe Claim?
AS6CQ2 Is our knowledge of the explanations for Claim complete?

Applying AS5 to KB2 gives A15 and applying AS6 to KB4 gives A16-18.

A15 ¬R since Q defeasibly inferred.
A16 ¬Q since there may be an unknown explanation for R
A17 ¬P since there may be an unknown explanation for Q
A18 ¬U since there may be an unknown explanation for Q

We can usefully label the arcs in the framework with the critical questions. If we add A16-A18 to AF4 we get AF4a as shown in Figure 3.

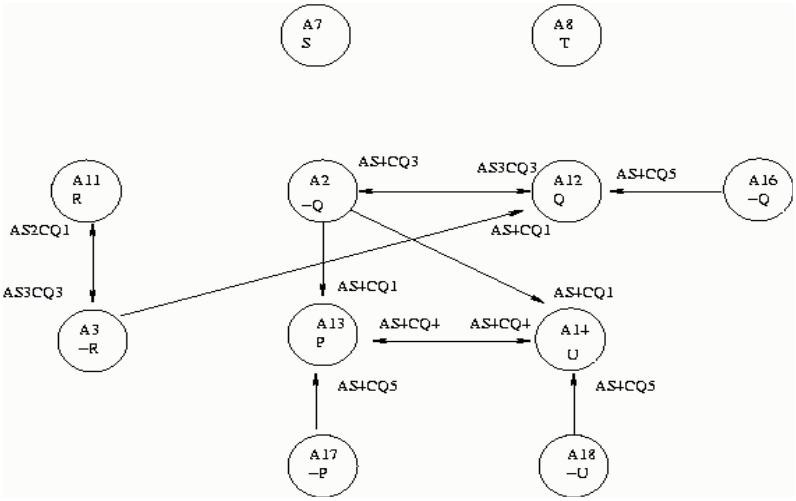

Figure 3: AF4a

6 Cases

We now need to return to the notion of a case. Recall that we decided to admit only arguments without sub-arguments into our framework, thus precluding the possibility of representing support for an argument as sub-argument. Also we want to stay within Dung's original intentions, and so do not wish to include an additional relation to show support, as is done, for example, in Cayrol and Lagasquie-Schiex (2005). We can, nevertheless, obtain a clear notion of support, and hence of arguments in Sense 2, by considering admissible sets.

An admissible set is conflict free and able to defend itself against attackers. This means that a given argument in the admissible set which is attacked will have defenders in the admissible set. Moreover if these defenders have attackers, they too will have defenders in the admissible set. Thus the minimal admissible set containing a given argument will contain all the arguments needed to make that given argument part of an admissible set. It is in this way that we can express the notion of support while staying within Dung's framework, as originally specified.

Consider, as an example, A13 in AF4 above. This argument appears in only one preferred extension: {A7, A8, A11, A12, A13}. A12 is needed to defend A13 against A2, and A11 is needed to defend A12 against A3. A7 and A8 are included only to make the extension maximal. Thus the minimal admissible set containing A13 is {A11, A12, A13}. Thus we can say that A13 is supported by A11 and A12, and that these three arguments form the case for the claim of A13, P. This would make the case something like "P is the best explanation of Q, which is the best explanation of R, which is known to hold." Had we adopted the sub-argument approach we would have had

C3: A11, A12, R1, so P,

showing the connection between chains of arguments and admissible sets.

Note, however, that on this notion of case, A2 is *not* supported by A7, which would, as being the datum required to infer ¬Q using A2, often be thought to be a sub-argument of A2. We argue that we should not see A7 as supporting A2, because this aspect of A2 is not in question, the only attack on A2 coming from A12, which is a rebuttal, not a premise defeat. In other words, A7 is accepted without question, and so its claim can be presumed in any argument that requires it, meaning that the argument stands in no need of support in this respect. Of course, if the logic level had in fact generated an argument with claim ¬S, we would have an argument attacking the datum of A2, but that argument would itself be attacked by A7. In that case A7 would be required to admit A2 into an admissible set, and so would be regarded as supporting it. We feel that this notion of support, which only calls in potential supporters if they are required, is clearer than notions which attempt to identify all potential supporters at the logical level and without regard to their supporting role in a debate.

7 Evaluation

When discussing AF2 and AF4, we used the standard notion of evaluating the argumentation framework in which all arguments have equal weight, and all attackers succeed, and where we calculate the grounded, preferred or stable extensions, according to our semantic preferences. Yet, as noted earlier, we may have multiple preferred extensions which we want to differentiate; we want to have some principled *reason* to choose between them.

The usual method of distinguishing between multiple preferred extensions, and so provide a reason to choose between them, is to ascribe some property to the arguments representing their strength, and to require an attacker to be at least as strong as the attacked argument if the attack is to succeed. In virtue of these more fine-grained attacks, we can distinguish among previously undistinguished preferred extensions. For example Amgoud and Cayrol (2002) use preferences in this way, and Bench-Capon (2003) uses the notion of value (the social interest promoted by the acceptance of an argument) to determine the relative strength of pairs of arguments. But where do these properties come from?

The answer must be that they come from the argument schemes instantiated to produce the arguments in the framework. At the very least therefore the arguments can be ascribed the property of being instantiations of a particular argument scheme. This in turn means that we could apply a preference order to schemes: for example we might rate Argument from Assertion most highly, since this requires a known fact in the database, then Defeasible Modus Ponens, then Abduction. Or we could choose a different order if we desired. The general idea is that the arguments can be ascribed properties, these properties can be ranked, and this ranking is used in determining the status of arguments in the framework. Note that although the schemes determine which properties can be ascribed to the arguments, the ranking is produced

independently, and that different rankings may be applied to the framework for different purposes or by different audiences.

If we use different argument schemes, we may be able to ascribe a wider range of properties. Three examples are:

- One well known argument scheme is Argument from Authority (e.g. Walton 1996). In order to instantiate this scheme an authority must be identified. All arguments instantiating this scheme therefore will have the property of being endorsed by some particular authority. If we have several competing authorities, we can use a ranking of confidence in these authorities to determine the strength of arguments.
- In Atkinson (2005) an argument scheme for practical reasoning is proposed. In this scheme the social value promoted by acceptance of the argument has to be identified in order to instantiate the scheme. This allows arguments from this scheme to be labeled with these values, which in turn means that the resulting framework can be regarded as a Value Based Framework (Bench-Capon 2003)), and evaluated according to a particular audience's ranking of the values.
- Work on case-based reasoning in law such as Ashley (1990), effectively identifies a set of argument schemes and critical questions tailored to reasoning with legal precedents. Each of these argument schemes is related to the citation of a legal decision, and so comes with information such as the date of the case, the jurisdiction in which it was decided, and the level of court which made the decision. All of these things represent useful properties of argument which can feed into the evaluation of the status of arguments when they are formed into a framework.

Properties of arguments will not, however, suffice for AF4a. The use of Argument Scheme AS6 means that any abductive argument will have an attacker. If attacks always succeeded, this means that we simply could not use abductive arguments. The implication is that we need to provide some way for attacks to fail. One obvious strategy is to use the labels on the attacks. For example it might be that one considered that AS4CQ5 should not defeat the argument it attacks, unless that argument is attacked by some other argument. Thus in AF4a, none of the abductive arguments will succeed, because they have independent attackers. But suppose we did not have the fact that S, so that A2 no longer can be made. Now if we accept A11 to defeat the other attacker of A12, we will accept A12. A13 and A14 are, however, still defeated since they mutually attack, as well as being attacked using AS4CQ5. This seems reasonable, since we do not have another explanation of R, but P and U are competing explanations for Q, and we have no reasons given for preferring one to another.

There are two important points to note here. First, the properties of arguments can play an important role in deciding the status of arguments in an argumentation framework, since they can form the basis for rational choice between competing preferred extensions. Second, the properties ascribed to arguments in the AF need to have their origin in the argument schemes which ground the arguments in the framework. The schemes used will thus determine the properties which are available at the framework level.

8 Summary

In this paper we have attempted to make clear distinctions between three senses in which "argument" may be used, and which can sometimes appear to be conflated in work on argumentation.

First we have the level of the atomic argument. For us this is an instantiation of an argument scheme, and cannot be divided into any constituent parts which are themselves arguments. There is a wide variety of argument schemes found in the literature: the choice of which schemes to use will depend on the nature of the application – different schemes are appropriate for legal, practical, scientific, mathematical and evidential reasoning. These schemes have associated with them critical questions, and various arguments will form the basis of these questions posed against other arguments. This provides a principled basis for deciding which arguments are in conflict, and whether the conflict is symmetric or not. Also the different critical questions permit attacks to be labeled according to the question being posed. Finally particular schemes will permit the ascription of properties to these arguments.

The above allows us to form the arguments into an argumentation framework, which represents the notion of argument as debate, sets of reasons for and against particular propositions. At this level it is possible to evaluate arguments to form a view as to which should be accepted and which should be rejected. Where suitable argument schemes have been used, properties of arguments and attacks can be used to inform the evaluation, according to rankings of these properties.

Finally we can define the notion of a case, a set of supporting arguments for a particular point of view, in terms of a minimal admissible set taken from the framework.

We believe that it is important to maintain a distinction between these three senses. Moreover we can see that our separation shows clearly the links between them. An argumentation framework is independent of the argument schemes used to form it. The properties of arguments do depend on the schemes used, and so some evaluations will be possible only if the arguments instantiate particular argument schemes. The notion of support is derived from the status of arguments in the framework level, rather than being identified at the logic level and thus is dependent on the method of evaluation for the framework.

Acknowledgments. The authors thank the support of the Estrella Project (The European project for Standardized Transparent Representations in order to Extend Legal Accessibility (Estrella, IST-2004-027655)) during the time when this paper was written.

References

1. Amgoud, L. and Cayrol, C.: Inferring from Inconsistency in Preference-Based Argumentation Frameworks. J. Autom. Reason., Kluwer Academic Publishers, *29* (2002) 125-169
2. Ashley, K.: Modeling Legal Argument: Reasoning with Cases and Hypotheticals. Bradford Books/MIT Press (1990)
3. Atkinson, K.: What Should We Do?: Computational Representation of Persuasive Argument in Practical Reasoning. Department of Computer Science, University of Liverpool, Liverpool, United Kingdom (2005)
4. Bench-Capon, T. J. M.: Persuasion in Practical Argument Using Value-based Argumentation Frameworks. J. Log. Comput.13 (2003) 429-448
5. Caminada, M. & Amgoud, L.: An Axiomatic Account of Formal Argumentation. AAAI (2005) 608-613
6. Cayrol, C. and Lagasquie-Schiex, M.: On the Acceptability of Arguments in Bipolar Argumentation Frameworks. ECSQARU (2005) 378-389
7. Dung, P. M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence 77 (1995) 321-358
8. Pollock, J.: Defensible Reasoning with Variable Degrees of Justification. Artificial Intelligence 133 (2001) 233-282
9. Prakken, H.: Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law. Kluwer Academic Publishers (1997)
10. Reed, C. and Rowe, G.: Araucaria: Software for Argument Analysis, Diagramming and Representation. International Journal on Artificial Intelligence Tools 13 (2004) 961-980.
11. Toulmin, S.: The Uses of Argument. Cambridge University Press (1958)
12. Walton, D.: Argumentation Schemes for Presumptive Reasoning. Erlbaum (1996)

An Abstract Presentation of Dialectical Explanations in Defeasible Argumentation

Alejandro J. García, Carlos I. Chesñevar, Nicolás D. Rotstein, and Guillermo R. Simari

Artificial Intelligence Research Group
Department of Computer Science and Engineering
Universidad Nacional del Sur, Av.Alem 1253, (8000) Bahía Blanca, ARGENTINA
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
e-mail: {ajg, cic, ndr, grs}@cs.uns.edu.ar

Abstract. Abstract argumentation frameworks have played a major role as a way of understanding argument-based inference, resulting in different argument-based proof procedures. We will provide an abstract characterization of the warrant construction in the context of Skeptical Argumentation Frameworks. Often in the literature an argument is regarded as an explanation as well as a form of support for a claim, and this argument is evaluated to decide if the claim is accepted. The concept of explanation has received attention from different areas in Artificial Intelligence, particulary in the Knowledge-Based Systems community. Only a few of them consider explanations in relation with argument systems. In this paper, we propose a type of explanation that attempts to fill this gap providing a perspective from the point of view of argumentation systems.

1 Introduction and Motivations

Lately, interest in argumentation has expanded at increasing pace, driven in part by theoretical advances but also by successful demonstrations of a substantial number of practical applications, such as multiagent systems [17, 1], legal reasoning [18], knowledge engineering [4], and e-government [2], among many others. In this context, abstract argumentation frameworks [9] have played a major role as a way of understanding argument-based inference, resulting in different argument-based semantics. The final goal of such semantics is to characterize which are the rationally justified (or *warranted*) beliefs associated with a given set of arguments.

Dialectical analysis in argumentation involves the exploration of an *argument search space* in order to provide a proof-theoretic characterization of an argument-based semantics. Dialectical proof procedures provide the mechanism for performing computations of warranted arguments, traversing this argument search space by generating tree-like structures (called argument trees [3] or dialectical trees [11, 7] in the literature). We will provide an abstract characterization of the warrant construction in the context of *Skeptical Argumentation Frameworks*.

From another point of view, often in the literature an argument is regarded as an explanation for a claim that is represented by a literal. That is, the claim which is being explained is put under discussion, and only after evaluating its support it will be accepted or not. The role of explanations has received attention from several areas of

Artificial Intelligence –especially in the expert systems community [15, 20, 12]. A few of them consider explanations in relation with argument systems [16]. In belief revision, the role of explanations has also been studied [10]: new knowledge is accompanied by an explanation, which is used (when needed) to resolve inconsistency with the agent's current beliefs. The piece of knowledge having the "best" explanation is the one that prevails, and is accepted as a new belief.

We will focus our discussion on those explanations that give the necessary information to understand the warrant status of a literal. Since we consider only skeptical argumentation systems based on a dialectical proof procedure, we study *dialectical explanations* (from now on, δ-Explanations). Although we consider arguments as an explanation for a literal, we are interested in obtaining the complete set of dialectical trees that justify the warrant status of that literal. We show how δ-Explanations can be a useful tool to comprehend and analyze the interactions among arguments, and for aiding in the encoding and debugging of the underlying knowledge base. Several examples, generated with an implemented system that returns, for a given query, both the answer and the associated δ-Explanation, are given throughout the paper.

An interesting review about explanations in heuristic expert systems is given in [15], which offers the following definition: "...*explaining* consists in *exposing something* in such a way that it is *understandable* for the receiver of the explanation –so that he/she improves his/her knowledge about the object of the explanation– and *satisfactory* in that it meets the receiver's expectations." In our approach, we *explain* through *exposing* the whole set of dialectical trees related to the queried literal. This information is *understandable* from the receiver's point-of-view, because all the arguments built, their statuses (*i.e.*, defeated/undefeated), and their interrelations are explicitly shown. This type of information would be *satisfactory* for the receiver, because it contains all the elements at stake in the dialectical analysis that supports the answer.

An empirical analysis about the impact of different types of explanations in the context of expert systems is given in [20] which offers a typology that includes: 1) *trace:* a record of the inferential steps that led to the conclusion; 2) *justification:* an explicit description of the rationale behind each inferential step; and 3) *strategy:* a high-level goal structure determining the problem-solving strategy used. In this typology, the authors claim that their empirical analysis have shown that the most useful type of explanation is "justification". Our δ-Explanations match both the "justification" and the "strategy" types. That is, δ-Explanations give not only the strategy used by the system to achieve the conclusion, but also the rationale behind each argument supporting that conclusion as it is clearly stated in the corresponding dialectical tree.

We agree with [16], in that "*argumentation and explanation facilities in knowledge-based systems should be investigated in conjunction*". Therefore, we propose a type of explanation that attempts to fill the gap in the area of explanations in argument systems. Our approach is to provide a higher-level explanation in a way that the whole context of a query can be revealed. The examples given will stress this point.

The rest of this paper is structured as follows. Next, we will present the basic ideas of an abstract argumentation framework with dialectical constraints, which includes several concepts common to most argument-based formalisms. Then, we will present an abstract characterization of explanation along with a concrete reification based on Defeasible Logic Programming (DELP).

2 An Abstract Framework with Dialectical Constraints

Abstract argumentation frameworks [9, 13] are formalisms for modelling defeasible argumentation [19, 5] in which some components remain unspecified. In this paper we are concerned with the study of warrant computation in argumentation systems, with focus on skeptical semantics for argumentation. As a basis for our analysis we will use an abstract argumentation framework (following Dung's seminal approach to abstract argumentation [9]) enriched with the notion of *dialectical constraint*, which will allow us to model distinguished sequences of arguments. The resulting, extended framework will be called an *argumentation theory*.

Definition 1 (Argumentation framework). [9] *An argumentation framework Φ is a pair $\langle \mathfrak{Args}, \boldsymbol{R} \rangle$, where $\mathfrak{Args}$ is a finite set of arguments and $\boldsymbol{R}$ is a binary relation between arguments such that $\boldsymbol{R} \subseteq \mathfrak{Args} \times \mathfrak{Args}$. The notation $(\mathcal{A}, \mathcal{B}) \in \boldsymbol{R}$ (or equivalently $\mathcal{A}\, \boldsymbol{R}\, \mathcal{B}$) means that $\mathcal{A}$* attacks *$\mathcal{B}$.*

Given an argumentation framework $\Phi = \langle \mathfrak{Args}, \boldsymbol{R} \rangle$, we will write $\mathfrak{Lines}_\Phi$ to denote the set of all the singleton sequences $[\mathcal{A}]$ with $\mathcal{A} \in \mathfrak{Args}$ and all possible finite sequences of arguments $[\mathcal{A}_0, \ldots, \mathcal{A}_k]$, with $k \geq 1$, such that for any pair of arguments $\mathcal{A}_i$, $\mathcal{A}_{i+1}$ it holds that $\mathcal{A}_{i+1}\, \boldsymbol{R}\, \mathcal{A}_i$, for $i = 0$ to k. Argumentation lines define a domain onto which different constraints can be defined. As such constraints are related to sequences which resemble an argumentation dialogue between two parties, we call them *dialectical constraints*. Formally:

Definition 2 (Dialectical Constraint). *Let $\Phi = \langle \mathfrak{Args}, \boldsymbol{R} \rangle$ be an argumentation framework. A* dialectical constraint $\mathbf{C}$ in the context of Φ *is any function $\mathbf{C} : \mathfrak{Lines}_\Phi \rightarrow \{True, False\}$. A given argument sequence $\lambda \in \mathfrak{Lines}_\Phi$ satisfies $\mathbf{C}$ in Φ when $\mathbf{C}(\lambda) = True$.*

An argumentation theory is defined by combining an argumentation framework with a particular set of dialectical constraints. Formally:

Definition 3 (Argumentation Theory). *An* argumentation theory T *(or just* theory*) is a pair $(\Phi, \mathbf{DC})$, where Φ is an argumentation framework, and $\mathbf{DC} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ is a finite (possibly empty) set of* dialectical constraints*.*

Given a theory $T = (\Phi, \mathbf{DC})$, the intended role of $\mathbf{DC}$ is to avoid *fallacious* reasoning by imposing appropriate constraints on argumentation lines to be considered rationally *acceptable*. Such constraints are usually defined on disallowing certain moves which might lead to fallacious situations. Typical constraints to be found in $\mathbf{DC}$ are *non-circularity* (repeating the same argument twice in an argumentation line is forbidden), *commitment* (parties cannot contradict themselves when advancing arguments), etc. It must be noted that a full formalization for dialectical constraints is outside the scope of this work. We do not claim to be able to identify every one of such constraints either, as they may vary from one particular argumentation framework to another; that is the reason why $\mathbf{DC}$ is included as a parameter in T.[1]

[1] In this respect a similar approach is adopted in [14], where different characterizations of constraints give rise to different logic programming semantics.

2.1 Argumentation Lines

As already discussed before, argument games provide a useful form to characterize proof procedures for argumentation logics.Such games model defeasible reasoning as a dispute between two parties (*Proponent* and *Opponent* of a claim), who exchange arguments and counterarguments, generating *dialogues*. A proposition Q is provably justified on the basis of a set of arguments if its proponent has a *winning strategy* for an argument supporting Q, i.e. every counterargument (defeater) advanced by the Opponent can be ultimately defeated by the Proponent. Dialogues in such argument games have been given different names (dialogue lines, argumentation lines, dispute lines, etc.). A discussion on such aspects of different logical models of argument can be found in [5, 19]. The abstract framework presented in this section is based on the results presented in [6] and [8].

Definition 4 (Argumentation Line). *Let T be an argumentation theory. An* argumentation line *λ in T is any finite sequence of arguments $[\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n]$ such that every $\mathcal{A}_i$ attacks $\mathcal{A}_{i-1}$, for $0 < i \leq n$. If $\mathcal{A}_0$ is the first element in λ, we will also say that λ is* rooted in *$\mathcal{A}_0$. We will also write $\mid \lambda \mid = n$ to denote that λ has n arguments; we will also say that the* length *of λ is n.*

Definition 5 (Initial Argumentation Segment). *Let T be an argumentation theory and let $\lambda = [\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n]$ be an argumentation line in T. Then $\lambda' = [\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_k]$ will be called an* initial argumentation segment *in λ of length k, $k \leq n$, denoted $\lfloor \lambda \rfloor_k$. When $k < n$ we will say that λ' is a* proper *initial argumentation segment in λ. We will use the term* initial segment *to refer to initial argumentation segments when no confusion arises.*

Example 1. Consider a theory $T = (\Phi, \mathbf{DC})$, with $\mathbf{DC} = \emptyset$, where the set $\mathfrak{Args}$ is $\{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$, and assume that the following relationships hold: $\mathcal{A}_1$ attacks $\mathcal{A}_0$, $\mathcal{A}_2$ attacks $\mathcal{A}_0$, $\mathcal{A}_3$ attacks $\mathcal{A}_0$, $\mathcal{A}_4$ attacks $\mathcal{A}_1$. Three different argumentation lines rooted in $\mathcal{A}_0$ can be obtained, namely: λ_1 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_4$], λ_2 = [$\mathcal{A}_0$, $\mathcal{A}_2$], λ_3 = [$\mathcal{A}_0$, $\mathcal{A}_3$]. In particular, $\lfloor \lambda_1 \rfloor_2$ = $[\mathcal{A}_0, \mathcal{A}_1]$ is an initial argumentation segment in λ_1.

Example 2. Consider a theory $T' = (\Phi, \mathbf{DC})$ where the set $\mathfrak{Args}$ is $\{\mathcal{A}_0, \mathcal{A}_1\}$, and assume that the following relationships hold: $\mathcal{A}_0$ attacks $\mathcal{A}_1$, and $\mathcal{A}_1$ attacks $\mathcal{A}_0$. An infinite number of argumentation lines rooted in $\mathcal{A}_0$ can be obtained (e.g. λ_1 = [$\mathcal{A}_0$], λ_2 = [$\mathcal{A}_0$, $\mathcal{A}_1$], λ_3 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_0$], λ_4 = [$\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_0$, $\mathcal{A}_1$], etc.).

Remark 1. Note that from Def. 4, given an argumentation line $[\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n]$ every subsequence $[\mathcal{A}_i, \mathcal{A}_{i+1}, \ldots \mathcal{A}_{i+k}]$ with $0 \leq i \leq n - k$ is also an argumentation line. In particular, every initial argumentation segment is also an argumentation line.

Intuitively, an argumentation line λ is acceptable iff it satisfies every dialectical constraint of the theory it belongs to. Formally:

Definition 6. *Given an argumentation theory T = $(\Phi, \mathbf{DC})$, an argumentation line λ is* acceptable *wrt T iff λ satisfies every $c \in \mathbf{DC}$.*

In what follows, we will assume that the notion of acceptability imposed by dialectical constraints is such that if λ is acceptable wrt a theory $T = (\Phi, \mathbf{DC})$, then any subsequence of λ is also acceptable.

Assumption 1 *If λ is an acceptable argumentation line wrt a theory $T = (\Phi, \mathbf{DC})$, then any subsequence of λ is also acceptable wrt T.*

Example 3. Consider the theory T' in Example 2, and assume that $\mathbf{DC}$={ Repetition of arguments is not allowed }. Then λ_1 and λ_2 are acceptable argumentation lines in T', whereas λ_3 and λ_4 are not.

Definition 7 (λ' extends λ). *Let T be an argumentation theory, and let λ and λ' be two argumentation lines in T. We will say that λ'* extends *λ in T iff $\lambda = \lfloor \lambda' \rfloor_k$, for some $k < \mid \lambda' \mid$, that is, λ' extends λ iff λ is a proper initial argumentation segment of λ'.*

Definition 8. *Let T be an argumentation theory, and let λ be an acceptable argumentation line in T. We will say that λ is* exhaustive *if there is no acceptable argumentation line λ' in T such that $\mid \lambda \mid < \mid \lambda' \mid$, and for some k, $\lambda = \lfloor \lambda' \rfloor_k$, that is, there is no λ' such that λ' extends λ. Non-exhaustive argumentation lines will be referred to as* partial *argumentation lines.*

Example 4. Consider the theory T presented in Example 1. Then λ_1, λ_2 and λ_3 are exhaustive argumentation lines whereas $\lfloor \lambda_1 \rfloor_2$ is a partial argumentation line. In the case of the theory T' in Example 2, the argumentation line λ_2 extends λ_1. Argumentation line λ_2 is exhaustive, as it cannot be further extended on the basis of T' with the dialectical constraint introduced in Example 3.

Definition 9. *Given a theory T, a set $S = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of argumentation lines rooted in a given argument $\mathcal{A}$, denoted $S_{\mathcal{A}}$, is called a* bundle set *wrt T iff there is no pair $\lambda_i, \lambda_j \in S_{\mathcal{A}}$ such that λ_i extends λ_j.*

Example 5. Consider the theory $T = (\Phi, \mathbf{DC})$ from Example 1, and the argumentation lines λ_1, λ_2, and λ_3. Then $S_{\mathcal{A}_0} = \{\lambda_1, \lambda_2, \lambda_3\}$ is a bundle set wrt T.

2.2 Dialectical Trees

A bundle set $S_{\mathcal{A}}$ is a set of argumentation lines rooted in a given argument $\mathcal{A}$. Such set can be thought of as a tree structure, where every line corresponds to a branch in the tree. Formally:

Definition 10 (Dialectical tree). *Let T be a theory, and let $\mathcal{A}$ be an argument in T, and let $S_{\mathcal{A}} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ be an acceptable set of argumentation lines rooted in $\mathcal{A}$. The* dialectical tree *rooted in $\mathcal{A}$* based on *$S_{\mathcal{A}}$ (denoted $\mathcal{T}_{\mathcal{A}}$) is a tree-like structure defined as follows:*

1. *The root node of $\mathcal{T}_{\mathcal{A}}$ is $\mathcal{A}$.*

2. *Let* F=$\{\mathsf{tail}(\lambda)$, *for every* $\lambda \in S_{\mathcal{A}}\}$, *and* H=$\{\mathsf{head}(\lambda)$, *for every* $\lambda \in F\}$.[2]
If $H = \emptyset$ *then* $\mathcal{T}_{\mathcal{A}}$ *has no subtrees.*
Otherwise, if $H = \{\mathcal{B}_1, \ldots, \mathcal{B}_k\}$, *then for every* $\mathcal{B}_i \in H$, *we define*

$$\text{getBundle}(\mathcal{B}_i) = \{\lambda \in F \mid \mathsf{head}(\lambda) = \mathcal{B}_i\}$$

We put $\mathcal{T}_{\mathcal{B}_i}$ *as an immediate subtree of* $\mathcal{A}$, *where* $\mathcal{T}_{\mathcal{B}_i}$ *is a dialectical tree based on* getBundle$(\mathcal{B}_i)$.

We will write $\mathfrak{Tree}_{\mathcal{A}}$ *to denote the family of all possible dialectical trees based on* $\mathcal{A}$.
We will represent as $\mathfrak{Tree}_{T}$ *the family of all possible dialectical trees in the theory* T.

Example 6. Consider the theory $T = (\Phi, \mathbf{DC})$ from Example 1, and the acceptable set $S_{\mathcal{A}_0}$ from Example 5. Fig. 1(a) shows the associated exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}_0}$.

The above definition shows how to build a dialectical tree from a bundle set of argumentation lines rooted in a given argument. It is important to note that the "shape" of the resulting tree will depend on the order in which the subtrees are attached. Each possible order will produce a tree with a different geometric configuration. All the differently conformed trees are nevertheless "equivalent" in the sense that they will contain exactly the same argumentation lines as branches from its root to its leaves. This observation is formalized by introducing the following relation which can be trivially shown to be an equivalence relation.

Definition 11. *Let* T *be a theory, and let* $\mathfrak{Tree}_{\mathcal{A}}$ *be the set of all possible dialectical trees rooted in an argument* $\mathcal{A}$ *in theory* T. *We will say that* $\mathcal{T}_{\mathcal{A}}$ *is equivalent to* $\mathcal{T}'_{\mathcal{A}}$, *denoted* $\mathcal{T}_{\mathcal{A}} \equiv_{\tau} \mathcal{T}'_{\mathcal{A}}$ *iff they are obtained from the the same bundle set* $S_{\mathcal{A}}$ *of argumentation lines rooted in* $\mathcal{A}$.

Given an argument $\mathcal{A}$, there is a one-to-one correspondence between a bundle set $S_{\mathcal{A}}$ of argumentation lines rooted in $\mathcal{A}$ and the corresponding equivalence class of dialectical trees that share the same bundle set as their origin (as specified in Def. 10). In fact, a dialectical tree $\mathcal{T}_{\mathcal{A}}$ based on $S_{\mathcal{A}}$ is just *an alternative way* of expressing the same information already present in $S_{\mathcal{A}}$. Each member of an equivalence class represents a different way in which a tree could be built. Each particular computational method used to generate the tree from the bundle set will produce one particular member on the equivalence class. In that manner, the equivalence relation will represent a tool for exploring the computational process of warrant and as we will see later, trees provide a powerful way of conceptualize the computation of warranted arguments. Next, we will define mappings which allow to re-formulate a bundle set $S_{\mathcal{A}}$ as a dialectical tree $\mathcal{T}_{\mathcal{A}}$ and viceversa.

Definition 12 (Mapping $\mathbb{T}$). *Let* T *be an argumentative theory, and let* $S_{\mathcal{A}}$ *be a bundle set of argumentation lines rooted in an argument* $\mathcal{A}$ *of* T. *We define the mapping*

$$\mathbb{T} : \wp(\mathfrak{Lines}_{\mathcal{A}}) \setminus \{\emptyset\} \rightarrow \overline{\mathfrak{Tree}_{\mathcal{A}}}$$

as $\mathbb{T}(S_{\mathcal{A}}) =_{\text{def}} \overline{\mathcal{T}_{\mathcal{A}}}$, *where* $\mathfrak{Lines}_{\mathcal{A}}$ *is the set of all argumentation lines rooted in* $\mathcal{A}$, $\overline{\mathfrak{Tree}_{\mathcal{A}}}$ *is the quotient set of* $\mathfrak{Tree}_{\mathcal{A}}$ *by* $\equiv_{\tau}$, *and* $\overline{\mathcal{T}_{\mathcal{A}}}$ *denotes the equivalence class of* $\mathcal{T}_{\mathcal{A}}$.

[2] The functions $\mathsf{head}(\Delta)$ and $\mathsf{tail}(\Delta)$ have the usual meaning in list processing, returning the first element in a list and the list formed by all elements except the first, respectively.

Proposition 1. *For any argument $\mathcal{A}$ in an argumentative theory T, the mapping $\mathbb{T}$ is a bijection.*[3]

As the mapping $\mathbb{T}$ is a bijection, so that we can define also the inverse mapping $\mathbb{S} =_{def} \mathbb{T}^{-1}$ which allow us to determine the acceptable set of argumentation lines corresponding to an arbitrary dialectical tree rooted in an argument $\mathcal{A}$. In what follows, we will use indistinctly a *set notation* (an acceptable bundle set of argumentation lines rooted in an argument $\mathcal{A}$) or a *tree notation* (a dialectical tree rooted in $\mathcal{A}$), as the former mappings $\mathbb{S}$ and $\mathbb{T}$ allow us to go from any of these notation to the other.

Proposition 2. *Let T be an argumentation theory, and let $S_{\mathcal{A}}$ be an acceptable bundle set of argumentation lines rooted in a given argument $\mathcal{A}$, $S_{\mathcal{A}} = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Let $S'_{\mathcal{A}} = \{\lambda'_1, \ldots, \lambda'_m\}$, $m \leq n$, be a set of initial argumentation segments, where every $\lambda'_i = \lfloor \lambda_i \rfloor_{k_i}$, for some $k_i \leq \mid \lambda_i \mid$, $i \leq m$. Let*

$$S'' = S'_{\mathcal{A}} \setminus \{\lambda \in S'_{\mathcal{A}} \mid \text{ there exists } \lambda' \in S'_{\mathcal{A}} \text{ and } \lambda' \text{ extends } \lambda\}. \quad (1)$$

Then S'' is also an acceptable set of argumentation lines rooted in $\mathcal{A}$.

The following proposition shows that dialectical trees can be thought of as compositional structures, in the sense that any subtree $\mathcal{T}'_{\mathcal{A}}$ of a dialectical tree $\mathcal{T}_{\mathcal{A}}$ is also a dialectical tree.

Proposition 3. *Let T be a theory, and $\mathcal{T}_{\mathcal{A}}$ a dialectical tree in T. Then it holds that any subtree $\mathcal{T}'_{\mathcal{A}}$ in $\mathcal{T}_{\mathcal{A}}$ rooted in $\mathcal{A}$ is also a dialectical tree wrt T.*

2.3 Acceptable dialectical trees

The notion of acceptable argumentation line will be used to characterize acceptable dialectical trees, which will be fundamental as a basis for formalizing the computation of *warranted arguments* in our setting.

Definition 13 (Acceptable dialectical tree). *Let T be a theory, a dialectical tree $\mathcal{T}_{\mathcal{A}}$ in T is acceptable iff every argumentation line in the associated bundle set $\mathbb{S}(\overline{\mathcal{T}_{\mathcal{A}}})$ is acceptable. We will distinguish the subset $\mathfrak{AT}\mathfrak{ree}_{\mathcal{A}}$ (resp. $\mathfrak{AT}\mathfrak{ree}_T$) of all acceptable dialectical trees in $\mathfrak{Tree}_{\mathcal{A}}$ (resp. $\mathfrak{Tree}_T$).*

As acceptable dialectical trees are a subclass of dialectical trees, all the properties previously shown apply also to them. In the sequel, we will just write "dialectical trees" to refer to acceptable dialectical trees, unless stated otherwise.

Definition 14 (Exhaustive Dialectical tree). *A dialectical tree $\mathcal{T}_{\mathcal{A}}$ will be called* exhaustive *iff it is constructed from the set $S_{\mathcal{A}}$ of all possible exhaustive argumentation lines rooted in $\mathcal{A}$, otherwise $\mathcal{T}_{\mathcal{A}}$ will be called* partial.

Besides, the exhaustive dialectical tree for any argument $\mathcal{A}$ can be proven to be unique (up to an equivalence).

[3] Due to space constrains proofs will be omitted.

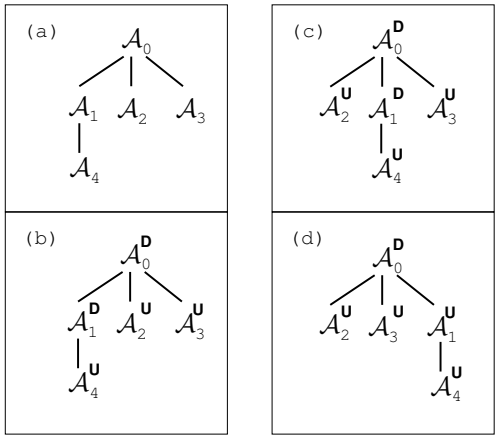

Fig. 1. (a) Exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}_0}$ for Example 6; (b) resulting tree after applying and-or marking; (c)–(d) two other exhaustive dialectical trees belonging to the equivalence class $\overline{\mathcal{T}_{\mathcal{A}_0}}$

Proposition 4. *Let T be a theory, for any argument $\mathcal{A}$ in T there is a unique exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}}$ in T (up to an equivalence wrt $\equiv_\tau$ as defined in Def. 11).*

Acceptable dialectical trees allow to determine whether the root node of the tree is to be accepted (ultimately *undefeated*) or rejected (ultimately *defeated*) as a rationally justified belief. A *marking function* provides a definition of such acceptance criterion. Formally:

Definition 15 (Marking criterion). *Let T be a theory. A marking criterion for T is a function* Mark : $\mathfrak{Tree}_T \rightarrow \{D, U\}$. *We will write* Mark$(\mathcal{T}_i) = U$ *(resp.* Mark$(\mathcal{T}_i) = D$*) to denote that the root node of $\mathcal{T}_i$ is marked as U-node (resp. D-node).*

Definition 16 (Warrant). *Let T be an argumentative theory and* Mark *a marking criterion for T. An argument $\mathcal{A}$ is a* warranted argument *(or just* warrant*) wrt a marking criterion* Mark *in T iff the exhaustive dialectical tree $\mathcal{T}_{\mathcal{A}}$ is such that* Mark$(\mathcal{T}_{\mathcal{A}}) = U$. *We will denote a marked dialectical tree as $\mathcal{T}^*_{\mathcal{A}}$.*

3 Answers and δ-Explanations

An argument is a piece of reasoning that supports a claim Q from certain evidence. The tenability of this claim must be confirmed by analyzing other arguments for and against such claim. Next, we will define *queries*, *answers* and *explanations* in the abstract context introduced in the previous Section.

The dialectical process for warranting a claim involves finding the arguments that either support or interfere with that claim. These arguments are connected through the defeat relation and are organized in dialectical trees. Observe that given a claim there

could exist in T different arguments that support it, and each argument will generate a different dialectical tree.

Definition 17 (T-Queries). *Let T be an argumentation theory. A* T-query Q *posed to the theory T will represent the process of finding out the existence, and the warranting status, of the posible arguments for Q and $\overline{Q}$.*[4]

We will show below that the returned answer for Q will be only the result of analyzing a set of dialectical trees that have been built and considered as to support this answer. Thus, to understand why a query has that particular answer, it is essential to consider which arguments have been considered and what connections exist among them.

It is important to notice that δ-Explanations are at the crux of an argumentation system whose proof procedure is based on the construction of dialectical trees. They present the reasoning carried out by the system, and they allow to visualize the support for the answer given. It is clear that without this information it will be very difficult to understand the returned answer.

Definition 18 (δ-Explanation). *Let T be an argumentation theory and let Q be a claim. Let $\mathcal{A}_0$,...,$\mathcal{A}_n$ be all the arguments for Q from T, and $\mathcal{B}_0$,...,$\mathcal{B}_m$ be all the arguments for $\overline{Q}$ from T. Then, the* explanation *for Q in T is the set of marked dialectical trees $\mathcal{E}_T(Q) = \{\mathcal{T}^*_{\mathcal{A}_0},...,\mathcal{T}^*_{\mathcal{A}_n}\} \cup \{\mathcal{T}^*_{\mathcal{B}_0},...,\mathcal{T}^*_{\mathcal{B}_m}\}$.*

Now it is possible to define T-answers in terms of the associated δ-Explanations.

Definition 19 (T-answer). *Given an argumentation theory T and a query Q, the answer for Q is:*

- YES, *if at least one tree in $\mathcal{E}_T(Q)$ warrants Q.*
- NO, *if at least one tree in $\mathcal{E}_T(Q)$ warrants $\overline{Q}$.*
- UNDECIDED, *if $\mathcal{E}_T(Q)$ is non empty, but no tree in $\mathcal{E}_T(Q)$ warrants Q nor $\overline{Q}$.*
- UNKNOWN, *if there is no argument for Q in T.*

Notice that if there is a dialectical that shows that an argument warrants Q then there is no argument that warrants $\overline{Q}$.

4 Answers and δ-Explanations in DeLP: A Reification

Next, we will define *queries*, *answers* and *explanations* using the framework provided by DeLP (see [11] for full details on DeLP). Extending the abstract presentation above, we will introduce two types of queries: ground (called DeLP-queries) and schematic. For both types of queries we will define explanations and a way to obtain the corresponding answer, that is: YES, NO, UNDECIDED or UNKNOWN.

Definition 20 (DeLP-queries). *A* DeLP-query *is a ground literal that* DeLP *will try to warrant. A query with at least one variable will be called* schematic query *and will account for the set of* DeLP*-queries that unify with the schematic one.*

[4] The notation $\overline{Q}$ is used to represent the complement of Q with respect to strong negation, *i.e.*, $\overline{a}$=$\sim a$ and $\overline{\sim a}$=a.

In DELP, δ-*Explanations* for answers will be the set of dialectical trees that have been explored to obtain a warrant for that query. The definition for a δ-Explanation for a DELP-query follows, whereas explanations for schematic queries will be introduced by the end of this Section. It is clear that without the information regarding the dialectical trees it will be very difficult to understand the returned answer. Next, we will introduce explanations for ground queries and we will generalize them for schematic queries.

Definition 21 (δ-Explanations for a DELP-query).
Let $\mathcal{P}$ be a DELP*-program and Q a* DELP*-query. Let $\langle \mathcal{A}_0, Q\rangle$,...,$\langle \mathcal{A}_n, Q\rangle$ be all the arguments for Q from $\mathcal{P}$, and $\langle \mathcal{B}_0, \overline{Q}\rangle$,...,$\langle \mathcal{B}_m, \overline{Q}\rangle$ be all the arguments for $\overline{Q}$ from $\mathcal{P}$. Then, the* explanation *for Q in $\mathcal{P}$ is the set of marked dialectical trees $\mathcal{E}_{\mathcal{P}}(Q) = \{\mathcal{T}^*\langle \mathcal{A}_0, Q\rangle,...,\mathcal{T}^*\langle \mathcal{A}_n, Q\rangle\} \cup \{\mathcal{T}^*\langle \mathcal{B}_0, \overline{Q}\rangle,...,\mathcal{T}^*\langle \mathcal{B}_m, \overline{Q}\rangle\}$.*

Using these concepts we can define DELP-answers.

Definition 22 (DELP-answer). *Given a* DELP*-program $\mathcal{P}$ and a* DELP*-query Q, the answer for Q is:*

- YES, *if at least one tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants Q.*
- NO, *if at least one tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants $\overline{Q}$.*
- UNDECIDED, *if no tree in $\mathcal{E}_{\mathcal{P}}(Q)$ warrants Q nor $\overline{Q}$.*
- UNKNOWN, *if Q is not in the signature of $\mathcal{P}$.*

Example 7. Consider the DELP-program (Π_7, Δ_7) where:

$$\Pi_7 = \left\{ \begin{array}{l} bird(X) \leftarrow chicken(X) \\ chicken(little) \\ chicken(tina) \\ scared(tina) \\ bird(rob) \end{array} \right\} \quad \Delta_7 = \left\{ \begin{array}{l} flies(X) \prec bird(X) \\ flies(X) \prec chicken(X), scared(X) \\ \sim flies(X) \prec chicken(X) \end{array} \right\}$$

From the DELP-program (Π_7, Δ_7) the following arguments can be obtained (due to space restrictions *'tina'* will be abbreviated to *'t'* and *'flies(tina)'* to *'f'*): $\langle \mathcal{A}_1, f\rangle$ = $\langle \{flies(t) \prec bird(t)\}, flies(t)\rangle$, $\langle \mathcal{A}_2, \sim f\rangle$ = $\langle \{\sim flies(t) \prec chicken(t)\}, \sim flies(t)\rangle$, and $\langle \mathcal{A}_3, f\rangle$ = $\langle \{flies(t) \prec chicken(t), scared(t)\}, flies(t)\rangle$. The argument $\langle \mathcal{A}_2, \sim f\rangle$ defeats $\langle \mathcal{A}_1, f\rangle$, $\langle \mathcal{A}_3, f\rangle$ defeats $\langle \mathcal{A}_2, \sim f\rangle$, and *[*$\langle \mathcal{A}_1, f\rangle$, $\langle \mathcal{A}_2, \sim f\rangle$, $\langle \mathcal{A}_3, f\rangle$*]* is an acceptable argumentation line.

Figure 2 shows the δ-Explanation for the DELP-query '$flies(tina)$', where two dialectical trees for '$flies(tina)$' are marked "U". Therefore, '$flies(tina)$' is warranted and the answer is YES. Note that the δ-Explanation of Figure 2 is also an explanation for query '$\sim flies(tina)$' whose answer is NO. Finally, observe that the answer for *'walks(tim)'* is UNKNOWN, because it is not in the program signature.

Remark 2. The explanation for complementary literals will always be the same, since it is composed by both the trees for the literal and the trees for its complement.

As we will show in the examples below, the semantics of the programs is *sensitive* to the addition or deletion of rules and facts. That is, a new fact added to a program

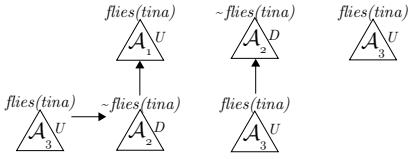

Fig. 2. δ-Explanation for $flies(tina)$

can have a big impact on the number of arguments that can be built from the modified program. Taking into account this characteristic and considering the many possible interactions among arguments via the defeat relation (that lead to the construction of different dialectical trees), δ-Explanations become essential for understanding the reasons that support an answer.

Example 8. Consider the DELP-program (Π_8, Δ_8): $\Pi_8 = \{q, t\}, \Delta_8 = \{(r \prec q), (\sim r \prec q, s), (r \prec s), (\sim r \prec t)\}$, where the following arguments can be built: $\langle \mathcal{R}_1, \sim r\rangle$ = $\langle \{\sim r \prec t\}, \sim r\rangle$, and $\langle \mathcal{R}_2, r\rangle$ = $\langle \{r \prec q\}, r\rangle$. From this program the answer for the query 'r' is UNDECIDED, and Figure 3 shows its δ-Explanation. Note that, although the literal 's' is in the program signature (in the body of a rule), there is no supporting argument for it. Therefore, the answer for query 's' is UNDECIDED, and the δ-Explanation is the empty set (*i.e.*, $\mathcal{E}_{(\Pi_8,\Delta_8)}(s)$=$\emptyset$).

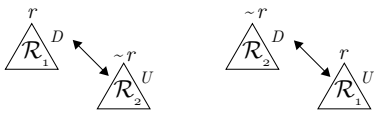

Fig. 3. δ-Explanation $\mathcal{E}_{(\Pi_8,\Delta_8)}(r)$

Remark 3. DELP-queries with UNKNOWN answers always have an empty δ-Explanation. However, DELP-queries that have UNDECIDED answers may have empty or non-empty explanations. Finally, DELP-queries with YES or NO answers will always have a non-empty explanation.

Example 9. (Extends Ex. 8) In this example we see how the introduction of a single fact in (Π_8, Δ_8) makes a significant difference in $\mathcal{E}_{(\Pi_8,\Delta_8)}(r)$. Consider the DELP-program $(\Pi_8 \cup \{s\}, \Delta_8)$ where the fact 's' is added to the program of Example 8. If we query for 'r' again, we get the answer NO with the δ-Explanation shown in Figure 4. Note that this δ-Explanation consists now of two more trees than the one in the previous example. This is so because there are two newly generated arguments: $\langle \mathcal{R}_3, r\rangle$ = $\langle \{r \prec s\}, r\rangle$, and $\langle \mathcal{R}_4, \sim r\rangle$ = $\langle \{\sim r \prec q, s\}, \sim r\rangle$

It is our contention that, in DELP, the answer for a query should be easily explained by presenting the user the associated dialectical trees. From this set of trees the answer

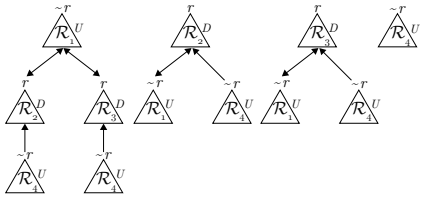

Fig. 4. δ-Explanation $\mathcal{E}_{(\Pi_{8\cup\{s\}},\Delta_8)}(r)$

becomes thoroughly justified, and the context of the query is revealed. The following examples have more elaborated DELP-programs and the δ-Explanations show that a defeater $\mathcal{D}$ for $\mathcal{A}$ may attack an inner point of $\mathcal{A}$.

Example 10. Consider the DELP-program (Π_{10}, Δ_{10}), where $\Pi_{10} = \{c, e, f\}$ and $\Delta_{10} = \left\{ \begin{array}{lllll} (a \prec b), & (b \prec c), & (\sim b \prec d), & (d \prec e), & (\sim d \prec f, e), \\ (a \prec x), & (x \prec c), & (\sim x \prec e), & (a \prec h), & (h \prec f), \end{array} \begin{array}{l} (\sim b \prec e), \\ (\sim h \prec i) \end{array} \right\}$

the following arguments can be built: $\langle \mathcal{A}_1, a\rangle = \langle\{(a \prec h), (h \prec f)\}, a\rangle$
$\langle \mathcal{B}_1, b\rangle = \langle\{b \prec c\}, b\rangle$ $\langle \mathcal{B}_2, \sim b\rangle = \langle\{\sim b \prec e\}, \sim b\rangle$
$\langle \mathcal{D}_1, d\rangle = \langle\{d \prec e\}, d\rangle$ $\langle \mathcal{D}_2, \sim d\rangle = \langle\{(\sim d \prec f, e)\}, \sim d\rangle$
$\langle \mathcal{X}_1, x\rangle = \langle\{x \prec c\}, x\rangle$ $\langle \mathcal{X}_2, \sim x\rangle = \langle\{\sim x \prec e\}, \sim x\rangle$

From (Π_{10}, Δ_{10}) the answer for 'a' is YES, and the answer for '$\sim a$' is NO. As stated in Remark 2, although both queries have different answers, they both have the same δ-Explanation, which is depicted in Figure 5.

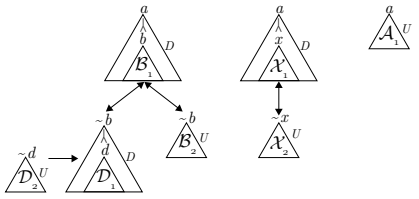

Fig. 5. δ-Explanation $\mathcal{E}_{(\Pi_{10},\Delta_{10})}(a)$

From the DELP programmer point-of-view, δ-Explanations give a global idea of the interactions among arguments within the context of a query. This is an essential debugging tool when programming: if unexpected behavior arises, the programmer can check the given explanations to detect errors.

In the previous examples we have not shown an explanation associated with a query with an UNKNOWN answer, because this type of answers have an empty δ-Explanation.

In a similar manner, observe that queries that do not correspond to the intended domain of the program will return the answer UNKNOWN. This will capture errors like querying for *"fly"* instead of *"flies"*, or a query like *"penguin(X)"* in Example 7.

Now we will extend the notion of explanation to encompass *schematic queries*. A schematic query is a query that has at least one variable (see Definition 20), and hence it represents the set of DELP-queries that unify with it. We will extend the definition of δ-Explanation to include schematic queries. In the DELP-program of Example 7, the schematic query $flies(X)$ will refer to $flies(tina)$ and $flies(little)$.

Observe that there are actually infinite terms that unify with variable X. However, all queries with terms that are not in the program signature will produce an UNKNOWN answer and therefore an empty explanation. Thus, the set of instances of a schematic query that will be considered for generating an explanation will refer only to those instances of DELP-queries that contain constants from the program signature.

Definition 23 (Generalized δ-Explanation).
Let $\mathcal{P}$ be a DELP*-program and Q a schematic query. Let $\{Q_1, \ldots, Q_z\}$ be all the instances of Q so that their* DELP*-answer is different from* UNKNOWN*. Let $\mathcal{E}_{\mathcal{P}}(Q_i)$ be the δ-Explanation for the* DELP*-query Q_i ($1 \leq i \leq z$) from program $\mathcal{P}$. Then, the* generalized δ-Explanation *for Q in $\mathcal{P}$ is $\mathcal{E}_{\mathcal{P}}(Q)$ = { $\mathcal{E}_{\mathcal{P}}(Q_1)$, ..., $\mathcal{E}_{\mathcal{P}}(Q_z)$}.*

Observe that a δ-Explanation (Definition 21) is a particular case of a Generalized δ-Explanation, where the set $\mathcal{E}_{\mathcal{P}}(Q)$ is a singleton.

Example 11. Consider again the DELP-program (Π_7, Δ_7), and suppose that we want to know if from this program it can be warranted that a certain individual does not fly. If we query for $\sim flies(X)$, the answer is YES, because there is a warranted instance: $\sim flies(little)$. The supporting argument is (*'little'* was abbreviated to *'l'*): $\langle \mathcal{B}_1, \sim flies(l) \rangle$ = $\langle \{ \sim flies(l) \prec chicken(l) \}, \sim flies(l) \rangle$. The trees of the generalized explanation are shown in Figure 6. This explanation also shows that the other instance ($\sim flies(tina)$) is not warranted. Note that the answer for the schematic query $flies(X)$ is also YES, but with a different set of warranted instances: $flies(tina)$ and $flies(rob)$. The supporting argument for instance '$X = tina$' was already discussed, and the undefeated argument for instance '$X = rob$' is: $\langle \mathcal{C}_1, flies(rob) \rangle$ = $\langle \{ flies(rob) \prec bird(rob) \}, flies(rob) \rangle$. The generalized δ-Explanation for $flies(X)$ is the same as the one for $\sim flies(X)$, depicted in Figure 6 (see Remark 2).

Definition 24 (DELP**-answer for a schematic query).** *Given a* DELP*-program $\mathcal{P}$ and a schematic query Q, the answer for Q is*

- YES*, if there exists an instance Q_i of Q such that at least one tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ warrants Q_i.*
- NO*, if there exists an instance Q_i of Q such that at least one tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ warrants $\overline{Q_i}$.*
- UNDECIDED*, if for every instance Q_i of Q that is in the signature of $\mathcal{P}$, there is no tree in $\mathcal{E}_{\mathcal{P}}(Q_i)$ that warrants Q_i nor $\overline{Q_i}$.*
- UNKNOWN*, if there is no instance Q_i of Q such that Q_i is in the signature of $\mathcal{P}$.*

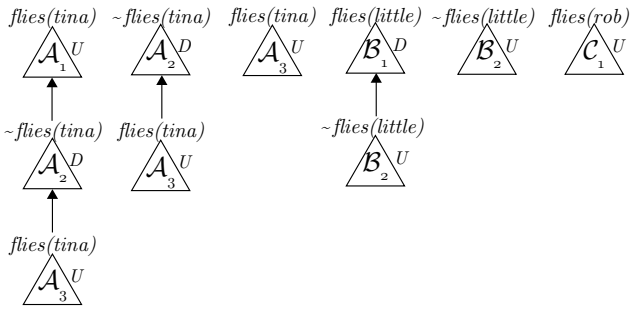

Fig. 6. Generalized δ-Explanation for '$\sim flies(X)$'

Observe that Definition 22 is a particular case of the previous definition, where there is a single instance of Q.

Example 12. Consider the following DELP-program:

$$\Pi_{12} = \left\{ \begin{array}{l} adult(peter) \\ adult(annie) \\ unemployed(peter) \\ student(annie) \end{array} \right\} \Delta_{12} = \left\{ \begin{array}{l} has\_a\_car(X) \prec adult(X) \\ \sim has\_a\_car(X) \prec unemployed(X) \\ \sim has\_a\_car(X) \prec student(X) \end{array} \right\}$$

where the following arguments can be built ('has_a_car' was replaced by 'car', '$annie$' by 'a', and '$peter$' by 'p'): $\langle \mathcal{A}_1, car(a) \rangle$ = $\langle \{car(a) \prec adult(a)\}, car(a) \rangle$,
$\langle \mathcal{A}_2, \sim car(a) \rangle$ = $\langle \{\sim car(a) \prec student(a)\}, \sim car(a) \rangle$,
$\langle \mathcal{P}_1, car(p) \rangle$ = $\langle \{car(p) \prec adult(p)\}, car(p) \rangle$, and
$\langle \mathcal{P}_2, \sim car(p) \rangle$ = $\langle \{\sim car(p) \prec unemployed(p)\}, \sim car(p) \rangle$.
When querying for '$has\_a\_car(X)$', variable 'X' unifies with both '$annie$' and '$peter$'. Then, DELP builds arguments for both instances: $\mathcal{A}_1$ and $\mathcal{A}_2$ for '$X = annie$', and $\mathcal{P}_1$ and $\mathcal{P}_2$ for '$X = peter$'. From Figure 7, it is clear that no argument is undefeated, *i.e.*, there is no tree that warrants '$has\_a\_car(X)$', for either of the two instances. Therefore, the answer is UNDECIDED, and the variable remains unbound.

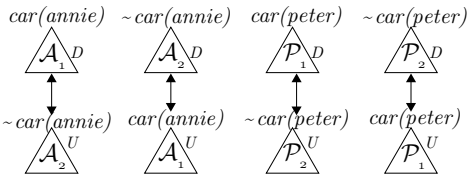

Fig. 7. Generalized δ-Explanation for '$has\_a\_car(X)$'

Schematic queries give us the possibility of asking more general questions than ground queries. Now we are not asking whether a certain piece of knowledge can be

believed, but we are asking if there exists an instance of that piece of knowledge (related to an individual) that can be warranted in the system. This could lead to deeper reasoning as we may pose a query, gather the warranted instances and continue reasoning with those individuals.

The δ-Explanations system receives a DELP-program P, a query Q, and an argument comparison criterion C, and returns a δ-Explanation EX and the corresponding answer ANS. The system is described by the following algorithm in a Prolog-like notation:

```
d_Explanations(P,C,Q,EX,ANS):-
    warrants(P,Q,C,WSQ), complement(Q,NQ), warrants(P,NQ,C,WSNQ),
    get_trees(WSQ,WSNQ,EX),  get_answer(Q,WSQ,WSNQ,ANS).

get_answer(_,WSQ,WSNQ,yes):-WSQ \= [].
get_answer(_,WSQ,WSNQ,no):-WSNQ \= [].
get_answer(Q,_,_,unknown):-not_in_signature(Q).
get_answer(_,_,_,undecided).
```

The above described system is fully implemented and offers support for queries, answers and explanations. Explanations are written into an XML file, which is parsed by a visualization applet. The visualization of trees belonging to dialectical explanations is enhanced by allowing the user to zoom-in/out, implode/explode arguments, *etc*. The internal structure of an argument is hidden when imploding, and a unique tag is shown instead.

Lemma 1 (δ-Explanation Soundness). *Let $\mathcal{P}$ be a* DELP*-program, C an argument comparison criterion, and Q a schematic query posed to $\mathcal{P}$. Let E be the δ-Explanation returned in support of the answer A. Then E justifies (Definition 24) A.*

Lemma 2 (δ-Explanation Completeness). *Let $\mathcal{P}$ be a* DELP*-program, C an argument comparison criterion, and Q a schematic query posed to $\mathcal{P}$. Let E be the δ-Explanation returned in support of the answer A. Then E contains all the possible justifications (Definition 24) for any instance of A.*

5 Conclusions

In this paper, we have addressed the problem of providing explanation capabilities to an argumentation system. This is an important, and yet undeveloped field in the area. Our focus is put on argumentation systems based on a dialectical proof procedure, studying *dialectical explanations*. We have defined an abstract system and a concrete reification with explanation facilities. We consider the structures that provide information on the warrant status of a literal. As the system has been implemented, we are developing applications that use the δ-Explanation system as subsystem.

Acknowledgments

This work was partially supported by CONICET (PIP 5050), ANPCyT (PICT 15043, PAV076), Project TIN2006-15662-C02-02 (MEC, Spain) and SGCyT-UNS.

References

1. L. Amgoud, N. Maudet, and S. Parsons. An argumentation-based semantics for agent communication languages. In *Proc. of the 15th. ECAI, Lyon, France*, pages 38–42, 2002.
2. K. Atkinson, T. J. M. Bench-Capon, and P. McBurney. Multi-agent argumentation for edemocracy. In *Proceedings of the Third European Workshop on Multi-Agent Systems, Brussels, Belgium*, pages 35–46. Koninklijke Vlaamse Academie, 2005.
3. P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 1:2(128):203–235, 2001.
4. D. Carbogim, D. Robertson, and J. Lee. Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*, 15(2):119–149, 2000.
5. C. Chesñevar, A. Maguitman, and R. Loui. Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.
6. C. Chesñevar and G. Simari. A lattice-based approach to computing warranted belief in skeptical argumentation frameworks. In *Proc. of the 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007), Hyberabad, India*, page (in press), January 2007.
7. C. Chesñevar, G. Simari, T. Alsinet, and L. Godo. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of the Intl. Conf. in Uncertainty in Art. Intelligence. (UAI 2004). Banff, Canada*, pages 76–84, July 2004.
8. C. Chesñevar, G. Simari, and L. Godo. Computing dialectical trees efficiently in possibilistic defeasible logic programming. *Proc. of the 8th Intl. Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, pages 158–171, September 2005.
9. P. Dung. On the Acceptability of Arguments and its Fundamental Role in Nomonotonic Reasoning and Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.
10. Marcelo A. Falappa, Gabriele Kern-Isberner, and Guillermo R. Simari. Explanations, belief revision and defeasible reasoning. *Artif. Intell.*, 141(1):1–28, 2002.
11. A. García and G. Simari. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.
12. Giovanni Guida and Marina Zanella. Bridging the gap between users and complex decision support systems: the role of justification. In *ICECCS '97: Proc. 3rd IEEE Int. Conf. on Engineering of Complex Computer Systems*, pages 229–238, Washington, DC, 1997.
13. H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *ICAIL*, pages 53–62, 1999.
14. A. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic Programming*, 9(4):515:562, 1999.
15. Carmen Lacave and Francisco J. Diez. A review of explanation methods for heuristic expert systems. *Knowl. Eng. Rev.*, 19(2):133–146, 2004.
16. B. Moulin, H. Irandoust, M. Bélanger, and G. Desbordes. Explanation and argumentation capabilities: Towards the creation of more persuasive agents. *Artif. Intell. Rev.*, 17(3):169–222, 2002.
17. S. Parsons, C. Sierrra, and N. Jennings. Agents that Reason and Negotiate by Arguing. *Journal of Logic and Computation*, 8:261–292, 1998.
18. H. Prakken and G. Sartor. The role of logic in computational models of legal argument - a critical survey. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, pages 342–380. Springer, 2002.
19. H. Prakken and G. Vreeswijk. Logical Systems for Defeasible Argumentation. In D. Gabbay and F.Guenther, editors, *Handbook of Philosophical Logic*, pages 219–318. Kluwer Academic Publishers, 2002.
20. L. Richard Ye and Paul E. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Q.*, 19(2):157–172, 1995.

Characterizing defeat graphs where argumentation semantics agree

Pietro Baroni and Massimiliano Giacomin

Dip. Elettronica per l'Automazione, University of Brescia,
via Branze 38, 25123 Brescia, Italy
baroni,giacomin@ing.unibs.it

Abstract. In the context of Dung's theory of argumentation frameworks, comparisons between argumentation semantics are often focused on the different behavior they show in some (more or less peculiar) cases. It is also interesting however to characterize situations where (under some reasonably general assumptions) different semantics behave exactly in the same way. Focusing on the general family of SCC-recursive argumentation semantics, the paper provides some novel results in this line. In particular, we study the characterization of defeat graphs where any SCC-recursive semantics admits exactly one extension coinciding with the grounded extension. Then, we consider the problem of agreement with stable semantics and identify the family of SCC-symmetric argumentation frameworks, where agreement is ensured for a class of multiple-status argumentation semantics including stable, preferred and $CF2$ semantics.

Key words: Argumentation semantics, Argumentation frameworks, Semantics comparison

1 Introduction

Interest in comparing argumentation semantics arises from the increasing variety of approaches proposed in the context of Dung's theory of argumentation frameworks [1]. Different behaviors exhibited by alternative semantics in specific cases (or families of cases) have often been the subject of detailed analyses and discussions about the "most intuitive" or "desired" outcome. While this is, by far, the most common kind of comparison found in the literature, a more systematic approach considering general principles that may or may not be satisfied by a semantics has also been addressed [2, 3].

A complementary kind of analysis concerns identifying situations where argumentation semantics agree, i.e. exhibit the same behavior in spite of their differences. This can be useful from several viewpoints. On one hand, situations where "most" (or even all) existing semantics agree can be regarded as providing a sort of reference behavior against which further proposals should be confronted. On the other hand, it may be the case that in a specific application domain there are some restrictions on the structure of the argumentation frameworks that need to be considered. It is then surely interesting to know whether

these restrictions lead to semantics agreement, since in this case it is clear that evaluations about arguments in that domain may not be affected by different choices of argumentation semantics and are, in a sense, universally supported.

In fact, the question of semantics agreement for particular classes of argumentation frameworks is explicitly considered in Dung's original paper [1] where sufficient conditions for agreement between grounded, preferred and stable semantics and between preferred and stable semantics are provided (these results will be recalled along the paper). More recently, the special class of symmetric argumentation frameworks [4] (where every attack is mutual) has been shown to ensure agreement between preferred, stable and naive semantics. The present paper provides some new results in this area by considering the recently introduced class of SCC-recursive semantics [5], namely a parametric family of semantics which has been shown to represent a quite general well-founded scheme where specific proposals, including all traditional semantics mentioned above, can be placed. In this context we obtain a characterization of some cases of agreement, by exploiting the decomposition of the defeat graph into strongly connected components.

The paper is organized as follows. After reviewing the necessary basic concepts in Section 2, the notions of strongly connected component (SCC) and SCC-recursiveness are introduced in Section 3. In section 4 the definition of $CF2$ semantics is recalled and a property of its extensions, as significant for the sequel of the paper, is proved. The issues of agreement with grounded and stable semantics are dealt with in Sections 5 and 6 respectively. Finally Section 7 concludes the paper.

2 Basic concepts

The present work lies in the frame of the general theory of abstract argumentation frameworks proposed by Dung [1].

Definition 1. *An argumentation framework is a pair* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$*, where* $\mathcal{A}$ *is a set, and* $\rightarrow \subseteq (\mathcal{A} \times \mathcal{A})$ *is a binary relation on* $\mathcal{A}$*, called attack relation.*

In the following we will always assume that $\mathcal{A}$ is finite. An argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ can be represented as a directed graph, called *defeat graph*, where nodes are the arguments and edges correspond to the elements of the attack relation. In the following, the nodes that attack a given argument α are called *defeaters* or *parents* of α and form a set which is denoted as $\mathrm{par}_{\mathrm{AF}}(\alpha)$.

Definition 2. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *and a node* $\alpha \in \mathcal{A}$*, we define* $\mathrm{par}_{\mathrm{AF}}(\alpha) = \{\beta \in \mathcal{A} \mid \beta \rightarrow \alpha\}$*. If* $\mathrm{par}_{\mathrm{AF}}(\alpha) = \emptyset$*, then* α *is called an* initial *node.*

Since we will frequently consider properties of sets of arguments, it is useful to extend to them the notations defined for the nodes.

Definition 3. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$*, a node* $\alpha \in \mathcal{A}$ *and two sets* $S, P \subseteq \mathcal{A}$*, we define:*

$$\begin{aligned} S \rightarrow \alpha &\equiv \exists \beta \in S : \beta \rightarrow \alpha \\ \alpha \rightarrow S &\equiv \exists \beta \in S : \alpha \rightarrow \beta \\ S \rightarrow P &\equiv \exists \alpha \in S, \beta \in P : \alpha \rightarrow \beta \end{aligned}$$

Two particular kinds of elementary argumentation frameworks need to be introduced as they will play some role in the following. The *empty argumentation framework*, denoted as $\mathrm{AF}_\emptyset$, is simply defined as $\mathrm{AF}_\emptyset = \langle \emptyset, \emptyset \rangle$. Furthermore, an argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ is *monadic* if $|\mathcal{A}| = 1$ and $\rightarrow = \emptyset$.

The notion of self-defeating argument will be used too.

Definition 4. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *an argument* $\alpha \in \mathcal{A}$ *is* self-defeating *if* $\alpha \rightarrow \alpha$*. An argumentation framework* AF *is* free of self-defeating arguments *if* $\nexists \alpha \in \mathcal{A}$ *such that* $\alpha \rightarrow \alpha$*.*

We will also consider the *restriction* of an argumentation framework to a given subset of its nodes:

Definition 5. *Let* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *be an argumentation framework, and let* $S \subseteq \mathcal{A}$ *be a set of arguments. The* restriction *of* AF *to* S *is the argumentation framework* $\mathrm{AF}{\downarrow_S} = \langle S, \rightarrow \cap (S \times S) \rangle$*.*

In Dung's theory, an argumentation semantics is defined by specifying the criteria for deriving, given a generic argumentation framework, the set of all possible extensions, each one representing a set of arguments considered to be acceptable together. Accordingly, a basic requirement for any extension E is that it is *conflict-free*, namely $\nexists \alpha, \beta \in E : \alpha \rightarrow \beta$. All argumentation semantics proposed in the literature satisfy this fundamental *conflict-free property*.

Given a generic argumentation semantics $\mathcal{S}$, the set of extensions prescribed by $\mathcal{S}$ for a given argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ is denoted as $\mathcal{E}_\mathcal{S}(\mathrm{AF})$. If it holds that $\forall \mathrm{AF}\ |\mathcal{E}_\mathcal{S}(\mathrm{AF})| = 1$, then the semantics $\mathcal{S}$ is said to follow the *unique-status approach*, otherwise it is said to follow the *multiple-status approach* [6]. We will say that two semantics $\mathcal{S}_1$ and $\mathcal{S}_2$ are in agreement on an argumentation framework AF if $\mathcal{E}_{\mathcal{S}_1}(\mathrm{AF}) = \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF})$.

3 Strongly connected components and SCC-recursiveness

SCC-recursiveness is a property of (the extensions prescribed by) a semantics based on the graph theoretical notion of *strongly connected components* (SCCs).

Definition 6. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$*, the binary relation of path-equivalence between nodes, denoted as* $PE_\mathrm{AF} \subseteq (\mathcal{A} \times \mathcal{A})$*, is defined as follows:*

- $\forall \alpha \in \mathcal{A}, (\alpha, \alpha) \in PE_\mathrm{AF}$

– *given two distinct nodes* $\alpha, \beta \in \mathcal{A}$, $(\alpha, \beta) \in PE_{\mathrm{AF}}$ *if and only if there is a path from* α *to* β *and a path from* β *to* α.

The *strongly connected components* of AF are the equivalence classes of nodes under the relation of path-equivalence. The set of the SCCs of AF is denoted as $\mathrm{SCCS}_{\mathrm{AF}}$. In the case of the empty argumentation framework, we assume $\mathrm{SCCS}_{\mathrm{AF}_\emptyset} = \{\emptyset\}$. Moreover, a strongly connected component $S \in \mathrm{SCCS}_{\mathrm{AF}}$ will be said to be monadic if $\mathrm{AF}{\downarrow_S}$ is monadic.

We extend to SCCs the notion of parents, namely the set of the other SCCs that attack a SCC S, which is denoted as $\mathrm{sccpar}_{\mathrm{AF}}(S)$, and we introduce the definition of *proper ancestors*, denoted as $\mathrm{sccanc}_{\mathrm{AF}}(S)$:

Definition 7. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *and a SCC* $S \in \mathrm{SCCS}_{\mathrm{AF}}$*, we define*

$$\mathrm{sccpar}_{\mathrm{AF}}(S) = \{P \in \mathrm{SCCS}_{\mathrm{AF}} \mid P \neq S \text{ and } P \rightarrow S\}$$

and

$$\mathrm{sccanc}_{\mathrm{AF}}(S) = \mathrm{sccpar}_{\mathrm{AF}}(S) \cup \bigcup_{P \in \mathrm{sccpar}_{\mathrm{AF}}(S)} \mathrm{sccanc}_{\mathrm{AF}}(P)$$

A SCC S such that $\mathrm{sccpar}_{\mathrm{AF}}(S) = \emptyset$ is called *initial*. The set of initial SCCs of AF, as it is easy to see, is non-empty and is denoted as $\mathcal{IS}(\mathrm{AF})$. The set of nodes of initial strongly connected components of AF is denoted as $IN(\mathrm{AF}) = \bigcup_{S \in \mathcal{IS}(\mathrm{AF})} S$.

It is well-known [7] that the graph obtained by considering SCCs as single nodes is acyclic, in other words SCCs can be partially ordered according to the relation of attack. This fact lies at the heart of the definition of SCC-recursiveness, which is based on the intuition that extensions can be built incrementally starting from initial SCCs and following the above mentioned partial order. In other words, the choices concerning extension construction carried out in an initial SCC do not depend on the choices concerning any other SCC, while they directly affect the choices about the subsequent SCCs and so on. While the basic underlying intuition is rather simple, the formalization of SCC-recursiveness is admittedly rather complex and involves some additional notions. Due to space limitations, we can only give here a quick account, while referring the reader to [5] for more details and examples. First of all, the choices (represented in the following definition by the set E, corresponding to a specific extension) in the antecedent SCCs determine a partition of the nodes of a set S (typically representing one or more subsequent SCCs) into three subsets:

Definition 8. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$*, a set* $E \subseteq \mathcal{A}$ *and a set* $S \subseteq \mathcal{A}$*, we define:*

– $D_{\mathrm{AF}}(S, E) = \{\alpha \in S \mid (E \setminus S) \rightarrow \alpha\}$
– $P_{\mathrm{AF}}(S, E) = \{\alpha \in S \mid (E \setminus S) \not\rightarrow \alpha \ \wedge \ \exists \beta \notin S : \beta \rightarrow \alpha \wedge E \not\rightarrow \beta\}$
– $U_{\mathrm{AF}}(S, E) = S \setminus (D_{\mathrm{AF}}(S, E) \cup P_{\mathrm{AF}}(S, E)) =$
$= \{\alpha \in S \mid (E \setminus S) \not\rightarrow \alpha \wedge \forall \beta \notin S : \beta \rightarrow \alpha \ E \rightarrow \beta\}$

Definition 8 is a slightly generalized version of the corresponding Definition 18 of [5]. In words, the set $D_{\mathrm{AF}}(S,E)$ consists of the nodes of S attacked by E from outside S, the set $U_{\mathrm{AF}}(S,E)$ includes any node α of S that is not attacked by E from outside S and is defended by E (i.e. the defeaters of α from outside S are all attacked by E), and $P_{\mathrm{AF}}(S,E)$ includes any node α of S that is not attacked by E from outside S and is not defended by E (i.e. at least one of the defeaters of α from outside S is not attacked by E). It is easy to verify that, when S is a SCC, as in the original Definition 18 of [5], $D_{\mathrm{AF}}(S,E)$, $P_{\mathrm{AF}}(S,E)$ and $U_{\mathrm{AF}}(S,E)$ are determined only by the elements of E that belong to the SCCs in $\mathrm{sccanc}_{\mathrm{AF}}(S)$.

Regarding E as a part of an extension which is being constructed, the idea is then that arguments in $D_{\mathrm{AF}}(S,E)$, being attacked by nodes in E, cannot be chosen in the construction of the extension E (i.e. do not belong to $E \cap S$). Selection of arguments to be included in E is therefore restricted to $(S \setminus D_{\mathrm{AF}}(S,E)) = (U_{\mathrm{AF}}(S,E) \cup P_{\mathrm{AF}}(S,E))$, which, for ease of notation, will be denoted in the following as $UP_{\mathrm{AF}}(S,E)$. On this basis and taking also into account the reinstatement principle [6, 2], we require the selection of nodes within a SCC S to be carried out on the restricted argumentation framework $\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}$ without taking into account the attacks coming from $D_{\mathrm{AF}}(S,E)$.

Combining these ideas and skipping some details not strictly necessary in the context of the present paper, we can finally recall the definition of *SCC-recursiveness*:

Definition 9. *A given argumentation semantics $\mathcal{S}$ is* SCC-recursive *if and only if for any argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$, $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \mathcal{GF}(\mathrm{AF}, \mathcal{A})$, where for any $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ and for any set $C \subseteq \mathcal{A}$, the function $\mathcal{GF}(\mathrm{AF}, C) \subseteq 2^{\mathcal{A}}$ is defined as follows:*
for any $E \subseteq \mathcal{A}$, $E \in \mathcal{GF}(\mathrm{AF}, C)$ if and only if

- *in case $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $E \in \mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, C)$*
- *otherwise, $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$*
  $(E \cap S) \in \mathcal{GF}(\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}, U_{\mathrm{AF}}(S,E) \cap C)$

where $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, C)$ is a function, called base function*, that, given an argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ such that $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$ and a set $C \subseteq \mathcal{A}$, gives a subset of $2^{\mathcal{A}}$.*

The base function $\mathcal{BF}_{\mathcal{S}}$ of a SCC-recursive semantics $\mathcal{S}$ is said to be conflict-free if $\forall \mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ and $\forall C \subseteq \mathcal{A}$ each element of $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, C)$ is conflict free. It is known from Theorem 48 of [5] that if $\mathcal{BF}_{\mathcal{S}}$ is conflict-free, then any $E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ is conflict free for any AF.

Since Definition 9 is somewhat arduous to examine in its full detail, we just give some "quick and dirty" indications which are useful for the sequel of the paper (in particular, we do not consider the meaning of the parameter C in the description, as not necessary for the comprehension of this paper). The set of extensions $\mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ of an argumentation framework AF is given by $\mathcal{GF}(\mathrm{AF}, \mathcal{A})$, namely by the invocation of the function $\mathcal{GF}$ which receives as parameters an

argumentation framework (in this case the whole AF) and a set of arguments (in this case the whole $\mathcal{A}$). The function $\mathcal{GF}(\mathrm{AF}, C)$ is defined recursively. The base of the recursion is reached when AF consists of a unique SCC: in this case the set of extensions is directly given by the invocation of a semantics-specific base function $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, C)$. In the other case, for each SCC S of AF the function $\mathcal{GF}$ is invoked recursively on the restriction $\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}$. Note that the restriction concerns $UP_{\mathrm{AF}}(S, E)$, namely the part of S which "survives" the attacks of the preceding SCCs in the partial order.

The definition has also a constructive interpretation, which suggests an effective (recursive) procedure for computing all the extensions of an argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ once a specific base function characterizing the semantics is assigned. A particular role in this context is played by the initial SCCs. In fact, for any initial SCC I, since by definition there are no outer attacks, the set of defended nodes coincides with I, i.e. $UP_{\mathrm{AF}}(I, E) = U_{\mathrm{AF}}(I, E) = I$ for any E. This gives rise to the invocation $\mathcal{GF}(\mathrm{AF}{\downarrow_I}, I)$ for any initial SCC I. Since $\mathrm{AF}{\downarrow_I}$ obviously consists of a unique SCC, according to Definition 9 the base function $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}{\downarrow_I}, I)$ is invoked, which returns the extensions of $\mathrm{AF}{\downarrow_I}$ according to the semantics $\mathcal{S}$. Therefore, the base function can be first computed on the initial SCCs, where it directly returns the extensions prescribed by the semantics. Then, the results of this computation are used to identify, within the subsequent SCCs, the restricted argumentation frameworks on which the procedure is recursively invoked.

All SCC-recursive semantics "share" this general scheme and only differ by the specific base function adopted. It has been shown [5] that all traditional semantics encompassed by Dung's framework (namely grounded, stable, complete, and preferred semantics) are SCC-recursive and the relevant base functions have been identified. In the following we will assume a basic knowledge of grounded semantics, denoted as $\mathcal{GR}$, stable semantics, denoted as $\mathcal{ST}$, and preferred semantics, denoted as $\mathcal{PR}$. We need to recall here only the formulation of the base function of grounded semantics (Proposition 44 of [5]):

Proposition 1. *For any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *such that* $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$*, and for any* $C \subseteq \mathcal{A}$*, we have that*

$$\mathcal{BF}_{\mathcal{GR}}(\mathrm{AF}, C) = \begin{cases} \{\{\alpha\}\}, & \text{if } C = \mathcal{A} = \{\alpha\} \text{ and } \rightarrow = \emptyset; \\ \{\emptyset\}, & \text{otherwise.} \end{cases}$$

It is well-known that grounded semantics belongs to the unique-status approach. In the following we will denote the grounded extension of an argumentation framework AF as GE(AF).

4 A property of *CF2* semantics

Besides encompassing many significant previous proposals, the SCC-recursive scheme allows the definition of novel semantics in a relatively easy way. Examples of non-traditional SCC-recursive semantics and their properties are discussed in

[5], the most significant among them being $CF2$ semantics. In fact, $CF2$ semantics exhibits rather interesting properties (in particular a "symmetric" treatment of odd- and even-length cycles [8]) while its base function is particularly simple: $\mathcal{BF}_{CF2}(\mathrm{AF}, C) = \mathcal{MCF}_{\mathrm{AF}}$, where $\mathcal{MCF}_{\mathrm{AF}}$ denotes the set made up of all the maximal conflict-free sets of AF (note that the parameter C of Definition 9 plays no role at all in this case).

As a first contribution of this paper, here we provide the proof of an important property of $CF2$ semantics which, in particular, will be useful for the characterization of the cases of agreement between $CF2$ and grounded semantics. In words, we will show that any extension prescribed by $CF2$ semantics for an argumentation framework AF is a maximal conflict free set of AF.

A preliminary Lemma is needed.

Lemma 1. *Given an argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *and a conflict free set* $E \subseteq \mathcal{A}$, $E \in \mathcal{MCF}_{\mathrm{AF}} \Leftrightarrow \forall \alpha \in \mathcal{A}$ *such that* $\alpha \not\rightarrow \alpha$, *the following disjunction of mutually exclusive conditions holds:* $\alpha \in E \vee E \rightarrow \alpha \vee \alpha \rightarrow E$.

Proof. $\Rightarrow$. Assume that $\exists \alpha \in \mathcal{A}$, $\alpha \not\rightarrow \alpha$, such that none of the three condition stated above holds. Then $\alpha \notin E \wedge E \not\rightarrow \alpha \wedge \alpha \not\rightarrow E$, which implies that $E \cup \{\alpha\}$ is conflict-free and a strict superset of E. But this contradicts the hypothesis that $E \in \mathcal{MCF}_{\mathrm{AF}}$.
$\Leftarrow$. Conversely assume that $E \notin \mathcal{MCF}_{\mathrm{AF}}$, then $\exists \alpha \in \mathcal{A}$ such that $\alpha \notin E$ and $E \cup \{\alpha\}$ is conflict-free, namely $\alpha \not\rightarrow \alpha$, $\alpha \notin E \wedge E \not\rightarrow \alpha \wedge \alpha \not\rightarrow E$, contradicting the hypothesis that one of the three conditions above holds.

Proposition 2. *For any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$, $\mathcal{E}_{CF2}(\mathrm{AF}) \subseteq \mathcal{MCF}_{\mathrm{AF}}$.

Proof. Since the base function of $CF2$ semantics is conflict-free, we know that any $E \in \mathcal{E}_{CF2}(\mathrm{AF})$ is conflict free. We have now to prove that it is maximal. First, recall that instantiating Definition 9 in the case of $CF2$ semantics we obtain: $E \in \mathcal{E}_{CF2}(\mathrm{AF})$ if and only if

- in case $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $E \in \mathcal{MCF}_{\mathrm{AF}}$
- otherwise, $\forall S \in \mathrm{SCCS}_{\mathrm{AF}} (E \cap S) \in \mathcal{E}_{CF2}(\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}})$

If $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $\mathcal{E}_{CF2}(\mathrm{AF}) = \mathcal{MCF}_{\mathrm{AF}}$ by definition and the thesis trivially follows.

Consider now the case $|\mathrm{SCCS}_{\mathrm{AF}}| > 1$ and assume recursively that $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $\forall E \in \mathcal{E}_{CF2}(\mathrm{AF})$ $(E \cap S) \in \mathcal{MCF}_{\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}}$: we need to prove that $E \in \mathcal{MCF}_{\mathrm{AF}}$. Suppose by contradiction that $E \notin \mathcal{MCF}_{\mathrm{AF}}$. By Lemma 1 the following condition (i) holds: $\exists \alpha \in \mathcal{A} : \alpha \not\rightarrow \alpha$, $\alpha \notin E \wedge E \not\rightarrow \alpha \wedge \alpha \not\rightarrow E$. Now consider the strongly connected component $S \in \mathrm{SCCS}_{\mathrm{AF}}$ such that $\alpha \in S$. Since $E \not\rightarrow \alpha$ it is the case that $\alpha \in UP_{\mathrm{AF}}(S, E)$. By the inductive hypothesis $(E \cap S) \in \mathcal{MCF}_{\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}}$, which, by Lemma 1 applied to $\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}$, entails that the following disjunction holds: $\alpha \in (E \cap S) \vee (E \cap S) \rightarrow \alpha \vee \alpha \rightarrow (E \cap S)$. This clearly implies the following condition in AF: $\alpha \in E \vee E \rightarrow \alpha \vee \alpha \rightarrow E$. However, this is absurd since it contradicts condition (i) above.

5 Agreement with grounded semantics

Grounded semantics [9, 1] plays an important role in argumentation theory as it features desirable properties, such as conceptual clarity and computational tractability. Moreover, it is often regarded as a paradigmatic unique-status sceptical approach that can be used as a reference to evaluate other semantics. For these reasons the issue of agreement with grounded semantics is particularly significant and has been first considered in [1], where it is shown that a sufficient condition for agreement between grounded, preferred and stable semantics is that the argumentation framework is well-founded.

Definition 10. *(Definition 29 of [1]) An argumentation framework is well-founded iff there exists no infinite sequence* $\alpha_0, \alpha_1, \ldots, \alpha_n, \ldots$ *of (not necessarily distinct) arguments such that for each* i, α_{i+1} *attacks* α_i.

In the case of a finite argumentation framework, well-foundedness coincides with acyclicity of the defeat graph. We now consider the problem of agreement with grounded semantics in the generalized context of SCC-recursive semantics.

5.1 Determined argumentation frameworks

We will show that a complete agreement among SCC-recursive semantics holds if and only if the considered argumentation framework is *determined*.

Definition 11. *An argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *is* determined *if and only if* $\nexists \alpha \in \mathcal{A} : \alpha \notin \mathrm{GE(AF)} \wedge \mathrm{GE(AF)} \not\rightarrow \alpha$.

In words, an argumentation framework AF is determined if and only if there are no "provisionally defeated" arguments in AF according to grounded semantics, i.e. the grounded extension is also a stable extension. Note that the empty argumentation framework is determined.

The set of determined argumentation frameworks, denoted as $\mathcal{DET}$, is of special interest because for any SCC-recursive semantics $\mathcal{S}$ respecting an obvious condition on the treatment of monadic argumentation frameworks it holds that $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \{\mathrm{GE(AF)}\}$ for any argumentation framework $\mathrm{AF} \in \mathcal{DET}$. In other words, a very comprehensive family of "reasonable" semantics show a uniform single-status behavior on these argumentation frameworks.

Proposition 3. *Let* $\mathcal{S}$ *be a SCC-recursive semantics identified by a conflict-free base function* $\mathcal{BF}_{\mathcal{S}}$ *such that*

$$\mathcal{BF}_{\mathcal{S}}(\langle \{\alpha\}, \emptyset \rangle, \{\alpha\}) = \{\{\alpha\}\}$$

(such a SCC-recursive semantics will be called grounded-compatible*).*

For any argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle \in \mathcal{DET}$ *it holds that* $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \{\mathrm{GE(AF)}\}$.

Proof. The proof immediately follows from the fact that for any such SCC-recursive semantics it holds that $\forall E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$, $\mathrm{GE}(\mathrm{AF}) \subseteq E$ (Proposition 51 of [5]) and E is conflict-free. Since $\mathrm{AF} \in \mathcal{DET}$ $\forall \alpha \notin \mathrm{GE}(\mathrm{AF})$ it necessarily holds that $\mathrm{GE}(\mathrm{AF}) \rightarrow \alpha$, and therefore $\alpha \notin E$. As a consequence, only the case $E = \mathrm{GE}(\mathrm{AF})$ is possible.

It is also immediate to note that no argumentation framework outside $\mathcal{DET}$ features this property, namely, for any $\mathrm{AF} \notin \mathcal{DET}$ there is a grounded-compatible SCC-recursive semantics $\mathcal{S}$ such that $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) \neq \{\mathrm{GE}(\mathrm{AF})\}$, namely stable semantics.

Well-founded argumentation frameworks [1] are a special case of determined argumentation frameworks. In fact, if no cycles are present, all SCCs in AF consist of a single node and it is then easy to see that $\mathrm{AF} \in \mathcal{DET}$. On the other hand, the absence of cycles is a sufficient but not necessary topological condition for $\mathrm{AF} \in \mathcal{DET}$. Actually the absence of cycles is necessary only in the initial SCCs (which need to be monadic), and then recursively in the initial SCCs of the restricted argumentation framework obtained by taking into account that the nodes corresponding to the initial SCCs are necessarily included in any extension. This observation gives rise to a characterization of determined argumentation frameworks.

Definition 12. *An argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *is initial-acyclic if* $\mathrm{AF} = \mathrm{AF}_{\emptyset}$ *or the following condition holds:* $\forall S \in \mathcal{IS}(\mathrm{AF})$ S *is monadic and* $\mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}((\mathcal{A} \setminus IN(\mathrm{AF})), IN(\mathrm{AF}))}$ *is initial-acyclic.*

The base of this recursive definition is represented by the empty argumentation framework. The recursion is well-founded as the set $IN(\mathrm{AF})$ is non-empty for a non-empty argumentation framework, which means that at each recursive step an argumentation framework with a strictly lesser number of nodes is considered. The set of initial-acyclic argumentation frameworks is denoted by $\mathcal{IAA}$. The following proposition shows that $\mathcal{IAA} = \mathcal{DET}$.

Proposition 4. *For any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$, $\mathrm{AF} \in \mathcal{IAA}$ *if and only if* $\mathrm{AF} \in \mathcal{DET}$.

Proof. Let us first show that if $\mathrm{AF} \in \mathcal{IAA}$ then the grounded extension is also stable. It is known [1] that, for any finite AF, $\mathrm{GE}(\mathrm{AF}) = \bigcup_{i \geq 1} \mathrm{F}^i_{\mathrm{AF}}(\emptyset)$, where, given a set $S \subseteq \mathcal{A}$, $\mathrm{F}_{\mathrm{AF}}(S) = \{\alpha \in \mathcal{A} : \forall \beta \in \mathrm{par}_{\mathrm{AF}}(\alpha), S \rightarrow \beta\}$, $\mathrm{F}^1_{\mathrm{AF}}(S) = \mathrm{F}_{\mathrm{AF}}(S)$, and $\mathrm{F}^i_{\mathrm{AF}}(S) = \mathrm{F}_{\mathrm{AF}}(\mathrm{F}^{i-1}_{\mathrm{AF}}(S))$. Now, since $\mathrm{AF} \in \mathcal{IAA}$, it holds that $\mathrm{F}^1_{\mathrm{AF}}(\emptyset) = IN(\mathrm{AF})$. After suppressing the arguments attacked by arguments in $IN(\mathrm{AF})$ we obtain $\mathrm{AF}' = \mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}((\mathcal{A} \setminus IN(\mathrm{AF})), IN(\mathrm{AF}))}$. Now, if AF' is empty the statement is proved, since any argument of AF is either included in or attacked by GE(AF). Otherwise we have, by hypothesis, that all initial strongly connected components of AF' are monadic. This entails that all their nodes belong to $\mathrm{F}^2_{\mathrm{AF}}(\emptyset)$ and therefore to GE(AF). Iterating the same reasoning as above we obtain a restricted argumentation framework AF'', and so on until we reach the case of an empty restricted argumentation framework. Since any

node considered at any step is either included in or attacked by GE(AF), it turns out that AF $\in \mathcal{DET}$.

Turning to the other part of the proof, let us show that if AF $\notin \mathcal{IAA}$ then AF $\notin \mathcal{DET}$. Let us first consider the case where some initial strongly connected component of AF is not monadic, then its elements are not included in nor attacked by GE(AF) and therefore AF $\notin \mathcal{DET}$. Otherwise with a similar reasoning as in the first part of the proof, we are lead to consider a sequence of restricted argumentation frameworks. Since at least one of them does not belong to $\mathcal{IAA}$, it turns out as before that some of its nodes are not included in nor attacked by GE(AF) and the conclusion follows.

5.2 Almost determined argumentation frameworks

While only determined argumentation frameworks ensure complete agreement among all grounded-compatible SCC-recursive semantics, it can be observed that there is a larger class of argumentation frameworks where an almost complete agreement is reached. Consider for instance the case of an argumentation framework consisting just of a self-defeating argument, namely AF $= \langle\{\alpha\}, \{(\alpha, \alpha)\}\rangle$. In this case we have that $\mathcal{E}_{\mathcal{GR}}(\text{AF}) = \{\emptyset\}$ and, in virtue of the conflict-free property, for any semantics $\mathcal{S}$ which admits extensions on AF it must also hold that $\mathcal{E}_{\mathcal{S}}(\text{AF}) = \{\emptyset\}$. However, since stable semantics is unable to prescribe estensions in this case, $\mathcal{E}_{\mathcal{ST}}(\text{AF}) = \emptyset \neq \{\emptyset\}$. In this case, disagreement arises from the non-existence of stable extensions rather than from the existence of extensions different from GE(AF). Therefore, excluding AF from the set of argumentation frameworks where semantics agree might be considered a little bit questionable and/or misleading, since, actually, all semantics able to prescribe extensions for AF are in agreement.

On the basis of this observation, it is useful to consider the question of agreement focusing on those semantics that are *universally defined*.

Definition 13. *An argumentation semantics $\mathcal{S}$ is universally defined if for any argumentation framework* AF $\mathcal{E}_{\mathcal{S}}(\text{AF}) \neq \emptyset$.

As to our knowledge, stable semantics is the only example in the literature of a semantics which is not universally defined.

As shown by the simple example above, the set of argumentation frameworks where universally defined semantics agree is larger than $\mathcal{DET}$: we will now characterize this class of argumentation frameworks, called *almost determined*.

Definition 14. *An argumentation framework* AF $= \langle\mathcal{A}, \rightarrow\rangle$ *is* almost determined *if and only if for any* $\alpha \in \mathcal{A}$, $(\alpha \notin \text{GE(AF)} \wedge \text{GE(AF)} \not\rightarrow \alpha) \Rightarrow (\alpha, \alpha) \in \rightarrow$.

In words, an argumentation framework is almost determined if all the nodes which are not attacked nor included in the grounded extension are self-defeating. The set of almost determined argumentation frameworks will be denoted as $\mathcal{AD}$. Clearly $\mathcal{DET} \subsetneq \mathcal{AD}$.

Proposition 5. *Let $\mathcal{S}$ be a universally defined and grounded compatible SCC-recursive semantics identified by a conflict-free base function $\mathcal{BF}_{\mathcal{S}}$. For any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle \in \mathcal{AD}$ *it holds that* $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \{\mathrm{GE}(\mathrm{AF})\}$.

Proof. We know that, since $\mathcal{BF}_{\mathcal{S}}$ is conflict-free, for any argumentation framework AF, $\forall E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ E is conflict free. Then, the statement follows from the fact that $\forall E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$, $\mathrm{GE}(\mathrm{AF}) \subseteq E$, which entails that the arguments attacked by the grounded extension are also attacked by any other extension. Therefore only arguments not included in and not attacked by GE(AF) can belong to $E \setminus \mathrm{GE}(\mathrm{AF})$. However, by hypothesis such arguments are self-defeating and, since any extension E is conflict-free, can not belong to E.

The proposition above shows that agreement is ensured on almost determined argumentation frameworks for any SCC-recursive semantics which satisfies the three very reasonable properties of being universally defined, grounded compatible and conflict-free. We now also show that such an agreement can not be achieved outside the class of almost determined argumentation frameworks.

Proposition 6. *For any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle \notin \mathcal{AD}$ *there is a universally defined and grounded compatible SCC-recursive semantics $\mathcal{S}$ identified by a conflict-free base function $\mathcal{BF}_{\mathcal{S}}$ such that* $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) \neq \{\mathrm{GE}(\mathrm{AF})\}$.

Proof. We prove that if $\mathrm{AF} \notin \mathcal{AD}$ then $\mathcal{E}_{CF2}(\mathrm{AF}) \neq \{\mathrm{GE}(\mathrm{AF})\}$. It is immediate to see that $CF2$ semantics is universally defined and grounded compatible and that its base function is conflict-free. By Proposition 2, $\mathcal{E}_{CF2}(\mathrm{AF}) \subseteq \mathcal{MCF}_{\mathrm{AF}}$, namely the extensions prescribed by $CF2$ semantics for an argumentation framework AF are maximal conflict free sets of AF. Now if $\mathrm{AF} \notin \mathcal{AD}$, $\exists \alpha \in \mathcal{A}$ such that α is not self-defeating, $\alpha \notin \mathrm{GE}(\mathrm{AF})$ and $\mathrm{GE}(\mathrm{AF}) \not\rightarrow \alpha$. This also implies $\alpha \not\rightarrow \mathrm{GE}(\mathrm{AF})$ due to the well-known property of admissibility of GE(AF) [1], namely $\alpha \rightarrow \mathrm{GE}(\mathrm{AF}) \Rightarrow \mathrm{GE}(\mathrm{AF}) \rightarrow \alpha$. Then, by Lemma 1, $\mathrm{GE}(\mathrm{AF}) \notin \mathcal{MCF}_{\mathrm{AF}}$ and necessarily $\mathcal{E}_{CF2}(\mathrm{AF}) \neq \{\mathrm{GE}(\mathrm{AF})\}$.

6 Agreement with stable semantics

Stable semantics represents a traditional and intuitively simple proposal among multiple-status approaches: a stable extension is simply a conflict-free set which attacks all arguments not included in it. For this reason, agreement with stable semantics represents a sort of uncontroversial situation where no argument is left in a sort of "undecided" status. In [1] an argumentation framework AF such that preferred and stable semantics are in agreement is said to be *coherent*. Here we will characterize a family of argumentation frameworks, called SCC-symmetric, where agreement is ensured for a class of multiple-status semantics including stable, preferred and $CF2$ semantics.

First we need to introduce the notion of symmetric argumentation framework (slightly different from the one proposed in [4]), noting also that symmetry is preserved by the restriction operator.

Definition 15. *An argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *is* symmetric *if for any* $\alpha, \beta \in \mathcal{A}$, $\alpha \rightarrow \beta \Leftrightarrow \beta \rightarrow \alpha$.

Lemma 2. *Given a symmetric argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *and a set* $S \subseteq \mathcal{A}$, $\mathrm{AF}{\downarrow_S}$ *is symmetric.*

Proof. Let us consider two arguments α, β in $\mathrm{AF}{\downarrow_S}$ such that $\alpha \rightarrow \beta$. It is immediate to see that this relation also holds in AF and, since the latter is symmetric, $\beta \rightarrow \alpha$ in AF. Since $\alpha, \beta \in S$, $\beta \rightarrow \alpha$ also holds in $\mathrm{AF}{\downarrow_S}$.

As it will be more evident from Proposition 7, it is quite natural that extensions of a symmetric argumentation framework free of self-defeating arguments coincide with its maximal conflict free sets, if the multiple-status approach is adopted. Argumentation semantics satisfying this requirement will be called *-symmetric.

Definition 16. *An argumentation semantics* $\mathcal{S}$ *is *-symmetric if for any argumentation framework* AF *which is symmetric and free of self-defeating arguments* $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \mathcal{MCF}_{\mathrm{AF}}$.

As one may imagine, a SCC-recursive semantics is *-symmetric if and only if its base function has a *-symmetric behavior on single-SCC argumentation frameworks.

Lemma 3. *A SCC-recursive semantics* $\mathcal{S}$ *is *-symmetric if and only if, for any argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *which is symmetric, free of self-defeating arguments and such that* $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) = \mathcal{MCF}_{\mathrm{AF}}$.

Proof. $\Rightarrow$. Assume that the base function satisfies the hypothesis and consider a generic argumentation framework AF which is symmetric and free of self-defeating arguments. Notice first that $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $\mathrm{sccpar}_{\mathrm{AF}}(S) = \emptyset$, i.e. all of the strongly connected components are initial. In fact, given $S_1, S_2 \in \mathrm{SCCS}_{\mathrm{AF}}$ such that $S_1 \rightarrow S_2$, since AF is symmetric also $S_2 \rightarrow S_1$ holds, entailing that all of the nodes of $S_1 \cup S_2$ are mutually reachable, i.e. $S_1 = S_2$. Then, $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $U_{\mathrm{AF}}(S, E) = UP_{\mathrm{AF}}(S, E) = S$, and it is easy to see that, according to Definition 9, $E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ if and only if $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $(E \cap S) = \mathcal{BF}_{\mathcal{S}}(\mathrm{AF}{\downarrow_S}, S)$. Now, $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $\mathrm{AF}{\downarrow_S}$ is free of self-defeating arguments and by Lemma 2 is also symmetric, thus by the hypothesis $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}{\downarrow_S}, S) = \mathcal{MCF}_{\mathrm{AF}{\downarrow_S}}$. In sum, we have that $E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ if and only if $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $(E \cap S) \in \mathcal{MCF}_{\mathrm{AF}{\downarrow_S}}$, and since all of the strongly connected components are initial the latter condition is equivalent to $E \in \mathcal{MCF}_{\mathrm{AF}}$.
$\Leftarrow$. Assuming by contradiction that the conclusion is not verified we are led to consider an argumentation framework AF, which is symmetric and free of self-defeating arguments, where $\mathcal{E}_{\mathcal{S}}(\mathrm{AF}) = \mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) \neq \mathcal{MCF}_{\mathrm{AF}}$, entailing that $\mathcal{S}$ is not *-symmetric.

Several significant multiple-status semantics, though their definition is based on quite different principles, share the property of being *-symmetric (a similar result is proved in [4]).

Proposition 7. *Stable semantics, preferred semantics and $CF2$ semantics are *-symmetric.*

Proof. According to Lemma 3, for any such semantics $\mathcal{S}$ we have to prove that, given an argumentation framework $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ which is symmetric, free of self-defeating arguments and such that $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) = \mathcal{MCF}_{\mathrm{AF}}$. For $CF2$ semantics this holds by definition. As for stable and preferred semantics, notice that, as $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$, $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) = \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$. Taking into account from [1] that $\mathcal{E}_{\mathcal{ST}}(\mathrm{AF}) \subseteq \mathcal{E}_{\mathcal{PR}}(\mathrm{AF})$, it is sufficient to prove that $\mathcal{MCF}_{\mathrm{AF}} \subseteq \mathcal{E}_{\mathcal{ST}}(\mathrm{AF})$ and that $\mathcal{E}_{\mathcal{PR}}(\mathrm{AF}) \subseteq \mathcal{MCF}_{\mathrm{AF}}$. First, let us consider a set $E \in \mathcal{MCF}_{\mathrm{AF}}$ and let us prove that it is a stable extension, i.e. that $\forall \alpha \notin E \; E \rightarrow \alpha$. Assuming by contradiction that $E \not\rightarrow \alpha$, since AF is symmetric also $\alpha \not\rightarrow E$ holds. Since α cannot be self-defeating by the hypothesis on AF, the set $E \cup \{\alpha\}$ is conflict-free, contradicting the fact that $E \in \mathcal{MCF}_{\mathrm{AF}}$. Let us turn now to the other inclusion condition, considering a set $E \in \mathcal{E}_{\mathcal{PR}}(\mathrm{AF})$ and assuming by contradiction that $E \notin \mathcal{MCF}_{\mathrm{AF}}$: since E is conflict-free, this entails that $\exists E' \subseteq \mathcal{MCF}_{\mathrm{AF}}$ such that $E \subsetneq E'$. However, by the first inclusion condition $E' \in \mathcal{E}_{\mathcal{PR}}(\mathrm{AF})$, contradicting the fact that E is a preferred extension.

In symmetric argumentation frameworks non-mutual attacks cannot exist: this seriously limits their applicability for modeling practical situations. Their properties however provide the basis for analyzing a more interesting family of argumentation frameworks called *SCC-symmetric*.

Definition 17. *An argumentation framework* AF *is SCC-symmetric if* $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $\mathrm{AF}{\downarrow_S}$ *is symmetric.*

Definition 17 is equivalent to forbidding non-mutual attacks only within cycles.

Proposition 8. *An argumentation framework* $\mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *is SCC-symmetric if and only if for every cycle* $\alpha_0 \rightarrow \alpha_1 \rightarrow \ldots \rightarrow \alpha_n \rightarrow \alpha_0$ *it holds that* $\forall i \in \{1 \ldots n\}$ $\alpha_i \rightarrow \alpha_{i-1}$.

Proof. As for the if part of the proof, notice that any two nodes $\alpha, \beta \in S$, such that $\alpha \neq \beta$ and $S \in \mathrm{SCCS}_{\mathrm{AF}}$, are mutually reachable, therefore in particular they belong to a cycle. As a consequence, if $\alpha \rightarrow \beta$ then by the hypothesis also $\beta \rightarrow \alpha$ holds. As for the other part of the proof, if α_i and α_{i-1} belong to a cycle then they are in the same strongly connected component, thus if $\alpha_{i-1} \rightarrow \alpha_i$ then by the SCC-symmetry of AF also $\alpha_i \rightarrow \alpha_{i-1}$ holds.

To prove, in Theorem 1, the main result about agreement in SCC-symmetric argumentation frameworks, we need a preliminary lemma concerning the SCC-recursive schema.

Lemma 4. *Given an SCC-recursive semantics* $\mathcal{S}$, $E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ *if and only if* $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}$ $(E \cap S) \in \mathcal{GF}_{\mathcal{S}}(\mathrm{AF}{\downarrow_{UP_{\mathrm{AF}}(S,E)}}, U_{\mathrm{AF}}(S, E))$, *where* $\mathcal{GF}_{\mathcal{S}}(\mathrm{AF}, C)$ *is a function specific for the semantics* $\mathcal{S}$. *Moreover,* $\forall \mathrm{AF} = \langle \mathcal{A}, \rightarrow \rangle$ *it holds that* $\mathcal{GF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) = \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$.

Proof. By Definition 9, if $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$ then $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A}) = \mathcal{GF}(\mathrm{AF}, \mathcal{A})$, which is also equal to $\mathcal{GF}(\mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$ since in this case $UP_{\mathrm{AF}}(S,E) = U_{\mathrm{AF}}(S,E) = \mathcal{A}$. From Definition 9 we have that $E \in \mathcal{E}_{\mathcal{S}}(\mathrm{AF})$ if and only if $E \in \mathcal{GF}(\mathrm{AF}, \mathcal{A})$ and in case $|\mathrm{SCCS}_{\mathrm{AF}}| = 1$ we can substitute $\mathcal{BF}_{\mathcal{S}}(\mathrm{AF}, \mathcal{A})$ with the expression above. This yields $E \in \mathcal{GF}(\mathrm{AF}, \mathcal{A})$ if and only if $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}(E \cap S) \in \mathcal{GF}(\mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$. Then the conclusion easily follows by taking into account that $\mathcal{GF}$ actually depends (through $\mathcal{BF}_{\mathcal{S}}$) on the specific semantics $\mathcal{S}$.

Theorem 1. *In any argumentation framework which is SCC-symmetric and free of self-defeating arguments all of the *-symmetric semantics are in agreement, i.e. they prescribe the same set of extensions.*

Proof. It is sufficient to show that, given an argumentation framework AF satisfying the hypothesis and two *-simmetric semantics $\mathcal{S}_1$ and $\mathcal{S}_2$, $\forall E \in \mathcal{E}_{\mathcal{S}_1}(\mathrm{AF})\ E \in \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF})$ (the reverse condition can then be obtained by the same reasoning). According to Lemma 4, given $E \in \mathcal{E}_{\mathcal{S}_1}(\mathrm{AF})$, we have to prove that $\forall S \in \mathrm{SCCS}_{\mathrm{AF}}(E \cap S) \in \mathcal{GF}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$. We reason by induction along the strongly connected components of the argumentation framework. In particular, at any step we consider a specific $S \in \mathrm{SCCS}_{\mathrm{AF}}$ and we prove the following conditions:

1. $UP_{\mathrm{AF}}(S,E) = U_{\mathrm{AF}}(S,E)$ (i.e., $P_{\mathrm{AF}}(S,E) = \emptyset$)
2. $(E \cap S) \in \mathcal{GF}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E)) = \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)})$
3. $\mathcal{E}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}) = \mathcal{E}_{\mathcal{ST}}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)})$

assuming that these conditions hold for any $S' \in \mathrm{sccanc}_{\mathrm{AF}}(S)$ (notice that the case $\mathrm{sccanc}_{\mathrm{AF}}(S) = \emptyset$, i.e. S is an initial strongly connected component, is covered in the following proof). Then the conclusion is immediate from the first and second conditions.
As for the first condition (which is obvious when S is initial), we have to prove that $\forall \alpha \in UP_{\mathrm{AF}}(S,E)$ if $\beta \rightarrow \alpha$ and $\beta \notin S$ then $E \rightarrow \beta$. Notice that $\beta \in S'$ with $S' \in \mathrm{sccanc}_{\mathrm{AF}}(S)$, and $\beta \notin E$ since $\alpha \notin D_{\mathrm{AF}}(S,E)$. Since by the first condition applied to S' $P_{\mathrm{AF}}(S',E) = \emptyset$, either $\beta \in D_{\mathrm{AF}}(S',E)$ or $\beta \in U_{\mathrm{AF}}(S',E)$. In the first case, $E \rightarrow \beta$ by definition. In the second case, since $(E \cap S') \in \mathcal{E}_{\mathcal{ST}}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S',E)})$ by the second and third conditions and $\beta \notin (E \cap S')$, it holds that $(E \cap S') \rightarrow \beta$, thus again $E \rightarrow \beta$.
Let us turn to the second condition. Since $E \in \mathcal{E}_{\mathcal{S}_1}(\mathrm{AF})$, according to Lemma 4 $(E \cap S) \in \mathcal{GF}_{\mathcal{S}_1}(\mathrm{AF}{\downarrow}_{UP_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$, which by the above proof is equal to $\mathcal{GF}_{\mathcal{S}_1}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$, the latter being equal to $\mathcal{E}_{\mathcal{S}_1}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)})$ by Lemma 4. Now, since AF is SCC-symmetric $\mathrm{AF}{\downarrow}_S$ is symmetric by definition, entailing by Lemma 2 that $\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}$ is symmetric in turn. Notice that this argumentation framework, as AF, is free of self-defeating arguments. Then, since both $\mathcal{S}_1$ and $\mathcal{S}_2$ are *-symmetric $\mathcal{E}_{\mathcal{S}_1}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}) = \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}) = \mathcal{MCF}_{\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}}$. In sum, $(E \cap S) \in \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)})$, which by Lemma 4 is equal to $\mathcal{GF}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow}_{U_{\mathrm{AF}}(S,E)}, U_{\mathrm{AF}}(S,E))$.

Finally, the third condition follows from Proposition 7, which states in particular that stable semantics is *-symmetric, entailing that $\mathcal{E}_{\mathcal{ST}}(\mathrm{AF}{\downarrow_{U_{\mathrm{AF}}(S,E)}}) = \mathcal{MCF}_{\mathrm{AF}{\downarrow_{U_{\mathrm{AF}}(S,E)}}} = \mathcal{E}_{\mathcal{S}_2}(\mathrm{AF}{\downarrow_{U_{\mathrm{AF}}(S,E)}})$.

The following result immediately follows from the previous theorem and Proposition 7.

Corollary 1. *For any argumentation framework* AF *which is SCC-symmetric and free of self-defeating arguments,* $\mathcal{E}_{\mathcal{PR}}(\mathrm{AF}) = \mathcal{E}_{CF2}(\mathrm{AF}) = \mathcal{E}_{\mathcal{ST}}(\mathrm{AF})$*, thus in particular* AF *is coherent.*

Theorem 1 and Corollary 1 generalize the results about agreement provided in [4], where only symmetric argumentation frameworks are considered (which, as already remarked, feature a limited expressivity since they prevent, for instance, that an initial argument attacks any other argument). Moreover, agreement is proved for a family of multiple-status SCC-recursive semantics, including the most significant literature proposals we are aware of.

In [1] it was shown that a sufficient condition for agreement between preferred and stable semantics is that the considered argumentation framework is *limited controversial*. A finite argumentation framework is limited controversial if it does not include any odd-length cycle. The classes of SCC-symmetric and limited controversial argumentation frameworks are non-disjoint but distinct. In fact, a SCC-symmetric argumentation framework may contain cycles of any length, while a limited controversial argumentation framework may consist, for instance, of an even-length cycle which is not symmetric.

It is interesting to note that the property of SCC-symmetry may be recovered from assumptions on the attack relation which have been previously considered in the literature and are not directly related to decomposition into SCCs. For instance in [10] the case is considered where conflicts among arguments arise only from contradicting conclusions, namely only the *rebutting* kind of defeat is allowed while *undercutting* defeat is not (we follow here the terminology of [9], note that the notion of rebutting defeat we adopt includes attack against subarguments, that some authors call instead undercut). It is shown in Proposition 26 of [10] that if only rebutting defeat is allowed, the defeat graph is SCC-symmetric (such a graph is called *r-type* in [10]). From another perspective, in [11] it is shown that when the attack relation results from a symmetric conflict relation and a transitive preference relation between arguments the defeat graph satisfies a property called *strict acyclicity*, which is actually equivalent to SCC-symmetry through the characterization given in Proposition 8.

7 Conclusions

In this paper we have analyzed the issue of characterizing argumentation frameworks where semantics agree, exploiting to this purpose the recently introduced notion of SCC-recursiveness and the relevant existing results. Focusing on the two traditional questions of agreement with grounded and stable semantics, some

novel results have been obtained. As to the first question, the family of determined argumentation frameworks where any "reasonable" SCC-recursive semantics agrees with grounded semantics has been identified. Adding the requirement that the semantics is universally defined, a larger family of argumentation frameworks where such an agreement is ensured has been characterized. As to the second question, it has been shown that agreement is ensured, for a class of semantics including stable, preferred and $CF2$ semantics, on the significant family of SCC-symmetric argumentation frameworks. Among future work directions, we mention in particular the definition and study of forms of agreement at the level of justification states of arguments rather than of extensions.

Acknowledgments. The authors are indebted to the anonymous referees for their helpful comments.

References

1. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. Artificial Intelligence **77**(2) (1995) 321–357
2. Baroni, P., Giacomin, M.: Evaluation and comparison criteria for extension-based argumentation semantics. In Dunne, P.E., Bench-Capon, T., eds.: Proc. of the 1st International Conference on Computational Models of Arguments (COMMA 2006), Liverpool, UK, IOS Press (2006) 157–168
3. Caminada, M.: On the issue of reinstatement in argumentation. In: Proc. of the 10th European Conference on Logics in Artificial Intelligence (JELIA 06), Liverpool, UK, Springer (2006) 111–123
4. Coste-Marquis, S., Devred, C., Marquis, P.: Symmetric argumentation frameworks. In: Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005), Barcelona, E (2005) 317–328
5. Baroni, P., Giacomin, M., Guida, G.: Scc-recursiveness: a general schema for argumentation semantics. Artificial Intelligence **168**(1-2) (2005) 165–210
6. Prakken, H., Vreeswijk, G.A.W.: Logics for defeasible argumentation. In Gabbay, D.M., Guenthner, F., eds.: Handbook of Philosophical Logic, Second Edition. Kluwer Academic Publishers, Dordrecht (2001)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press and McGraw-Hill (2001)
8. Baroni, P., Giacomin, M.: Solving semantic problems with odd-length cycles in argumentation. In: Proceedings of the 7th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2003), Aalborg, Denmark, LNAI 2711, Springer-Verlag (2003) 440–451
9. Pollock, J.L.: How to reason defeasibly. Artificial Intelligence **57**(1) (1992) 1–42
10. Baroni, P., Giacomin, M., Guida, G.: Self-stabilizing defeat status computation: dealing with conflict management in multi-agent systems. Artificial Intelligence **165**(2) (2005) 187–259
11. Kaci, S., van der Torre, L.W.N., Weydert, E.: Acyclic argumentation: Attack = conflict + preference. In: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006), Riva del Garda, Italy, IOS Press (2006) 725–726

A Sound and Complete Dialectical Proof Procedure for Sceptical Preferred Argumentation

Phan Minh Dung and Phan Minh Thang

Department of Computer Science, Asian Institute of Technology
GPO Box 4, Klong Luang, Pathumthani 12120, Thailand
dung@cs.ait.ac.th, thangphm@ait.ac.th

Abstract. We present a dialectical proof procedure for computing skeptical preferred semantics in argumentation frameworks. The proof procedure is based on the dispute derivation introduced for assumption-based framework. We prove the soundness of the procedure for any argumentation frameworks and the completeness for a general class of finitary argumentation frameworks containing the class of finite argumentation frameworks as a subclass.

1 Introduction

Argumentation is a form of reasoning, that could be viewed as a debate, in which the participants present their arguments to establish, defend, or attack certain propositions. An argument could be said to represent a consensus if it is accepted by all participants. For example, in legal domain, different members of a jury could have different views of the presented evidence (different preferred extensions) but a guilty verdict is the result of a consensus among members. This form of reasoning to find a consensus is characterized by the skeptical semantics in argumentation. Skeptical semantics is also useful in AI systems for negotiation and decision making [14–17, 21].

Several procedures for computation of skeptical preferred semantics have been proposed, e.g the TPI procedure [19] for coherent argumentation framework [4, 10], and a dialectic procedure for finding ideal skeptical semantics, an approximation of the skeptical preferred semantics [9]. In [4], an algorithm for computing sceptical preferred semantics is proposed. Given an argument a, the algorithm proceeds in two separate steps: It first checks that a is not attacked by any admissible set. In the second step, it looks for an admissible set that can not be extended into a bigger one containing a. Failure to find such a set implies that a is included in each preferred extension. In other words, the algorithm represents an indirect way of proving that a is skeptically preferred based on the idea that failure to show that a is not skeptical preferred implies the contrary. Though the idea is intuitively clear, no formal proof for the soundness of the algorithm is given.

In contrast, in this paper we present a direct dialectical proof procedure for general skeptical preferred semantics. We prove the soundness of the procedure for any argumentation frameworks and its completeness for a general class of finitary argumentation frameworks containing the class of finite argumentation frameworks as a subclass.

The structure of the paper is as follows. In section 2 we recall and introduce notions of proof tree, proof derivation and proof procedure for credulous preferred semantics. We introduce finitary argumentation frameworks and prove soundness and completeness of credulous proof procedure for them. In section 3 we present proof theories and algorithm for general skeptical preferred semantics.

2 Credulous Acceptance

Following [7], we define an argumentation framework as a pair $AF = (\mathcal{A}, att)$, where A is a set of arguments, and att is a binary relation on $\mathcal{A}$ ($att \subseteq \mathcal{A} \times \mathcal{A}$). Given two arguments A and B, (A,B)$\in$ att means A attacks B. A set S of arguments attacks an argument A if there is an argument B in S such that B attacks A. The definitions of conflict-free set, admissible set and preferred extension are recalled from [7] as follows:

Let S be a set of arguments

1. S is conflict-free iff there exist no arguments A, B in S such that A attacks B
2. Argument A is acceptable with respect to S iff for each argument B if B attacks A then S attacks B
3. S is admissible iff S is conflict-free and each argument in S is acceptable with respect to S
4. S is a preferred extension of AF iff S is a maximal admissible set of AF
5. Argument A is credulously accepted iff A is contained in at least one preferred extension of AF
6. Argument A is skeptically accepted iff A is contained in every preferred extensions of AF

To prove the credulous acceptance of an argument, a proof tree is constructed. A proof tree can be viewed as a specification of a debate between a proponent and an opponent, where an initial argument is put forward by the proponent, and then the opponent and proponent alternatively present their arguments to attack the arguments of the other. The proponent wins the dispute if he can attack every attacking argument of the opponent. We recall the definition of proof tree from [8, 9]:

Definition 1. *A proof tree for an argument A with respect to an argument framework AF is defined as following:*

1. *Nodes are labeled by arguments and the root is labeled by A. The argument labeling a child node attacks the argument labeling its parent.*

2. *There are two types of nodes: proponent nodes and opponent nodes*
3. *Each opponent node has exactly one child that is a proponent node*
4. *For each proponent node N labeled by an argument B, N has as many children nodes as the number of arguments attacking B, and for every argument C attacking B, there is a child node of N, which is an opponent node labeled by C.*

Definition 2. *A proof tree is said to be admissible if there is no argument that labels both a proponent node and an opponent node.*

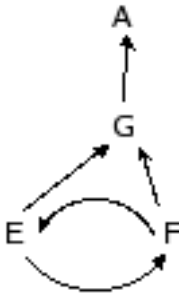

Fig. 1.

Fig. 2.

Example 1. The argumentation framework AF=($\mathcal{A}$,att) is depicted on Fig. 1, where: $\mathcal{A}$={A,G,E,F} and att = {(G, A),(E, G),(F, G),(E,F),(F, E)}.

A proof tree for argument A is depicted in Fig. 2 where:

- argument A labels the root
- arguments A, E label proponent nodes
- arguments G, F label opponent nodes
- this tree is admissible, since there is no argument labeling both opponent and proponent nodes.

The following lemma is similar to a theorem from [8] and proved in the extended version of [9].

Lemma 1.

1. *Let T be an admissible proof tree for A and S the set of all arguments labeling proponent nodes of T. Then S is admissible.*
2. *Let S be an admissible set of arguments and A in S. Then there exists an admissible proof tree T for A such that the set of arguments labeling the proponent nodes in T is a subset of S*

A proof tree is often infinite as example 1 shows. A proof derivation is a finite top-down construction of an (possibly infinite) admissible proof tree by a sequence of tuples $< P, O, SP, SO >$, where P is a set of arguments, put forward by the proponent but which have not been attacked yet, and O is a set of arguments put forward by the opponent to attack the proponent's arguments, against which the proponent doesn't have counter-attack until now. SP is the set of arguments presented by proponent, and SO is the set of arguments put forward by the opponent, and already counter-attacked by the proponent. Our proof derivation is defined in a spirit like the dispute derivation in [8] and the dialectical games in [4]. In each step of a proof derivation building process only one argument is selected. Let B be a selected argument and let O_B be the set of all arguments attacking B. In the first case if B labeling a proponent node, then O_B consists of all arguments labeling opponent child nodes of the node labeled by B. In the second case if B labeling an opponent node, then O_B is a set of arguments, from which one argument is chose to label a proponent child node of the node labeled by B. Hence there exists no proof derivation if there is one argument in O_B labeling a proponent node in the first case, or there is one argument in O_B labeling an opponent node or $O_B = \emptyset$ in the second case, because our proof tree is not admisible.

Definition 3. *A proof derivation $\mathcal{D}$ for an argument A is a sequence $< P_0, O_0, SP_0, SO_0 > ... < P_n, O_n, SP_n, SO_n >$ where:*

1. *P_i, O_i, SP_i, and SO_i are argument sets*
2. *$P_0 = SP_0 = \{A\}$, $SO_0 = O_0 = \emptyset$, $P_n = O_n = \emptyset$*
3. *Let B be the argument selected at step i, and let O_B be the set consisting of all arguments attacking B.*
   (a) *If $B \in P_i$ and $O_B \cap SP_i = \emptyset$ then*
   $P_{i+1} = P_i \setminus \{B\}$
   $O_{i+1} = O_i \cup (O_B \setminus SO_i)$
   $SP_{i+1} = SP_i$
   $SO_{i+1} = SO_i$
   (b) *If $B \in O_i$ then select an argument $C \in O_B$ such that $C \notin (SO_i \cup O_i)$*
   $P_{i+1} = Pi \cup \{C\}$ *if* $C \notin SP_i$, *otherwise* $P_{i+1} = P_i$
   $O_{i+1} = O_i \setminus \beta$ *where* $\beta = \{B' \mid C$ *attacks* $B'\}$
   $SP_{i+1} = SP_i \cup \{C\}$
   $SO_{i+1} = SO_i \cup (\beta \cap O_i)$ *(Note that* $B \in \beta \cap O_i$*)*

Example 2. Let argumentation framework AF=($\mathcal{A}$,att), where $\mathcal{A} = \{A\}$ and att=$\emptyset$ then a sequence $< \{A\}, \emptyset, \{A\}, \emptyset >< \emptyset, \emptyset, \{A\}, \emptyset >$ is the proof derivation for A.

Example 3. (*Continue example 1*) A proof derivation for A is presented in following table, where the notation $\underline{\text{X}}$ means that X is selected in step 3 of definition 3.

i	$\mathbf{P_i}$	$\mathbf{O_i}$	$\mathbf{SP_i}$	$\mathbf{SO_i}$	comment
0	$\underline{\text{A}}$	$\emptyset$	A	$\emptyset$	O_A={G} according to step3.a
1	$\emptyset$	$\underline{\text{G}}$	A	$\emptyset$	O_G={E,F}, E is selected form O_G, β={G,F} according to step3.b
2	$\underline{\text{E}}$	$\emptyset$	A, E	G	O_E={F} according to step3.a
3	$\emptyset$	$\underline{\text{F}}$	A, E	G	O_F={E}, E is selected and E$\in SP_3$, β={G,F} according to step3.b
4	$\emptyset$	$\emptyset$	A, E	G, F	

Table 1. The construction of a proof derivation for A

Theorem 1.

1. *Suppose* $< P_0, O_0, SP_0, SO_0 > ... < P_n, O_n, SP_n, SO_n >$ *is a proof derivation for A. Then* SP_n *is admissible and* $A \in SP_n$.

2. *Let AF be a finite argumentation framework, and let A be an argument of AF. If A belongs to an admissible set then there is a proof derivation for A.*

Consider the infinite argumentation framework in Fig. 3. It is not difficult to see that there is an unique preferred extension consisting of arguments $A_0, A_2, ..., A_{2n}$, ... It is obvious that for each argument A_{2n} there is a proof derivation for A_{2n}. The reason for the existence of a proof derivation for A_{2n} is that the argumentation framework consisting of the arguments from which there is a directed path to A_{2n} is finite. In the following, we introduce the class of finitary argumentation frameworks generalizing this property.

Fig. 3.

Let AF=($\mathcal{A}$, *att*) and $A \in \mathcal{A}$.[1] The environment of A denoted by ENV_A is the set of all arguments B in $\mathcal{A}$ such that there is a directed path from B to A in the graph of AF (i.e.there is a sequence $B_1, B_2..., B_n$ such that B_i attacks B_{i+1} and B=B_1 and A=B_n). Let $AF_A = (ENV_A, att_A)$, where att_A is the restriction of *att* to ENV_A.

[1] For purpose of reference, we often identify AF with the graph representing it.

Definition 4. *An argumentation framework is said to be finitary if for each argument A, AF_A is finite.*

Lemma 2.

1. *Let S be an admissible set of arguments in AF. Then $S \cap ENV_A$ is also admissible in both AF and AF_A.*
2. *Let $S \subseteq ENV_A$ be an admissible set in AF_A. Then S is also admissible in AF.*

From the lemma 2, it is obvious that

Corollary 1. *A is credulously accepted in AF iff A is credulously accepted in AF_A.*

The soundness and completeness of proof derivation for finitary argumentation frameworks follows immediately from the above corollary and theorem 1.

Theorem 2. *Let AF be a finitary argumentation framework, and A be an argument of AF. A belongs to an admissible set iff there is a proof derivation for A.*

3 Skeptical Acceptance

An argumentation framework AF is said to be coherent if each preferred extension of AF is stable. In other words, coherence implies the coincidence between stable and preferred semantics. TPI procedures are based on the following proposition [4, 11, 19] to check whether a given argument A is skeptically accepted in coherent argumentation frameworks: An argument A is skeptically accepted in a coherent argumentation frameworks if A is credulously accepted and there exists no admissible set attacking A.

The following example shows that TPI procedures can not be used for answering whether a given argument belongs to all preferred extensions in general cases.

Example 4. The argumentation framework AF=($\mathcal{A}$,*att*) is depicted in Fig. 4, where $\mathcal{A}$ = {A,B,G,E,F} and *att* = {(G,A),(A,B),(B,G),(E,G),(F,E),(E,F)}

It is clear that {A,E} and {F} are the only preferred extensions, and argument A is not skeptically accepted, although A is credulously accepted and there exist no admissible set attacking A.

In this chapter we introduce a proof procedure for skeptical preferred semantics in general cases, which is based on the following simple lemma.

Lemma 3. *Let S be an admissible set of arguments and E be a preferred set of arguments, and S is not a subset of E. Then E attacks S (and S also attacks E).*

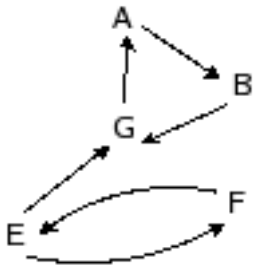

Fig. 4.

Definition 5. *Let A be an argument, and let $\mathcal{B}$ be a set of admissible sets such that each element of $\mathcal{B}$ contains A.*

1. *If for each preferred extension E such that A$\in$ E, there exists an admissible set $S \in \mathcal{B}$ such that $S \subseteq E$ then $\mathcal{B}$ is called a base of A.*
2. *A base $\mathcal{B}$ of A is said to be complete if for each preferred extension E, there is a set S$\in$ B such that $S \subseteq E$*

Lemma 4. *(Skeptical Lemma) An argument A is sceptically accepted iff there exist a complete base $\mathcal{B}$ of A.*

The skeptical lemma suggests that a proof procedure for showing that A is skeptically accepted, could proceed in two steps:

1. Generate a base $\mathcal{B}$ of A
2. Verify that $\mathcal{B}$ is a complete base of A

3.1 Generating a Base of A

We define a $\mathcal{BG}$[2]-derivation for an argument A by constructing all possible proof derivations for A.

Definition 6. *$\mathcal{BG}$-derivation for A is a sequence $T_0, T_1, ..., T_n$, where:*

1. *T_i is a set of tuples of the form $< P, O, SP, SO >$*
2. *$T_0 = \{< \{A\}, \emptyset, \{A\}, \emptyset >\}$*
3. *Each tuple t of T_n has the form $< \emptyset, \emptyset, SP, SO >$*
4. *At each step T_i one tuple $t_i =< P_i, O_i, SP_i, SO_i >$ is selected from T_i and one argument B is selected from P_i or O_i.*
   (a) *If B is selected from P_i, then: $T_{i+1} = (T_i \setminus \{t_i\}) \cup \{t'\}$, where t' is computed from t_i as in definition 3 step 3.a.*

[2] $\mathcal{BG}$ stands for "Base Generation"

(b) If B is selected from O_i and let O_B be the set consisting of all arguments attacking B, then: $T_{i+1} = (T_i \setminus \{t_i\}) \cup \{t' \mid t'$ is computed from t_i as in definition 3 step 3.b for some argument $C \in O_B$ such that $C \notin (SO_i \cup O_i)\}$

It is not difficult to see the following:

Theorem 3.

1. *Let $T_0, T_1, ..., T_n$ be a BG-derivation for A. Let $\mathcal{B} = \{SP \mid < \emptyset, \emptyset, SP, SO > \in T_n\}$. Then $\mathcal{B}$ is a base of A.*
2. *Let AF be finitary. Then there exists a BG-derivation for A.*

3.2 Verifying the Completion of a Base

Before giving the procedure for verifying the completeness of a base, we need a few technical results.

Lemma 5. *Let $\mathcal{B}$ be a base of argument A. $\mathcal{B}$ is a complete base of A iff there exist no preferred extension E attacking every element of $\mathcal{B}$.*

A proof derivation for a given argument A is constructed to find an admissible set of arguments defending A. However in some cases we want to answer the question "can the proponent admissibly attack arguments proposed by the opponent". A notion of a proof derivation $\mathcal{D}$ against S is introduced for this purpose.

Definition 7. *A proof derivation $\mathcal{D}$ against a set S of arguments is defined as a sequence $< P_0, O_0, SP_0, SO_0 > ... < P_n, O_n, SP_n, SO_n >$ where:*

1. *$P_i, O_i, SP_i,$ and SO_i are argument sets*
2. *$P_0 = SP_0 = \emptyset, O_0 = S, SO_0 = \emptyset, P_n = O_n = \emptyset$*
3. *$< P_{i+1}, O_{i+1}, SP_{i+1}, SO_{i+1} >$ is constructed from $< P_i, O_i, SP_i, SO_i >$ as in definition 3 step 3.*

Lemma 6. *For finitary argumentation frameworks, there exists a proof derivation $\mathcal{D}$ against a set S iff there exist an admissible set S' attacking every element in S.*

Let A be an argument and $\mathcal{B} = \{S_1, ..., S_n\}$ where S_i is an admissible set containing A, and let $\mathcal{CB} = \{S \mid \exists e \in S_1 \times S_2 \times ... \times S_n$ and S is the set of arguments appearing in e$\}$, and let $\mathcal{XB} = \{S \mid S \in \mathcal{CB}$ and S is minimal in $\mathcal{CB}$ wrt set inclusion$\}$

Lemma 7. *For finitary argumentation frameworks, let $\mathcal{B}$ be a base of A. $\mathcal{B}$ is a complete base of A iff for each S$\in \mathcal{XB}$ there exist no proof derivation $\mathcal{D}$ against S.*

Based on lemma 7 we define now a $\mathcal{CB}$[3]-verification for a base $\mathcal{B}$ of an argument A to verify the completeness condition of $\mathcal{B}$.

Definition 8. *Let $\mathcal{B}$ be a finite set. A $\mathcal{CB}$-verification for $\mathcal{B}$ is a sequence $J_0, J_1 ... J_n$ where*

1. *J_i is a set of tuples of the form $< P, O, SP, SO >$*
2. *$J_0 = \{< \emptyset, O, \emptyset, \emptyset > | \ O \in \mathcal{XB}\}$*
3. *$J_n = \emptyset$*
4. *J_{k+1} is obtained from J_k like T_{k+1} is obtained from T_k in definition 6.*

Theorem 4. *Let AF be a finitary framework and $\mathcal{B}$ be a finite base of argument A. There exists a $\mathcal{CB}$-verification for $\mathcal{B}$ iff $\mathcal{B}$ is a complete base of A.*

3.3 Proof Procedure for Skeptical Acceptance

We define a $\mathcal{SA}$[4]-derivation for A as a combination of a $\mathcal{BG}$-derivation for A and a $\mathcal{CB}$-verification for the base created by the $\mathcal{BG}$-derivation.

Definition 9. *Let A be an argument. An $\mathcal{SA}$-derivation for A is a sequence $T_0, T_1, ..., T_n, J_0, J_1 ... J_m$ where:*

1. *The sequence $T_0, T_1, ..., T_n$ is a $\mathcal{BG}$-derivation for A*
2. *The sequence $J_0, J_1 ... J_m$ is a $\mathcal{CB}$-verfication for $\mathcal{B}$, where $\mathcal{B} = \{SP \mid < \emptyset, \emptyset, SP, SO > \in T_n\}$*

The following theorem follows directly from theorems 3, 4

Theorem 5. *Let AF be a finitary argumentation framework and A be an argument in AF. A is sceptically accepted iff there exists a SA-derivation for A.*

Example 5. (*Continue example 1*) Our proof procedure shows that A is skeptically accepted (see table 2). The notion fails means 'fails to build a proof derivation'.

Example 6. (*Continue example 4*) Our proof procedure shows that A is not skeptically acceptedm(see table 3), something that can not be done using TPI-procedures.

From table 3 we see that there exist no $\mathcal{SA}$-derivation for A. Hence A is not sceptically accepted.

[3] $\mathcal{CB}$ stands for Complete Base
[4] $\mathcal{SA}$ stands for Skeptical Acceptance

$\mathcal{BG}$-derivation for A					
	P	**O**	**SP**	**SO**	**comment**
T_0	A	∅	A	∅	step4.a, O_A={G}
T_1	∅	G	A	∅	step4.b, O_G={E,F}
T_2	E	∅	A, E	G	step4.a, O_E={F}
	F	∅	F, A	G	
T_3	∅	F	A, E	G	step4.b, O_F∩ SP={E}
	F	∅	F, A	G	
T_4	∅	∅	A, E	G	
	F	∅	A, F	G	step4.a O_F={E}
T_5	∅	∅	E,A	G	
	∅	E	F,A	G	step4.b O_E∩SP={F}
T_6	∅	∅	E,A	G	$\mathcal{B}$={{E,A}, {F,A}} and
	∅	∅	F,A	G	$\mathcal{XB}$ ={{A},{E,F}}

$\mathcal{CB}$-verification for $\mathcal{B}$					
	P	**O**	**SP**	**SO**	**comment**
J_0	∅	A	∅	∅	step4.b, O_A={G}
	∅	E, F	∅	∅	
J_1	G	∅	G	A	step4.a, O_G={E,F}
	∅	E,F	∅	∅	
J_2	∅	E,F	G	A	step4.b O_E∩O={F}, **fails**
	∅	E,F	∅	∅	
J_3	∅	E,F	∅	∅	step4.b O_E∩O={E}, **fails**
J_4	**Empty**				

Table 2. Construction of a BG for A and CB-verification for $\mathcal{B}$

	P	**O**	**SP**	**SO**	**comment**
T_0	A	∅	A	∅	step3.a, O_A={G}
T_1	∅	G	A	∅	step3.b, O_G={E,B}
T_2	E	∅	A, E	G	step3.a, O_E={F}
	B	∅	A, B	G	
T_3	∅	F	A, E	G	step3.b, O_F={E}, {E}*in* SP
	B	∅	A, B	G	
T_4	∅	∅	A, E	G, F	
	B	∅	A, B	G	step3.a O_B∩ SP={A} **fails**
T_5	∅	∅	A, E	G, F	$\mathcal{B}$={{A, E}} and $\mathcal{XB}$={{A}, {E}}

	P	**O**	**SP**	**SO**	**comment**
J_0	∅	E	∅	∅	step3.b, O_E={F}
	∅	A	∅	∅	
J_1	F	∅	F	E	step3.a, O_F=SO={E}
	∅	A	∅	∅	
J_2	∅	∅	F	E	
	∅	A	∅	∅	step3.b O_A={G}
J_3	∅	∅	F	E	
	G	∅	G	A	step3.a O_G={E, B}
J_4	∅	∅	F	E	
	∅	B, E	G	A	step3.b O_B∩SO={A} **fails**
J_5	∅	∅	F	E	**not empty**

Table 3. Construction of the SA-derivation for A

4 Conclusion and Discussions

It is a well-known result from [11] that skeptical acceptance is $\prod_2^{(p)}$-complete. Therefore in worst cases, computing a $\mathcal{SA}$ derivation is not polynomial.

Consider the argumentation framework in Fig. 5. Using the $\mathcal{BG}$-derivation, we would be able to generate a base $\mathcal{B}$ = {{A,E,C},{A,E,D},{A,F,C},{A,F,D}}. Looking at the subgraph consisting of only E,F, we could realize that if there is any attack against E or F, it should come from within this subgraph. Similarly for C,D. Hence, it would be enough if in the $\mathcal{CB}$-verification, we consider only derivations againsts {A},{E,F},{C,D}. Structuring argumentation frameworks into strongly connected component like in [1] would facilitate optimizing the $\mathcal{SA}$-derivations in this direction.

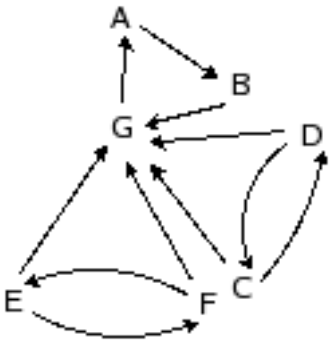

Fig. 5.

5 Acknowledgements

We would like to thank the referees for their constructive comments and criticisms. The authors are partially supported by the EU sponsored project, ArguGrid.

A Appendix

A.1 Proof of lemma 2

1. Let R=$S \cap ENV_A$. It is obvious that R is conflict-free. Let B be an argument attacking R. It is obvious that B $\in ENV_A$. Hence there is C $\in S$ such that C attacks B. Hence C $\in ENV_A$. Hence C $\in R$. Hence R is admissible both wrt AF and AF_A.
2. It is clear that S is conflict-free in AF. Let B be an argument attacks S in AF. Hence B $\in ENV_A$. Hence S attacks B in AF_A. Hence S attacks B in AF.

A.2 Proof of lemma 3

It is clear that if E attacks S then S also attacks E and vice versa. Assume that S, E do not attack each other. Hence C=S ∪ E is conflict-free. For each argument A in C if there is an argument B attacking A then B is attacked by S or by E since A is in S or E. So B is attacked by C. Hence each argument in C is acceptable wrt C. Then C is admissible and contains E and there exists an argument G in C which is not in E because S is not subset of E. Contradiction since E is preferred. Hence S attacks E and E also attacks S.

A.3 Proof of lemma 4

1. Only if part
   Let $\mathcal{B}$ be the set of all preferred extensions, then $\mathcal{B}$ is a complete base of A.
2. If part
   Let $\mathcal{B}$ be a complete base of A, then for each preferred extension E there exists a set $S \in \mathcal{B}$ such that $S \subseteq E$. Since A $\in S$ for each $S \in \mathcal{B}$ then A is contained in each preferred extension E. Hence A is sceptically accepted.

A.4 Proof of lemma 5

1. Only if part
   Let $\mathcal{B}$ be a complete base of A, and E be an arbitrary preferred extension. Then there is an admissible set S of $\mathcal{B}$ such that $S \subseteq E$. Hence E does not attack S. Hence E does not attack every element of $\mathcal{B}$.
2. If part
   Assume $\mathcal{B}$ is not a complete base of A. Then there exists a preferred extension E such that $E \not\supseteq S$ for every element S of $\mathcal{B}$. Hence E attacks every element S of $\mathcal{B}$ (lemma 3). Contradiction.

A.5 Proof of lemma 6

Let AF=($\mathcal{A}$,att) be the argumentation framework we are working in. Let AF'=($\mathcal{A}'$,att') be another argumentation framework such that $\mathcal{A}'$= $\mathcal{A}$∪ {T}, where T is a new argument not in $\mathcal{A}$, att'=att∪ {(C, T) | C ∈ S}. Then each proof derivation $\mathcal{D}$ against S can be transferred into a proof derivation $\mathcal{D}'$ for T by adding the tuple $< \{T\}, \emptyset, \{T\}, \emptyset >$ to the beginning of $\mathcal{D}$ and add T to the SP component in each tuple in $\mathcal{D}$

1. Only if part
   Since there is a proof derivation $\mathcal{D}$ against a set S in AF, then there is a proof derivation $\mathcal{D}'$ for T in AF'. Hence there exists an admissible set R in AF' containing T. Since T is attacked by every element of S, then each element of S is attacked by R. Let S'=R \ {T}. Hence each element of S is attacked by S'. S' is conflict-free, because R is an admissible set. Since $S' \subseteq \mathcal{A}$, every argument attacking S' belongs to $\mathcal{A}$. For each argument B

attacking *S'*, there is an argument B' $\in$ *S'* such that (B',B)$\in$ *att*, because *att'*=*att* $\cup$ {(C, T) | C $\in$ *S*). Hence *S'* is an admissible set wrt *AF*. So there exists an admissible set attacking every element in *S*.

2. If part
   Let *R*=*S'* $\cup$ {T}. T is not in $\mathcal{A}$, then T is not in *S'*. Furthermore *S'* is admissible, and set *S'* defends T, then *R* is admissible. Hence there is a proof derivation $\mathcal{D}'$ for T. Hence there is a proof derivation $\mathcal{D}$ against a set *S* by dropping the first tuple from $\mathcal{D}'$.

A.6 Proof of lemma 7

1. Only if part
   Let *E* be a preferred extension. Since $\mathcal{B}$ is a complete base for A, then there is a set $S_i \in \mathcal{B}$ such that $S_i \subseteq E$. That means for each $S \in \mathcal{XB}$ there is an argument C $\in (S \cap S_i)$ such that *E* doesn't attack C. Hence for each $S \in \mathcal{XB}$ there exist no preferred extension attacking every element in *S*. Then for each $S \in \mathcal{XB}$ there exists no admissible set attacking every element in *S*. Hence there exists no proof derivation $\mathcal{D}$ against *S* (lemma 6).
2. If part
   Assume the contradiction, that means $\mathcal{B}$ is not complete base of A. Hence there exists a preferred extension *E* attacking every element S_i of $\mathcal{B}$ (lemma 5). Hence for each S_i there is an argument C_i in S_i such that *E* attacks C_i. Hence *E* attacks every element in *S*=$\{C_1, C_2 ... C_n\} \in \mathcal{XB}$. Hence there exists proof derivation $\mathcal{D}$ against *S* (lemma 6). Contradiction.

A.7 Proof of theorem 4

Let $\mathcal{XB} = \{O_1, ..., O_n\}$. Let *AF*=($\mathcal{A}$,*att*). Let *AF'*=($\mathcal{A}'$,*att'*) and $\mathcal{A}' = \mathcal{A} \cup R$ where *R*={A',Q,$G_1, ..., G_n$} and $\mathcal{A} \cap R = \emptyset$, *att'*=*att* $\cup \{(C_1, G_1) \mid C_1 \in O_1\} \cup$ ... $\cup \{(C_n, G_n) \mid C_n \in O_n\} \cup \{(G_1, Q), ..., (G_n, Q), (Q, A')\}$ (figure 6). A $\mathcal{CB}$-verification $\mathcal{D}$ for $\mathcal{B}$ can be transferred to a proof derivation $\mathcal{D}'$ for A' by

adding a sequence T_0, T_1, T_2 to the beginning of $\mathcal{D}$, and

$$T_0 = < \{A'\}, \emptyset, \{A'\}, \emptyset >$$

$$T_1 = < \emptyset, \{Q\}, \{A'\}, \emptyset >$$

$$T_2 = \{< \{G_i\}, \emptyset, \{G_i, A'\}, \{Q\} > \mid i \in [1, n]\}$$

$$T_3 = \{< \emptyset, O_i, \{G_i, A'\}, \{Q\} > \mid i \in [1, n]\}.$$

It is not difficult to see that T_3 corresponds to J_0 of $\mathcal{D}$ in the sense that the first two components of the tuples in T_3 coincide with the first two components of the tuples in $\mathcal{D}$. $\mathcal{D}$ could be easily modified to have T_3 as its first element since G_i, A', Q do not have any effects on the status of the elements in $\mathcal{A}$. Abusing the notation, we still identify T_3 and J_0.

1. Only if part

   There is a $\mathcal{CB}$-verification for $\mathcal{B}$ wrt AF. Hence there exists no proof derivation for A' in AF'. Hence there exists no admissible set containing one of $G_1, ..., G_n$. That means $\forall$i there is no admissible set containing G_i. Therefore $\forall$i there is no admissible set attacks each argument in O_i. Hence there is no admissible set attacking each element in $\mathcal{B}$. Hence $\mathcal{B}$ is complete.
2. If part

   $\mathcal{B}$ is complete then there is no admissible set attacking every $S_i \in \mathcal{B}$ wrt AF. So there is no admissible set attacking every element of O_i for each $O_i \in \mathcal{XB}$. We prove that there is no proof derivation for A' wrt AF'. Assume contradiction, that means there is an admissible set S containing A'. Let S'=$S \setminus \{A'\}$. Since S is admissible and A' doesn't defend any argument, then S' is admissible. G_i defends A', then at least one $G_i \in S'$. Set O_i attacks G_i then every element of O_i is attacked by S' in AF'. Let S''=$S' \setminus \{G_i\}$. Since S' is admissible and G_i does not attack any argument in S', then S'' is admissible, and every element of O_i is attacked by S'' in AF'. Since $R \cap S''=\emptyset$ and S'' is not attacked by R and S'' is admissible, then every element of O_i is attacked by S'' in AF. Contradiction. Hence there exists no proof derivation for A'. Hence there is a $\mathcal{CB}$-verification for $\mathcal{B}$.

References

1. P. Baroni, M. Giacomin, and G. Guida SCC-recursiveness: a general schema for argumentation sematics. In *Artificial Intelligence, 168(1-2):162-210,2005.*
2. A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni An abstract, argumentation theoretic approach to default reasoning. In *Artificial Intelligence, 93:63-101, 1997.*
3. M. Caminada On the Issue of Reinstatement in Argumentation. *Technical report UU-CS-023-2006.*
4. C. Cayrol, S. Doutre, and M. Jerome On Decision Problems related to the Preferred Semantics for Argumentation Framework. In *Journal of logic and computation, 13(3):377403,2003.*
5. C.R. Chesnevar, A.G. Maguitman, and R.P. Loui Logical Models of Argument. In *ACM Computing Surveys,32(4):337383,2000.*
6. S. Doutre, and M. Jerome On sceptical vs credulous acceptance for abstract systems. In *Lecture notes in computer science, 3229:462-473,2004.*
7. P.M. Dung On the acceptability of arguments and its fundamental role in monotonic reasoning, logic programming and n-person games. In *Artificial Intelligence, 77:321-357, 1995.*
8. P.M. Dung, R.A. Kowalski, and F. Toni Dialectic proof procedures for assumption-base, admissible argumentation. In *ArtificialIntelligence, 170: 114-159, 2006.*
9. P.M. Dung, P. Mancarella, and F. Toni A dialectic procedure for sceptical, assumption-based argumentation. In *COMMA 2006.10S Press.*
10. Paul E. Dunne, and T.J.M. Bench-Capon Two party immediate response disputes: Properties and efficiency. In *Artificial Intelligence, 149:221-250, 2003.*

11. Paul E. Dunne, and TJ.M. Bench-Capon Coherence in finite argument systems. In *Artificial Intelligence, 141: 187203,2002.*
12. H. Iakobovits and D. Vermeir Dialectic Semantics for Argumentation Frameworks. In *Proc. ICAIL'99, pp 53-62. ACM Press, 1999.*
13. A.C. Kakas and F. Toni Computing Argumentation in Logic Programming. In *Journal of Logic and Computation, 9(4):515562, August 1999.*
14. S. Kraus, K. Sycara, and A. Evenchik Reaching agreements throughs argumentation: a logical model and implementation. In *Artificial Intelligence, 104: 1-69, 1998.*
15. S. Parson, C. Sierra, and N.R. Jennings Agent that reason and negotiate by arguing. In *Journal of Logic and Computation, 8(3):261-292, 1998.*
16. I. Rahwan, S.D. Ramchurn, N.R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-Base Negotiation. In *The Knowledge Engineering Review, 18(4):343-375,2004.*
17. I. Rahwan, L. Sonenberg, and F. Dignum. Towards Interest-Based Negotiation. In *MMAS'03, pp 773-780, ACM Press, 2003.*
18. B. Verheij A Labeling Approach to the Computation of Credulous Acceptance in Argumentation. In *IJCAI07, pp 623-628*
19. G.A.W. Vreeswijk, and H. Prakken. Credulous and Sceptical Argument Games for Preferred Semantics. In *Proc. JELIA'2000, pp 239-253. LNAI 1919,2000.*
20. G. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proceedings of the First International Conference on Computational Models of Argument, pp 109-120. IOS Press 2006.*
21. M. Wooldridge. An Introduction to Multiagent Systems. *John Wiley & Sons, Ltd, 2002.*

Argumentation-based Proof for an Argument in a Paraconsistent Setting ⋆

Iara Carnevale de Almeida[2] and José Júlio Alferes[1]

[1] CENTRIA, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
jja@di.fct.unl.pt
[2] CITI, Departamento de Informática, Universidade de Évora
Colégio Luis Verney; 7000-671 Évora, Portugal
ica@di.uevora.pt

Abstract. The paradigm of argumentation has been used in the literature to assign meaning to knowledge bases in general, and logic programs in particular. With this paradigm, rules of logic program are viewed as encoding arguments of an agent, and the meaning of the program is determined by those arguments that somehow (depending on the specific semantics) can defend themselves from the attacks of others arguments, named acceptable arguments. In previous work we presented an argumentation based declarative semantics allowing paraconsistent reasoning and also dealing with sets of logic programs that argue and cooperate among each other. In this paper we focus on the properties of this semantics in what regards paraconsistency and propose a procedure for proving an argument according to that semantics.

1 Introduction

In logic programming, several ways to formalise argumentation-based semantics have been studied for logic programs. Intuitively, argumentation-based semantics treat the evaluation of a logic program as an argumentation process, i.e. a goal G is true if at least one argument for G cannot be successfully attacked. The ability to view logic programming as a non-monotonic knowledge representation language, in equal standing with other non-monotonic logics, brought to light the importance of defining clear declarative semantics for logic programs, for which proof procedures (and attending implementations) are then defined (e.g. [8, 9, 15, 2, 20, 12, 18, 7, 14, 10]).

In [5] we proposed an argumentation based semantics for sets of logic programs that are able to cooperate and argue with each other. In it each program relies on a set of other programs with which it has to agree in order to accept an argument, and a set of programs with which it can cooperate to build arguments. Besides this distributed nature, the semantics in [5] also allows for paraconsistent forms of argumentation. In fact, it was also a goal of that work to be able to deal with mutually inconsistent, and even inconsistent, knowledge bases. Moreover, when in presence of contradiction we

⋆ The work was partially supported by the Brazilian CAPES, and by the European Commission within the 6th Framework Programme project REWERSE, number 506779.

wanted to obtain ways of agent reasoning, ranging from consistent (in which inconsistencies lead to no result) to paraconsistent. For achieving this, we considered strong and weak arguments.

The paraconsistency in the argumentation also yield a refinement of the possible status of arguments: besides the justified, overruled and defensible arguments as in [16], justified arguments may now be contradictory, based on contradiction or non-contradictory. Moreover, in some applications it might be interesting to change easily from a paraconsistent to a consistent way of reasoning (or vice-verse).

In this paper we focus on the properties of that semantics in what regards paraconsistency which are interesting by themselves, and independent from its distributed nature. With this purpose, we restrict our attention to the special case of the semantics in [5], where only a single logic programs is in the set of programs. Moreover, we provide a notion of proof for an argument for that semantics in that class.

In the next section we present a version of the proposed declarative semantics simplified for the case of a single program, study some of its most significant properties regarding paraconsistency, and illustrate it in one example. We then define the proof method for it, and end with some conclusions. Due to lack of space all proofs have been removed from this version of the paper, and they can be found in a longer version available as a technical report from the first author.

2 Paraconsistent Argumentation Semantics

As motivated in the introduction, in our framework [5] the knowledge base of an agent is modelled by a logic program. More precisely, we use *Extended Logic Program with denials* (ELPd), itself an extension of Extended Logic Programs [11] for modelling the knowledge bases. Besides default and explicit negation, as usual in extended logic programs, we allow a program to have denials of the form

$$\bot \leftarrow L_1, \ldots, L_l, not\ L_{l+1}, \ldots, not\ L_n\ (0 \leq l \leq n)$$

where each of the L_is is an objective literal (i.e. an atom A in the language of the program, or an explicitly negated atom $\neg A$). In other words, denial are simply rules where the head is the special, reserved, symbol $\bot$.

An argument for some objective literal L is a *complete well-defined sequence* concluding L over a *set of rules* of the knowledge base Kb. By *complete* here we mean that all rules required for concluding L are in the sequence. By *well-defined sequence* we mean a (minimal) sequence of rules concluding L as follows: the head of the last rule in the sequence is an objective literal L; furthermore, if some atom L' (ignoring default literals) appears in the body of a rule then there must be a rule before this one with L' in the head; moreover, the sequence must not be circular and only use rules that are strictly necessary.

Definition 1 (Complete Well-defined Sequence). *Let P be an ELPd, and L an objective literal in the language of P. A* well-defined sequence *for L over a set of rules S is a finite sequence $[r_1; \ldots; r_m]$ of rules r_i from S of the form $L_i \leftarrow Body_i$ such that*

- *L is the head of the rule r_m, and*

– *an objective literal L' is the head of a rule r_i ($1 \leq i < m$) only if L' is not in the body of any r_k ($1 \leq k \leq i$) and L' is in the body of some rule r_j ($i < j \leq m$).*

We say that a well-defined sequence for L is complete *if for each objective literal L' in the body of the rules r_i ($1 \leq i \leq m$) there is a rule r_k ($k < i$) such that L' is the head of r_k.*

By the conclusions *of a sequence we mean the set of all objective literals in the head of some rule of the sequence, and by the* assumptions *we mean the set of all default literal in bodies.*

For dealing with consistent and paraconsistent reasoning, we define strong and weak arguments, based on strong and weak sets of rules, the former being simply the rules in the Kb . A *weak set of rules* results from adding to all rule bodies the default negation of the head's complement, and of $\bot$, thus making the rules weaker (more susceptible to being contradicted/attacked). Intuitively, if there is a potential inconsistency, be it by proving the explicit complement of a rules head or by proving $\bot$ then the weak argument is attacked, whereas the strong is not.

Definition 2 (Strong and Weak Arguments). *Let P be an ELPd, and L a literal in its language. Let the* weak set of rules of P *be defined as*

$$R_P^w = \{\ L \leftarrow Body, not\ \neg L, not\ \bot \mid L \leftarrow Body \in P\ \}$$

A strong *(resp.* weak*)* argument *of P for L, A_L^s (resp. A_L^w), is a complete well-defined sequence for L over P (resp. R_P^w).*

Let A_L^w and A_L^s be two arguments of P. A_L^w is the weak argument corresponding *to A_L^s, and vice-verse, if both use exactly the same rules of the original program P (the former by having rules R_P^w and the latter from P alone).*

We say that A_L is an argument *of P for L if it is either a strong argument or a weak one of P for L. We also say that A_L^k is a* k-argument of P *for L (where k is either s, for strong arguments, or w, for weak ones).*

After defining how arguments are built, we now move on to defining the attacking relation between these arguments. By using two kinds of arguments, strong and weak arguments as just exposed, we may rely on a single kind of attack. Indeed the different kinds of attacks usually considered in argumentation semantics for extended logic programs, *undercuts* and *rebuts* as in [15], can be captured by a single notion of attack. If an argument for an objective literal L (denoted by A_L) has a default negation $not\ L'$ in it, any argument for L' attacks (by undercut) A_L. The rebut attacking relation states that an argument also attacks another one when both arguments have complementary conclusions (i.e. one concludes L and the other $\neg L$). It is easy to see that with strong and weak arguments, *rebut can be reduced to undercut*: rebutting reduces to undercut attacks to weak arguments.

In our definition of attacks care must be taken in what regards arguments for $\bot$. By simply using undercut attacks any argument for $\bot$ attacks every weak argument. However, it does not make sense to attack arguments for objective literals if they do not

lead to *falsity*. Informally, an objective L literal leads to *falsity*if there is an argument A_L such that $A_\perp$ is built based on such an argument, e.g.

$$A^s_\perp : A^s_L + [\perp \leftarrow L, not\ L']$$

We only consider objective literals that are in the body of the rule for $\perp$ because these literals immediately lead to *falsity*. We assume that the involvement of other objective literals are not as strong as those in the body of the denial[3]. Then objective literals are *directly conflicting with* $A_\perp$ if the following holds:

Definition 3 (Directly Conflict with $A_\perp$). *Let $A_\perp$ be an argument for $\perp$, '$\perp \leftarrow Body$' be the rule in $A_\perp$ and $\{L_1, \ldots, L_n\}$ be the set of all objective literals in $Body$. The set of objective literals* directly conflicting with $A_\perp$ *is*

$$DC(A_\perp) = \{\perp\} \cup \{L_1, \ldots, L_n\}.$$

Definition 4 (Attack). *Let P be an ELPd. An argument A_L of P for L* attacks *an argument $A_{L'}$ of P for L' iff*

- *L is the symbol $\perp$, $not\ \perp$ belong to the body of some rule in $A_{L'}$, and $L' \in DC(A_L)$; or*
- *L is an objective literal different from $\perp$, and $not\ L$ belongs to the body of some rule in $A_{L'}$.*

Since attacking arguments can in turn be attacked by other arguments, comparing arguments is not enough to determine their acceptability w.r.t. the set of overall arguments. What is also required is a definition that determines the acceptable arguments on the basis of all the ways in which they interact, by proposing arguments and so opposing them. A subset S of proposed arguments of P is acceptable only if the set of all arguments of P does not have some valid opposing argument attacking the proposed arguments in S. As in [8, 15], we demand acceptable sets to contain all such arguments. Two questions remain open: how to obtain opposing arguments and, among these, which are valid?

An opposing argument for a proposed argument which makes an assumption, say $not\ L$, is simply an argument for a conclusion L. For an opposing argument A^o to be valid for attacking a proposed argument A^p in S, S should not have another argument that, in turn, attacks A^o (i.e. another argument that reinstates[4] A^p). In this case, we say that S cannot defend itself against A^o. This motivation points to a definition of acceptable sets of arguments S^i in P such as a set S is *acceptable* if it can attack all opposing arguments. So, we can say that a proposed argument A^p is acceptable w.r.t. a set S of acceptable arguments if and only if each opposing argument A^o attacking A^p is (counter-)attacked by an argument in S.

[3] We further assume they can be detected in a process of "belief revision", e.g. [3]. However, a discussion of this issue is beyond the scope of this proposal.

[4] The key observation is that an argument A that is attacked by another argument B can only be acceptable if it is *reinstated* by a third argument C, i.e by an acceptable argument C that attacks B.

However, it is still necessary to determine how strong arguments and weak arguments should interact w.r.t. such a set S of arguments. Based on the idea of reinstatement, both attacked and counter-attacking arguments should be of the same kind. For instance, if a proposing argument is strong (resp. weak) then every counter-attack against its opposing argument should be strong (resp. weak). A similar reason can be applied for opposing arguments. Therefore, proposed (resp. opposing) arguments should be of the same kind.

In the remainder of this paper we will use the notation p and o to distinguish the proposed argument from the opponent one, i.e. p (resp. o) is a (strong or weak) proposed (resp. opponent) argument. Since there are four possibilities of interaction between a proposed argument, A^p, and an opposing argument, A^o, the definition of arguments' acceptability (and the corresponding characteristic function) is generalised by parametrising the possible kinds of arguments, viz. strong arguments and weak arguments.

Definition 5 (Acceptable Argument). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument, $\mathcal{A}rgs^p(P)$ ($\mathcal{A}rgs^o(P)$) be the set of all arguments in P of kind p (resp. o) , and $S \subseteq \mathcal{A}rgs^p(P)$. An argument $A_L \in \mathcal{A}rgs^p(P)$ is an* acceptable$_{p,o}$ argument *w.r.t. S iff each argument $A_{L'} \in \mathcal{A}rgs^o(P)$ attacking A_L is attacked by an argument $A_{L''} \in S$.*

Note that this proposal is in accordance with the 'Compositional Principle' of [20]: "If an argument SA is a sub-argument of argument A, and SA is not acceptable w.r.t. a set of arguments S, then A is also not acceptable w.r.t. S". We now formalise the concept of acceptable arguments with a fixpoint characteristic function $p\,o$ of P:

Definition 6 (Characteristic Function). *Let P be an ELPd, and p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of P, and $S \subseteq \mathcal{A}rgs^p(P)$. The* characteristic function $p\,o$ *of P and over S is:*

$$F_P^{p,o} : 2^{\mathcal{A}rgs(P)} \rightarrow 2^{\mathcal{A}rgs(P)}$$
$$F_P^{p,o}(S) = \{Arg \in \mathcal{A}rgs(P) \mid Arg \text{ is } acceptable_{p,o} \text{ w.r.t. } S\}.$$

It can be proven that this function is monotonic, and so it has a least fixpoint that can be obtained iteratively as usual:

Proposition 1. *Define for any P the following transfinite sequence of sets of arguments:*

- $S^0 = \emptyset$
- $S^{i+1} = F_P^{p,o}(S^i)$
- $S^\delta = \bigcup_{\alpha<\delta} S^\alpha$ *for limit ordinal δ*

Given that $F_P^{p,o}$ is monotonic, there must exist a smallest λ such that S^λ is a fixpoint of $F_P^{p,o}$, and $S^\lambda = lfp(F_P^{p,o})$.

Note that $lfp(F_P^{p,o})$ is well-behaved, i.e. arguments in it are $acceptable_{p,o}$ w.r.t. the set of all argument of P. By definition $lfp(F_P^{p,o})$ is minimal, which guarantees that it does not contain any argument of which acceptance is not required. Moreover, when $F_P^{p,o}$ is finitary the iterative process above is guaranteed to terminate after an enumerable number of steps.

Proposition 2. *$F_P^{p,o}$ is $finitary$ if each argument in S is attacked by at most a finite number of arguments in S.*

By knowing the set of all acceptable$_{p,o}$ arguments of P, we can split all arguments from $\mathcal{A}rgs(P)$ into three classes: justified arguments, overruled arguments and defensible arguments. Our definition of overruled is different from [15]'s proposal. In its proposal, the restriction applies that overruled arguments cannot be also justified and so [15]'s argumentation semantics is always consistent. Since we aim to obtain a paraconsistent way of reasoning, the status of an argument is defined as follows:

Definition 7 (Justified, Overruled or Defensible Argument). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of an argument of P, and $F_P^{p,o}$ be the characteristic function p o of P. An argument A_L^p is*

- justified$_P^{p,o}$ *iff it is in $lfp(F_P^{p,o})$*
- overruled$_P^{p,o}$ *iff the A_L^o corresponding to A_L^p is attacked by a justified$_P^{p,o}$ argument*
- defensible$_P^{p,o}$ *iff it is neither a justified$_P^{p,o}$ nor an overruled$_P^{p,o}$ argument*

We denote the $lfp(F_P^{p,o})$ by $JustArgs_P^{p,o}$.

We may also iteratively obtain overruled arguments based on the greatest fixpoint of the characteristic function which, by monotonicity of the characteristic function is guaranteed to exist and can also be obtained iteratively as usual. In fact:

Lemma 1. $gfp(F_P^{o,p}) = \{A_{L_1}^o : \neg(\exists A_{L_2}^p \in lfp(F_P^{p,o}) \mid A_{L_2}^p \text{ attacks } A_{L_1}^o)\}$

Lemma 2. $lfp(F_P^{p,o}) = \{A_{L_1}^p : \neg(\exists A_{L_2}^o \in gfp(F_P^{o,p}) \mid A_{L_2}^o \text{ attacks } A_{L_1}^p)\}$

Then, the following holds:

Theorem 1. *A_L^p is overruled$_P^{p,o}$ iff the A_L^o corresponding to A_L^p is not in $gfp(F_P^{o,p})$.*

Due to space limitations we do not detail here general properties when some other weaker restriction are imposed. Instead, we discuss some properties of $JustArgs_P^{p,o}$ and comparisons. Since p (resp. o) denote the kind of a proposed (resp. an opposing) argument, i.e. strong argument or weak argument, assume that p (resp. o) in $\{s, w\}$. Both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ are both conflict-free[5] and non-contradictory[6]. Thus, every argument in both $JustArgs_P^{w,w}$ and $JustArgs_P^{w,s}$ is non-contradictory, i.e. it is not related to a contradiction at all. Furthermore, $F_P^{w,w}$ has more defensible arguments than $F_P^{w,s}$. Therefore, we obtain a consistent way of reasoning in Ag if we apply $F_P^{w,w}$ over $\mathcal{A}rgs(P)$.

In contrast, $JustArgs_P^{s,s}$ and $JustArgs_P^{s,w}$ may be contradictory. However, to evaluate the acceptability of available arguments without considering the presence of *falsity* or both arguments for L and $\neg L$, the proposed arguments should be strong ones, and every opposing argument is a weak argument. Since $F_P^{s,w}$ respects the 'Coherence Principle' of [13, 1], i.e. given that every opposing argument is a weak one, it can be

[5] A set S of arguments is conflict-free if there is no argument in S attacking an argument in S.

[6] A set S of arguments is non-contradictory if neither an argument for *falsity* nor both arguments for L and $\neg L$ are in S.

attacked by any proposed argument for its explicit negation. Therefore, we obtain a paraconsistent way of reasoning if we apply $F_P^{s,w}$ over $\mathcal{A}rgs(P)$.

Moreover, a justified$_P^{s,w}$ argument of an agent can be related to a contradiction with respect to $JustArgs_P^{s,w}$ as follows. We first define that an argument that reinstate another argument is its *counter-attack*:

Definition 8 (Counter-Attack). *Let P be an ELPd, S a set of arguments from P, A_L be an argument in S, and $A_{L'}$ be an argument of P attacking A_L. A* counter-attack *for A_L against $A_{L'}$ is an argument in S that attacks $A_{L'}$. $CA_{A_L}(A_{L'}, S)$ is the set of all counter-attacks for A_L against $A_{L'}$ in S*

Definition 9 (Relation to a Contradiction). *Let P be an ELPd. A justified$_P^{s,w}$ argument A_L^s is*

- contradictory$_P^{s,w}$ *if $JustArgs_P^{s,w}$ is contradictory w.r.t. L, or there exists a contradictory$_P^{s,w}$ argument $A_\perp^s$ and $L \in DC(A_\perp^s)$; or*
- based-on-contradiction$_P^{s,w}$ *if for all $A_{L'}^w$ attacking A_L^s there exists a contradictory$_P^{s,w}$ or based-on-contradiction$_P^{s,w}$ argument in $CA_{A_L}(A_{L'}^w, JustArgs_P^{s,w})$, or there exists an L' in the head of some rule in A_L^s, different from L and $\perp$, such that $JustArgs_P^{s,w}$ is contradictory w.r.t. L'; or*
- non-contradictory$_P^{s,w}$ *iff it is neither contradictory$_P^{s,w}$ nor it is based-on-contradiction$_P^{s,w}$.*

Proposition 3. *A justified$_P^{s,w}$ argument A_L^p is non-contradictory$_P^{s,w}$ if for no head L' of a rule in A_L^s, $JustArgs_P^{s,w}$ is contradictory w.r.t. L', and every counter-attack for A_L^s is a non-contradictory$_P^{s,w}$ argument.*

A *truth value* of an agent's conclusion in a (consistent or paraconsistent) way of reasoning is as follows:

Definition 10 (Truth Value of a Conclusion). *Let P be an ELPd, and $L \in \mathcal{H}(P)$, and $k \in \{s, w\}$. A literal L over P is*

- false$_P^{k,w}$ *iff every k-argument for L is overruled$_P^{k,w}$*
- true$_P^{k,w}$ *iff there exists a justified$_P^{k,w}$ argument for L. Moreover, L is*
  - contradictory$_P^{k,w}$ *if L is the symbol $\perp$ or there exists a justified$_P^{k,w}$ argument for $\neg L$*
  - based-on-contradiction$_P^{k,w}$ *if it is both true$_P^{k,w}$ and false$_P^{k,w}$*
  - non-contradictory$_P^{k,w}$*, otherwise*
- undefined$_P^{k,w}$ *iff L is neither true$_P^{k,w}$ nor false$_P^{k,w}$ (i.e. there is no justified$_P^{k,w}$ argument for L and at least one k-argument for L is not overruled$_P^{k,w}$).*

Example 1 (Privacy of Personal Life – PPL). Usually, any person deserves privacy with respect to her personal life. However, when such a person behaves in a way that is not acceptable (e.g. selling drugs) she will suffer the consequences. The first consequence is the focus of media attention on her personal life and consequent loss of privacy. The personal life of such a person might be exposed by the "results" of media attention (e.g.

photos, reports, and so on), unless there is a law that protects her against it. The above description can be expressed by the following extended logic programming rules.

$$focusOfMediaAttention(X) \leftarrow person(X),\ \neg acceptableBehavior(X).$$
$$\neg acceptableBehavior(X) \leftarrow involved(X,Y),\ againstSociety(Y).$$
$$\neg hasPrivacy(X) \leftarrow focusOfMediaAttention(X).$$
$$personalLifeExposed(X) \leftarrow \neg hasPrivacy(X), not\ protectedByLaw(X).$$
$$hasPrivacy(X) \leftarrow person(X),\ not\ \neg hasPrivacy(X).$$

In contrast, it is considered an absurdity that someone may lose her privacy when she is involved in some event for which there is no evidence of being public (e.g. someone starting a long-term treatment for drugs dependency). The absurdity in the rule below is represented as a denial:

$$\bot \leftarrow \neg hasPrivacy(X),\ event(X,Y),\ not\ publicEvent(Y).$$

Moreover, modern society normally tries to protect children, and so their privacy is guaranteed until evidence appears of some unusual behaviour (e.g. by having unacceptable behaviour).

$$hasPrivacy(X) \leftarrow child(X),\ not\ unusualChild(X).$$
$$unusualChild(X) \leftarrow child(X), \neg acceptableBehavior(X).$$
$$person(X) \leftarrow child(X).$$

However, famous persons are inherently the focus of media attention:

$$focusOfMediaAttention(X) \leftarrow famousPerson(X).$$
$$person(X) \leftarrow famousPerson(X).$$

Assume an agent Ag with the knowledge above, plus some facts about Potira and Ivoti [7]. Potira is a famous child, and Ivoti is a famous soccer player in treatment for drugs dependency:

$$child(potira). \qquad famousPerson(potira).$$
$$famousPerson(ivoti). \quad event(ivoti, treatmentForDrugsDependency).$$

Figure 1 illustrates, with obvious abbreviations, the possible attacks of arguments for "privacy of Potira's life" over $\mathcal{A}rgs(PPL)$. The notation for that figure is as follows: Arguments are represented as nodes. A solid line from argument A to argument B means "A attacks B", a dotted line from A to B means "A is built based on B", and a line with dashes means "A reinstates B". A round node means "it is an acceptable argument" and a square node means "it is not an acceptable argument", which are w.r.t. the set of arguments of $\mathcal{P}$. Then we can presume both the status of the arguments and the truth value of the conclusions of PPL.

[7] The following names are from Native South American, more specifically from the Tupi-Guarani family, Potira and Ivoti both mean "flower".

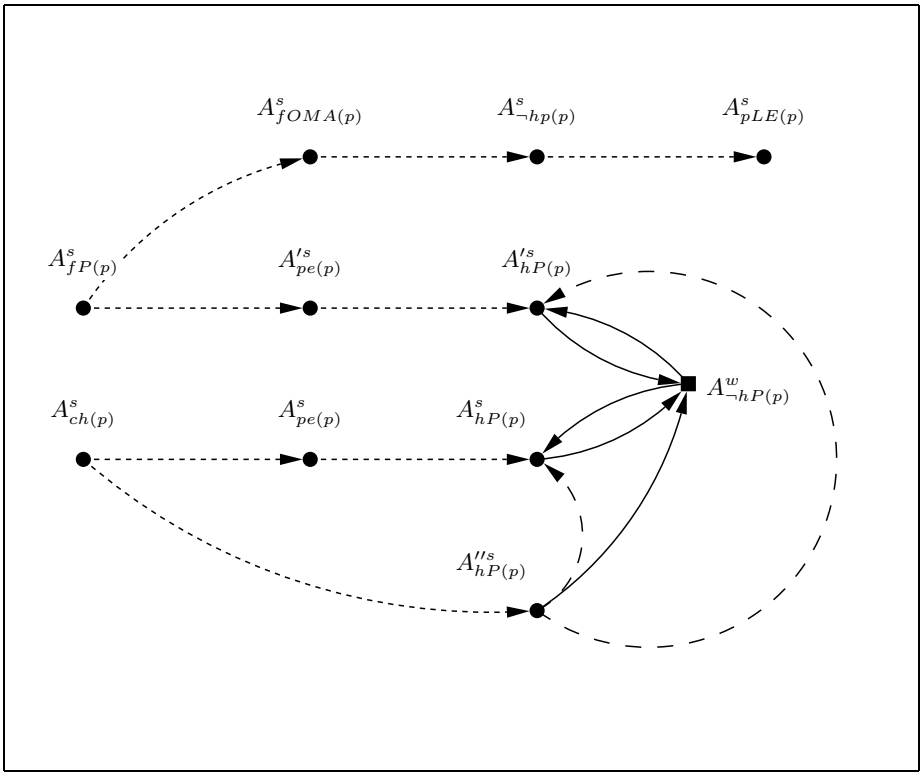

Fig. 1. Acceptable arguments in $Args(PPL)$ for Potira

The argument for "Potira has no Privacy" ($A^s_{\neg hp(p)}$) and also the arguments for "Potira has privacy" ($A^s_{hp(p)}$, $A'^s_{hp(p)}$, $A''^s_{hp(p)}$) are contradictory$^{s,w}_{PPL}$; the argument "Portira has her personal life exposed" ($A^s_{pLE(p)}$) is either based-on-contradiction$^{s,w}_{PPL}$ and overruled$^{s,w}_{PPL}$. The other arguments are non-contradictory$^{s,w}_{PPL}$. Therefore, the truth values for conclusions about Potira are as follows:

- $fP(p)$, $ch(p)$, $fOMA(p)$ and $pe(p)$ are non-contradictory$^{s,w}_{PPL}$;
- $hp(p)$ and $\neg hp(p)$ are both (true$^{s,w}_{PPL}$ and) contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$; and
- $pLE(p)$ is both based-on-contradiction$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$.

Moreover, the truth values for conclusions regarding Ivoti are as follows:

- $fP(i)$, $pe(p)$ and $fOMA(i)$ are non-contradictory$^{s,w}_{PPL}$;
- $hp(i)$ and $\neg hp(i)$ are both contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$; and
- "There is *falsity* in PPL" (i.e. $\bot$) is both (true$^{s,w}_{PPL}$ and) contradictory$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$. Then
- $ev(i, TFDD)$ and $pLE(i)$ are both based-on-contradiction$^{s,w}_{PPL}$ and false$^{s,w}_{PPL}$.

3 A proof for an argument

Though the declarative semantics just exposed may rely on an iterative procedure, its usage for computing arguments may not always be appropriate. This is specially the case when we are only interested in the proof for a (query) argument, rather than all acceptable arguments, as is obtained by the iterative process. Such a query oriented proof procedure can be viewed as conducting a "dispute between a proponent player and an opponent player" in which both proponent and opponent exchange arguments. In its simplest form, the dispute can be viewed as a sequence of alternating arguments:

$$PR_0, OP_0, PR_1, \ldots, PR_i, OP_{i+1}, PR_{i+2}, \ldots$$

The proponent puts forward an initial argument PR_0. For every argument PR_i put forward by the proponent, the opponent attempts to respond with an attacking argument OP_{i+1} against PR_i. For every attacking argument OP_{i+1} put forward by the opponent, the proponent attempts to counter-attack with a proposed argument PR_{i+2} against OP_i. To win the dispute, the proponent needs to have a proposed argument against every opposing argument of the opponent. Therefore, a winning dispute can be represented as a dialogue tree, which represents the top-down, step-by-step construction of a proof tree. We follow [15]'s proposal, which defines a proof for an argument A_L as a dialogue tree for A_L. However, our definition of dialogue tree is in accordance with the acceptability of the arguments of an ELPd P (see Def. 5):

> A proposed argument $A_L \in \mathcal{A}rgs^p(P)$ is acceptable if all of its opposing arguments in $\mathcal{A}rgs^o(P)$ are attacked by acceptable arguments from $\mathcal{A}rgs^p(P)$.

To define a dialogue tree for an argument A_L we need first a definition of *dialogue for an argument*. A dialogue for A_L is a sequence of PR and OP moves of proposed arguments and opposing arguments, such that the first PR move is the argument A_L. Each OP (resp. PR) move of a dialogue consists of an argument from $\mathcal{A}rgs^o(P)$ (resp. $\mathcal{A}rgs^p(P)$) attacking the previous proposed (resp. opposing) argument in such a dialogue. Intuitively, we can see that every PR move wants the conclusion of A_L to be acceptable, and each OP move only wants to prevent the conclusion of A_L from being acceptable. In the case of PR moves, we can further say that if we impose a restriction that proposing arguments cannot be used more than once in a sequence of moves of a dialogue, then the dialogue will have a finite sequence of PR and OP moves. Therefore, none of the proposed arguments can be used more than once in the same dialogue, but any of the opposing arguments can be repeated as often as required to attack a proposed argument.

Definition 11 ($dialogue^{p,o}_{A_L}$**).** *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of a proposed (resp. an opposing) argument of P, and $\mathcal{A}rgs^p(P)$ and $\mathcal{A}rgs^o(P)$ be the set of p-arguments and o-arguments of P, respectively. A* dialogue *p o (in P) for an argument $A_L \in \mathcal{A}rgs^p(P)$, called $dialogue^{p,o}_{A_L}$, is a finite non-empty sequence of m moves $move_i = A_{L_i}$ ($1 \leq i \leq m$) such that*

1. $move_1 = A_L$

2. *for every* $1 < i \leq m$, A_{L_i} *attacks* $A_{L_{i-1}}$ *and*
   - *if* i *is odd then* $A_{L_i} \in \mathcal{A}rgs^p(P)$ *and there is no odd* $j < i$ *such that* $A_{L_j} = A_{L_i}$, *or*
   - *if* i *is even then* $A_{L_i} \in \mathcal{A}rgs^o(P)$.

We say that $move_i$ *is odd if* i *is odd; otherwise,* $move_i$ *is even.*

A dialogue for A_L *succeeds* if its last move is a PR move. In this proposal, we want to guarantee that a dialogue tree for an argument A_L is finitary (cf. Prop. 2). Nevertheless, we only consider grounded finite ELPd in order to relate the declarative semantics (presented in the previous section) to this proposal of operational semantics. By considering this, every dialogue in such a dialogue tree finishes because there will always be a last move PR (resp. OP) in such a dialogue, so no opposing (resp. proposed) argument against it exists. For non-grounded (infinite) programs, there may be (failed) dialogues with an infinite sequence of moves. In such a case, these dialogues should be considered failures, and the argument for such a dialogue tree should be deduced as defensible. The main problem of such an approach is detecting an infinite sequence of moves in a dialogue. However, the following definition will consider cases of both 'grounded finite ELPd' and 'non-grounded (infinite) ELPd'.

Definition 12 (The Status of a dialogue). *Let* P *an ELPd. A dialogue* $p\ o$ *(in* P*) for an argument* $A_L \in \mathcal{A}rgs^p(P)$ *is* completed *iff its last move is* m*, and*

- *if* m *is odd then there is no argument in* $\mathcal{A}rgs^o(P)$ *attacking* A_{L_m}*, or*
- *if* m *is even then there is no argument in* $\mathcal{A}rgs^p(P) - S_p$ *attacking* A_{L_m} *where* S_p *is the set of all* A_{L_j} *in the sequence such that* j *is odd*

(or it is infinite*). A completed dialogue is* failed *iff its last move is* odd *(or it is infinite); otherwise, it* succeeds.

Note that a dialogue$^{p,o}_{A_L}$ in P and the $lfp(F^{p,o}_P)$ "grow up" in different ways. In the former, an argument A in the last move, $move_f$, is not attacked by any argument in $\mathcal{A}rgs(P)$. Since A attacks the previous move, $move_{f-1}$, we can say that the argument B in $move_{f-2}$ was reinstated by A. Thus, each $move_i$ ($1 \leq i < f-1$) is reinstated by $move_{i+2}$. The latter evaluates argument A as acceptable in the first iteration of the characteristic function $F^{p,o}_P$. In the second iteration, A reinstates B, so that B is acceptable and might reinstate other arguments in all following iterations. We can further say that dialogue$^{p,o}_{A_L}$ decreases (in a top-down way) and $lfp(F^{p,o}_P)$ increases (in a bottom-up way) the set of evaluated arguments.

Proposition 4. *Let* $move_m = A_L$ *be the last move of a succeeded dialogue*$^{p,o}_{A'_L}$ *in* P*. Then,* $A_L \in F^{p,o}_P(\emptyset)$.

A dialogue tree DT for A_L is held between a proposed argument PR and its opposing argument OP against PR, where the root of DT is A_L. The dialogue tree DT considers all possible ways in which A_L can be attacked because each branch of DT is a dialogue for A_L, i.e. every single dialogue for A_L is built because we should consider the overall arguments in $\mathcal{A}rgs(P)$ to deduce the status of A_L. The dialogue tree DT for an argument A_L succeeds if every dialogue of DT succeeds.

Definition 13 ($DT^{p,o}_{A_L}$). *Let P be an ELPd, p (resp. o) be the kind (strong or weak) of the proposed (resp. opposing) argument of $\mathcal{A}rgs(P)$, and $\mathcal{A}rgs^p(P)$ (resp. $\mathcal{A}rgs^o(P)$) be the set of p-arguments (o-arguments) of P. A* dialogue tree *$p\ o$ (in P) for $A_L \in \mathcal{A}rgs^p(P)$, called $DT^{p,o}_{A_L}$, is a finite tree of moves $move_i = A_{L_i}$ ($i > 0$) such that*

1. *each branch of $DT^{p,o}_{A_L}$ is a $dialogue^{p,o}_{A_L}$, and*
2. *for all i, if $move_i$ is*
   - *even then its only child is a p-argument attacking $A_{L_i} \in \mathcal{A}rgs^o(P)$, or*
   - *odd then its children are all o-arguments attacking $A_{L_i} \in \mathcal{A}rgs^p(P)$*

A $DT^{p,o}_{A_L}$ succeeds *iff all branches (i.e. all $dialogue^{p,o}_{A_L}$) of the tree succeeds.*

Based on the second condition of Definition 13, we might obtain more than one dialogue tree for an argument. This occurs because only one proponent's move is built for each opponent's move of a dialogue tree. For instance,

Example 2. Let $P = \{p \leftarrow not\ a;\ a \leftarrow not\ b, not\ c;\ a \leftarrow not\ d;\ b;\ c \leftarrow not\ g;\ g\}$. There are two possible $DT^{s,s}_{A_p}$ in P: the first dialogue tree does not succeed because there is a last move which is an o-argument, viz $[a \leftarrow not\ b]$; the second one also does not succeed because every last move is an o-argument, viz $[g]$ and $[a \leftarrow not\ d]$.

At this point we can relate, for grounded finite programs, the results from a $DT^{p,o}_{A_L}$ to the status of the argument A_L (see Def. 7), as follows:

Proposition 5. *An argument A^p_L in a grounded finite P is*

- *justified$^{p,o}_P$ iff there exists a successful $DT^{p,o}_{A_L}$*
- *overruled$^{p,o}_P$ iff for all $DT^{p,o}_{A_L}$: there exists a $move_2 = A^o_{L'}$ such that $DT^{o,p}_{A_{L'}}$ succeeded*
- *defensible$^{p,o}_P$ iff it is neither justified$^{p,o}_P$ nor overruled$^{p,o}_P$.*

The following example illustrate the concepts presented in Proposition 5.

Example 3. Let $P2 = \{a \leftarrow not\ b;\ \neg a;\ b;\ \neg b;\ c;\ \bot \leftarrow not\ c\}$. On the top of Figure 2, it is illustrated the possible $DT^{w,w}_{A_L}$ in $P2$. Note that each dialogue tree does not succeed because its last move is an o-argument. Nevertheless, all arguments are defensible$^{w,w}_{P2}$ because none of these last moves are justified$^{w,w}_{P2}$. On the bottom of the Figure 2 it is also illustrated the possible $DT^{s,w}_{A_L}$ in $P2$. In such a case, all arguments are justified$^{s,w}_{P2}$.

Proposition 6. *A justified$^{s,w}_P$ argument A^s_L in a finite ground P is*

- contradictory$_{s,w}$ *iff L is the symbol $\bot$, or different from $\bot$ and there exists at least a successful $DT^{s,w}_{A_{\neg L}}$; or*
- based-on-contradiction$_{s,w}$ *iff A^s_H is not contradictory$_{s,w}$ and*
  - *there exists a contradictory^{s,w} $A^s_{L'}$ (with a rule $L' \leftarrow Body$) such that $L \in Body$, or*
  - *there exists an L'' in the head of a rule in A^s_L such that $A^s_{L''}$ is contradictory^{s,w}, or*

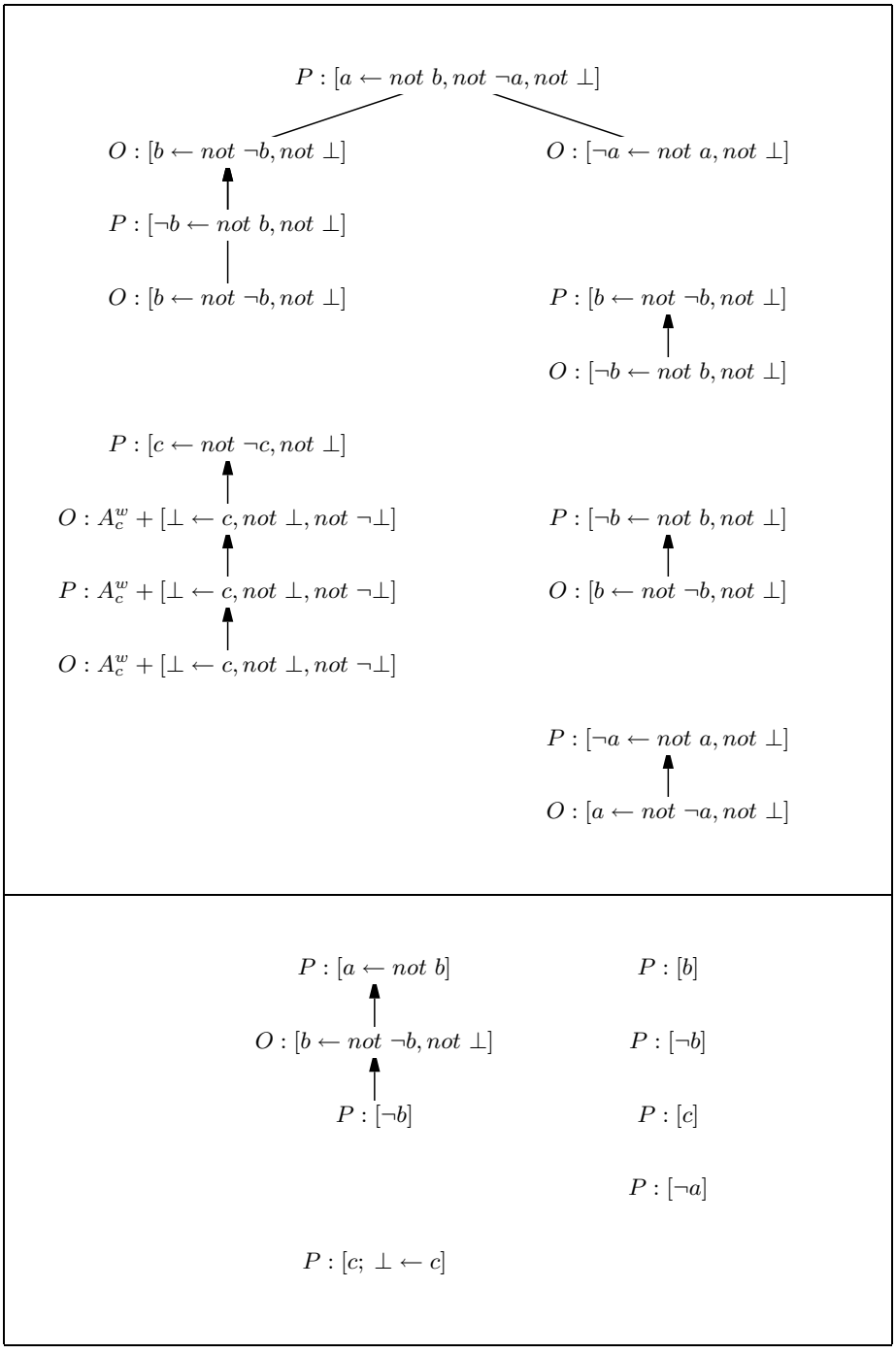

Fig. 2. Some $DT^{w,w}_{A_L}$ and $DT^{s,w}_{A_L}$ in $\{a \leftarrow not\ b; \neg a; b; \neg b; c; \bot \leftarrow not\ c\}$

  - *for all* $dialogue^{s,w}_{A_H}$ *in* $DT^{s,w}_{A_H}$*: the last move has not a non-contradictory*$_{s,w}$ *argument; or*
- non-contradictory$_{s,w}$*, otherwise.*

To conclude about the truth value of an objective literal L we evaluate more than one dialogue tree of each argument for such L:

Proposition 7. *An objective literal* H *is*

- $true^{p,o}_P$ *iff there exists a successful* $DT^{p,o}_{A_H}$*. Thus,* H *is*
  - *contradictory*$^{p,o}_P$ *iff for all successful* $DT^{p,o}_{A_H}$*:* A^p_H *is contradictory*$^{p,o}_P$*, or*
  - *based-on-contradiction*$^{p,o}_P$ *iff for all successful* $DT^{p,o}_{A_H}$*:* A^p_H *is based-on-contradiction*$^{p,o}_P$*, or*
  - *non-contradictory*$^{p,o}_P$ *iff there exists a successful* $DT^{p,o}_{A_H}$ *such that* A^p_H *is non-contradictory*$^{p,o}_P$*;*
- $false^{p,o}_P$ *(in* P*) iff* $\forall DT^{p,o}_{A_H}$*:* A^p_H *is overruled*$^{p,o}_P$*;*
- $undefined^{p,o}_P$ *(in* P*) iff* $\forall DT^{p,o}_{A_H}$*:* A^p_H *is neither justified*$^{p,o}_P$ *nor overruled*$^{p,o}_P$*.*

Example 4. Following Example 3, all literals of $P2$ are justified$^{s,w}_{P2}$. However, all literals of $P2$ are undefined$^{w,w}_{P2}$.

4 Conclusions and Further Work

Our argumentation semantics is based on the argumentation metaphor, in the line of the work developed in [9, 15, 2, 18] for defining semantics of single extended logic programs. In these argumentation-based semantics, rules of a logic program are viewed as encoding arguments of an agent. More precisely, an argument for an objective literal L is a sequence of rules that "proves" L, if all default literals (of the form $not\ L'$) in the body of those rules are assumed true. In other words, arguments encoded by a program can attack – by undercut – each other. Moreover, an argument for L attacks – by rebut – another argument if this other argument assumes its explicit negation (of the form $\neg L$). The meaning of the program is so determined by those arguments that somehow (depending on the specific semantics) can defend themselves from the attacks of other arguments.

We generalise [15]'s definition of argument by proposing two kind of arguments, viz. strong arguments and weak arguments. By having two kinds of arguments, viz. strong arguments and weak arguments, the attack by undercut is not needed. Simply note that rebut are undercut attacking weak arguments. Therefore, rebut is not considered in our proposal since, as already shown in [17, 6, 18], it can be reduced to undercut by considering weaker versions of arguments. [2] also defines a methodology for transforming non-exact, defensible rules into exact rules with explicit non-provability conditions and shows that this transformation eliminates the need for rebuttal attacks and for dealing with priorities in the semantics.

Similar to [9, 15] we formalise the concept of acceptable arguments with a fixpoint operator. However, the acceptability of an argument might have different results and it depends on which kind of interaction between (strong and weak) arguments is chosen. Therefore, our argumentation semantics assigns different levels of acceptability to an

argument for an objective literal L and so it can be justified, overruled, or defensible. Moreover, a justified argument for L can be contradictory, based on contradiction, or non contradictory. Consequently, a truth value of L can be true (and contradictory, based on contradiction, or non contradictory), false or undefined.

Since our argumentation semantics is parametrised by the kind of interaction between arguments, we obtain results from a consistent way of reasoning to a paraconsistent way of reasoning. A consistent way of reasoning neither concludes that L nor $\neg L$ are true, even if one of these is a fact. A paraconsistent way of reasoning can conclude L is true even if it also concludes that $\neg L$ is true. Given that we consider denials in the agent's knowledge base – in a conflicting situation – a consistent way of reasoning cannot conclude that a given L is true if L is related with the presence of the *falsity*; a paraconsistent way of reasoning might conclude L even it is related with *falsity*. Furthermore, our argumentation semantics (and the corresponding proof procedure) succeeds in detecting conflicts in a paraconsistent extended logic program with denials, i.e. it handles with contradictory arguments and with the presence of *falsity*.

For this proposal we have made two implementations, both in XSB System (by resorting to tabling) [19] which computes the argumentation Prolog implementation over an agent's knowledge base. One bottom-up implementation of the semantics, following closely its declarative definition; another of query-driven proof procedures for the semantics. The proof procedure has also been implemented by using the toolkit Interprolog [4], a middle-ware for Java and Prolog which provides method/predicate calling between both.

As we mentioned, the original semantics, defined in [5], is a generalisation of the one presented here to a distributed argumentation-based negotiation semantics. As future work we intend to generalise this (centralised) proof procedure to a distributed proof procedure seeing the negotiation process as a forest of dialogue trees, rather than a single tree as here.

References

1. J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.
2. A. Bondarenko, P. M. Dung, R. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Journal of Artificial Intelligence*, 93(1–2):63–101, 1997.
3. L. M. Pereira e M. Schroeder C. V. Damásio. Revise: Logic programming and diagnosis. In U. Furbach J. Dix and A. Nerode, editors, *4th International Conference (LPNMR'97)*, volume LNAI 1265 of *Logic Programming and NonMonotonic Reasoning*, pages 353–362. Springer, July 1997.
4. M. Calejo. Interprolog: Towards a declarative embedding of logic programming in java. In J. J. Alferes and J. Leite, editors, *9th European Conference (JELIA 2004)*, LNAI, pages 714–71. Springer, 2004. Toolkit available at http://www.declarativa.com/InterProlog/.
5. Iara Carnevale de Almeida and José Júlio Alferes. An argumentation-based negotiation framework. In K. Inoue, K. Satoh, and F Toni, editors, *VII International Workshop on Computational Logic in Multi-agent Systems (CLIMA)*, volume 4371 of *LNAI*, pages 191–210. Springer, 2006. Revised Selected and Invited Papers.

6. Iara de Almeida Móra and José Júlio Alferes. Argumentative and cooperative multi-agent system for extended logic programs. In F. M. Oliveira, editor, *XIVth Brazilian Symposium on Artificial Intelligence*, volume 1515 of *LNAI*, pages 161–170. Springer, 1998.
7. P. Dung, P. Mancarella, and F. Toni. *Computational Logic: Logic Programming and Beyond – Essays in Honour of Robert A. Kowalski*, volume 2408, chapter Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning, pages 289–310. Springer, 2002.
8. P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *10th International Conference on LP (ICLP)*, pages 616–630. MIT Press, 1993.
9. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Journal of Artificial Intelligence*, 77(2):321–357, 1995.
10. P. M. Dung, R. Kowalski, and F. Toni. Argumentation-theoretic proof procedures for default reasoning. Technical Report. Available at http://www.doc.ic.ac.uk/˜ft/PAPERS/arg03.pdf, May 2003.
11. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Warren and Szeredi, editors, *7th International Conference on LP (ICLP)*, pages 579–597. MIT Press, 1990.
12. R. P. Loui. Process and policy: Resource-bounded non-demonstrative reasoning. *Journal of Computational Intelligence*, 14:1–38, May 1998.
13. L. M. Pereira and J. J. Alferes. Well founded semantics for logic programs with explicit negation. In *European Conference on Artificial Intelligence (ECAI)*, pages 102–106. John Wiley & Sons, 1992.
14. J. L. Pollock. Defeasible reasoning with variable degrees of justification. *Journal of Artificial Intelligence*, 133:233–282, 2002.
15. H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.
16. H. Prakken and G. A. W. Vreeswijk. *Handbook of Philosophical Logic*, volume 4, chapter Logics for Defeasible Argumentation, pages 218–319. Kluwer Academic, 2 edition, 2002.
17. Michael Schroeder, Iara de Almeida Móra, and José Júlio Alferes. Vivid agents arguing about distributed extended logic programs. In Ernesto Costa and Amilcar Cardoso, editors, *Progress in Artificial Intelligence, 8th Portuguese Conference on Artificial Intelligence (EPIA)*, volume 1323 of *LNAI*, pages 217–228. Springer, 1997.
18. R. Schweimeier and M. Schroeder. Notions of attack and justified arguments for extended logic programs. In F. van Harmelen, editor, *15th European Conference on Artificial Intelligence*. IOS Press, 2002.
19. T. Swift and et all. Xsb - a logic programming and deductive database system for unix and windows. Technical report, XSB project, 2003. Toolkit available at *http://xsb.sourceforge.net/*.
20. G. A. W. Vreeswijk. Abstract argumentation systems. *Journal of Artificial Intelligence*, 90(1–2):225–279, 1997.

CaSAPI: a system for credulous and sceptical argumentation

Dorian Gaertner and Francesca Toni

Department of Computing
Imperial College London
Email: {dg00,ft}@doc.ic.ac.uk

Abstract. We present the CaSAPI system, implementing (a generalisation of) three existing computational mechanisms [8–10] for determining argumentatively whether potential beliefs can be deemed to be acceptable and, if so, for computing supports for them. These mechanisms are defined in terms of dialectical disputes amongst two fictional agents: a proponent agent, eager to determine the acceptability of the beliefs, and an opponent agent, trying to undermine the existence of an acceptable support for the beliefs, by finding attacks against it that the proponent needs to counter-attack in turn. The three mechanisms differ in the level of scepticism of the proponent agent and are defined for (flat) assumption-based argumentation frameworks [3]. Thus, they can serve as decision-making mechanisms for all instances of these frameworks. In this paper we show how they can be used for logic programming, legal reasoning, practical reasoning, and agent reasoning.

1 Introduction

Assumption-based argumentation [3] has been proven to be a powerful mechanism to understand commonalities and differences amongst many existing frameworks for non-monotonic reasoning, including logic programming [3]. It has also been studied in the context of legal reasoning [14]. Furthermore, the computational complexity of several instances of assumption-based argumentation frameworks for non-monotonic reasoning has been studied in [7].

Assumption-based argumentation frameworks can be coupled with a number of different semantics, all defined in dialectical terms, some credulous and some sceptical, of various degrees. Different computational mechanisms can be defined to match these semantics. In this paper, we consider three existing such mechanisms: GB-dispute derivations for computing the sceptical grounded semantics [9], AB-dispute derivations for computing the credulous admissible semantics [8, 9] and IB-dispute derivations for computing the sceptical ideal semantics [9, 10].

All mechanisms are defined as "dialogues" between two fictional agents: the proponent and the opponent, trying to establish the acceptability of given beliefs with respect to the chosen semantics. The three mechanisms (and corresponding semantics) differ in the level of scepticism of the proponent agent: in GB-dispute derivations the agent is not prepared to take any chances and is completely

sceptical in the presence of seemingly equivalent alternatives; in AB-dispute derivations the agent would adopt any alternative that is capable of counter-attacking all attacks without attacking itself; in IB-dispute derivations, the agent is wary of alternatives, but is prepared to accept common ground between them.

In this paper we describe the CaSAPI [1] system implementating these mechanisms and we illustrate the system and its potential for application in the context of some application scenarios. The system relies upon a generalisation of the original assumption-based argumentation framework and of the computational mechanisms, whereby multiple contraries are allowed. This generalisation is useful to widen the applicability of assumption-based argumentation (*e.g.* for reasoning about decisions). We provide this generalisation explicitly for AB-dispute derivations. The application scenarios we consider are non-monotonic reasoning (using logic programming), legal reasoning (where different regulations need to be applied, taking into account dynamic preferences amongst them), practical reasoning (where decisions need to be made as to which is the appropriate course of action for a given agent), and reasoning to support autonomous agents (about their individual beliefs, desires and intentions, as well as relationships amongst them). Most of these application scenarios require a mapping from appropriate frameworks into assumption-based argumentation.

The paper is structured as follows: in the next section, we will briefly introduce the concept of assumption-based argumentation and describe the three dispute derivations upon which our system is based. Section 3 presents the generalised assumption-based framework and the generalised AB-dispute derivations that our system implements. Section 4 provides a brief description of the CaSAPI system. Applications of this system to the areas of non-monotonic reasoning and legal, practical and agent reasoning are given in Section 5. Finally, we conclude and discuss future work.

2 Background

The definitions and notions in this section are adapted from [3, 8, 10, 9].

Definition 1. *An* assumption-based argumentation framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ *where*

- $(\mathcal{L}, \mathcal{R})$ *is a deductive system, with a language* $\mathcal{L}$ *and a set* $\mathcal{R}$ *of inference rules,*
- $\mathcal{A} \subseteq \mathcal{L}$ *is a (non-empty) set, whose elements are referred to as* assumptions*,*
- $\overline{\ }$ *is a (total) mapping from* $\mathcal{A}$ *into* $\mathcal{L}$*, where* $\overline{\alpha}$ *is the* contrary *of* α*.*

We will assume that the inference rules in $\mathcal{R}$ have the syntax

$$c_0 \leftarrow c_1, \ldots, c_n$$

with $n > 0$ or

$$c_0$$

[1] CaSAPI stands for Credulous and Sceptical Argumentation: Prolog Implementation.

where each $c_i \in \mathcal{L}$. c_0 is referred to as the *head* and $c_1, \ldots, c_n$ as the *body* of a rule $c_0 \leftarrow c_1, \ldots, c_n$. The body of a rule c_0 is considered to be empty.

As in [8], we will restrict attention to *flat assumption-based frameworks*, such that if $c \in \mathcal{A}$, then there exists no inference rule of the form $c \leftarrow c_1, \ldots, c_n \in \mathcal{R}$.

Example 1. $\mathcal{L} = \{p, a, \neg a, b, \neg b\}$, $\mathcal{R} = \{p \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a\}$, $\mathcal{A} = \{a, b\}$ and $\overline{a} = \neg a$, $\overline{b} = \neg b$.

An *argument* in favour of a sentence or belief x in $\mathcal{L}$ supported by a set of assumptions $X \subseteq \mathcal{A}$ is a *backward* (or *tight*) *deduction* [8] from x to X, via the backward application of rules in $\mathcal{R}$. For the simple assumption-based framework in example 1, an argument in favour of p supported by $\{a\}$ may be obtained by applying $p \leftarrow a$ backwards.

In order to determine whether a belief is to be held, a set of assumptions needs to be identified that would provide an "acceptable" support for the belief, namely a "consistent" set of assumptions including a "core" support as well as assumptions that defend it. This informal definition of "acceptable" support can be formalised in many ways, using a notion of "attack" amongst sets of assumptions:

Definition 2. X attacks Y *iff there is an argument in favour of some* $\overline{y}$ *supported by (a subset of)* X*, where* y *is in* Y*.*

In Example 1 above, $\{b\}$ attacks $\{a\}$. In this paper we are concerned with the following formalisations of the notion of "acceptability":

- a set of assumptions is *admissible*, iff it does not attack itself and it counter-attacks every set of assumptions attacking it;
- *complete*, iff it is admissible and it contains all assumptions it can defend, by counter-attacking all attacks against them;
- *grounded*, iff it is minimally complete;
- *ideal*, iff it is admissible and contained in all maximally admissible sets.

In the remainder of this section we will illustrate, by means of examples, the three forms of dispute derivations presented in [8, 10, 9], for computing grounded, admissible and ideal sets of assumptions (respectively) in support of given beliefs. For any formal details and results see [8, 10, 9].

2.1 GB-dispute derivations

GB-dispute derivations compute grounded sets of assumptions supporting a given input belief. They are finite sequences of tuples

$$\langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i \rangle$$

where $\mathcal{P}_i$ and $\mathcal{O}_i$ represent (the set of sentences held by) the proponent and opponent in a dispute, A_i holds the set of assumptions generated by the proponent in support of its belief and to defend itself against the opponent, and C_i holds

the set of assumptions in attacks generated by the opponent that the proponent has chosen as "culprits" to be counter-attacked. Each derivation starts with a tuple

$$\langle \mathcal{P}_0 = \{x\}, \mathcal{O}_0 = \{\}, A_0 = \mathcal{A} \cap \{x\}, C_0 = \{\} \rangle$$

where x is the belief whose acceptability the derivation aims at establishing. Then, for every $0 \leq i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and [2]:

1. If $\sigma \in \mathcal{P}_i$ is selected then
   (i) if σ is an assumption, then
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$
   (ii) if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup R$
   $A_{i+1} = A_i \cup (\mathcal{A} \cap R)$.
2. If S is selected in $\mathcal{O}_i$ and σ is selected in S then
   (i) if σ is an assumption, then
   (a) either σ is ignored, i.e.
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\}\}$
   (b) or $\sigma \notin A_i$ and
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$
   $A_{i+1} = A_i \cup (\{\overline{\sigma}\} \cap \mathcal{A})$
   $C_{i+1} = C_i \cup \{\sigma\}$
   (ii) if σ is not an assumption, then
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\} \cup R \,|\, \sigma \leftarrow R \in \mathcal{R}\}$.

In Example 1, no GB-dispute derivation for p exists (and, indeed, p is not an acceptable belief according to the grounded semantics) as the search for any such derivation loops and hence no finite sequence of tuples can be found:

$\langle \mathcal{P}_0 = \{p\}, \mathcal{O}_0 = \{\}, A_0 = \{\}, C_0 = \{\} \rangle$,
$\langle \mathcal{P}_1 = \{a\}, \mathcal{O}_1 = \{\}, A_1 = \{a\}, C_1 = \{\} \rangle$, by using rule $p \leftarrow a$,
$\langle \mathcal{P}_2 = \{\}, \mathcal{O}_2 = \{\{\neg a\}\}, A_2 = \{a\}, C_2 = \{\} \rangle$, since $\overline{a} = \neg a$,
$\langle \mathcal{P}_3 = \{\}, \mathcal{O}_3 = \{\{b\}\}, A_3 = \{a\}, C_3 = \{\} \rangle$, by using rule $\neg a \leftarrow b$,
$\langle \mathcal{P}_4 = \{\neg b\}, \mathcal{O}_4 = \{\}, A_4 = \{a\}, C_4 = \{b\} \rangle$, since $\overline{b} = \neg b$,
$\langle \mathcal{P}_5 = \{a\}, \mathcal{O}_5 = \{\}, A_5 = \{a\}, C_5 = \{b\} \rangle$, by using rule $\neg b \leftarrow a$,
...

Note that $\mathcal{P}_1 = \mathcal{P}_5$ and $\mathcal{O}_1 = \mathcal{O}_5$, so both proponent and opponent are repeating their arguments.

[2] For brevity, we indicate here and in all the dispute derivations in the paper only the items of the $i+1$-th tuple that are different from the corresponding items in the i-th tuple: all other items are as in the i-th tuple. For full details, see [9].

2.2 AB-dispute derivations

AB-dispute derivations[3] are modifications of GB-dispute derivation to determine whether beliefs can be held according to the admissible semantics, as follows:

- at step 1.(ii): $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup (R - A_i)$
- step 2.(i)(b) becomes: $\sigma \notin A_i$ and $\sigma \in C_i$ and $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
- a new step 2.(i)(c) is added: $\sigma \notin A_i$ and $\sigma \notin C_i$ and

(c.1) if $\overline{\sigma}$ is not an assumption, then
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
$\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{\overline{\sigma}\}$
$C_{i+1} = C_i \cup \{\sigma\}$

(c.2) if $\overline{\sigma}$ is an assumption, then
$\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
$A_{i+1} = A_i \cup \{\overline{\sigma}\}$
$C_{i+1} = C_i \cup \{\sigma\}$

- at step 2. (ii): $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \big\{S - \{\sigma\} \cup R \,\big|\, \sigma \leftarrow R \in \mathcal{R}, \text{and} R \cap C_i = \{\}\big\}$.

For Example 1, an AB-dispute derivation exists, following up from the earlier GB-derivation with a terminating step

$$\langle \mathcal{P}_5 = \{\}, \mathcal{O}_5 = \{\}, A_5 = \{a\}, C_5 = \{b\} \rangle$$

computing an admissible support $\{a\}$ for p ($\{a\}$ is indeed admissible since it does not attack itself and it counter-attacks $\{b\}$, the only attack against it).

2.3 IB-dispute derivations

IB-dispute derivations are extensions of AB-dispute derivations, in that they are finite sequences of tuples

$$\langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i, \mathcal{F}_i \rangle$$

where the new component $\mathcal{F}_i$ holds the set of all (potential) attacks against $\mathcal{P}_i$. IB-dispute derivations deploy Fail-dispute derivations to check that no admissible extensions of any element in any $\mathcal{F}_i$ exists. For lack of space we simply exemplify IB-dispute derivations here (see [10, 9] for details).

Example 2. $\mathcal{L} = \{a, \neg a, b, \neg b, c, \neg c, d, \neg d\}$, $\mathcal{R} = \{\neg a \leftarrow a; \neg a \leftarrow b; \neg b \leftarrow a; \neg c \leftarrow d; \neg d \leftarrow c\}$, $\mathcal{A} = \{a, b, c, d\}$ and $\overline{x} = \neg x$, for all $x \in \mathcal{A}$.

Given the framework in Example 2, an IB-derivation for $\neg a$ is:

$\langle \mathcal{P}_0 = \{\neg a\}, \mathcal{O}_0 = \{\}, A_0 = \{\}, C_0 = \{\}, \mathcal{F}_0 = \{\} \rangle$
$\langle \mathcal{P}_1 = \{b\}, \mathcal{O}_1 = \{\}, A_1 = \{b\}, C_1 = \{\}, \mathcal{F}_1 = \{\} \rangle$,

[3] AB-dispute derivations are a slight modification of the dispute derivations of [8], presented in [9].

$\langle \mathcal{P}_2 = \{\}, \mathcal{O}_2 = \{\{\neg b\}\}, A_2 = \{b\}, C_2 = \{\}, \mathcal{F}_2 = \{\}\rangle$,
$\langle \mathcal{P}_3 = \{\}, \mathcal{O}_3 = \{\{a\}\}, A_3 = \{b\}, C_3 = \{\}, \mathcal{F}_3 = \{\}\rangle$,
$\langle \mathcal{P}_4 = \{\neg a\}, \mathcal{O}_4 = \{\}, A_4 = \{b\}, C_4 = \{a\}, \mathcal{F}_4 = \{\{a\}\}\rangle$,
$\langle \mathcal{P}_5 = \{\}, \mathcal{O}_5 = \{\}, A_5 = \{b\}, C_5 = \{a\}, \mathcal{F}_5 = \{\{a\}\}\rangle$,
$\langle \mathcal{P}_6 = \{\}, \mathcal{O}_6 = \{\}, A_6 = \{b\}, C_6 = \{a\}, \mathcal{F}_6 = \{\}\rangle$.

The transition between the penultimate and the last tuple in the sequence above requires the existence of a Fail-dispute derivation confirming that no admissible extension of $\{a\} \in \mathcal{F}_5$ exists.

The derivation succeeds in computing support $\{b\}$ for $\neg a$. The set $\{b\}$ is ideal as it is admissible and contained in every maximally admissible set of assumptions (there are two such sets: $\{b, c\}$ and $\{b, d\}$).

Note that there is no GB-dispute derivation for $\neg a$ (which indeed is not supported by any grounded set of assumptions). Also, note that there exists an AB-dispute derivation for $\neg a$, as well as for $\neg c$ and $\neg d$, but no GB- or IB-dispute derivation exists for the latter two beliefs. Thus, the proponent agent in GB-derivations is the most sceptical, followed by the proponent agent in IB-derivations. The proponent agent in AB-derivations on the other hand is completely credulous.

3 Generalisation of assumption-based argumentation frameworks

In order to widen their applicability (*e.g.* for practical reasoning), assumption-based argumentation frameworks need to be generalised as follows:

Definition 3. *A* generalised assumption-based framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{C}on\rangle$ *where* $\mathcal{L}$, $\mathcal{R}$, $\mathcal{A}$ *are as in conventional assumption-based frameworks, and* $\mathcal{C}on$ *is a (total) mapping from assumptions in* $\mathcal{A}$ *into* sets of sentences *in* $\mathcal{L}$.

Intuitively, in this generalised framework, assumptions admit multiple contraries. Given a generalised assumption-based argumentation framework, the notion of attack between sets of assumptions becomes:

Definition 4. X attacks Y *iff there is an argument in favour of some* x *supported by (a subset of)* X *where* $x \in \mathcal{C}on(y)$ *and* y *is in* Y.

All dispute derivations defined in previous works [8, 10, 9] can thus be modified for generalised assumption-based frameworks. In what follows, we show how this can be done for AB-dispute derivations (the modifications are typeset in bold font), as this is the "core" form of dispute derivation (GB-dispute derivations are a simplification and IB-dispute derivation an extension of AB-dispute derivations):

Definition 5. *Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \mathcal{C}on \rangle$ be a generalised assumption-based argumentation framework. Given a selection function, a* generalised AB-dispute derivation of a defence set *A for a sentence α is a finite sequence of* quadruples

$$\langle \mathcal{P}_0, \mathcal{O}_0, A_0, C_0 \rangle, \ldots, \langle \mathcal{P}_i, \mathcal{O}_i, A_i, C_i \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{O}_n, A_n, C_n \rangle$$

where $\mathcal{P}_0 = \{\alpha\}$ $\quad A_0 = \mathcal{A} \cap \{\alpha\}$ $\quad \mathcal{O}_0 = C_0 = \{\}$
$\mathcal{P}_n = \mathcal{O}_n = \{\}$ $\quad A = A_n$

and for every $0 \leq i < n$, only one σ in $\mathcal{P}_i$ or one S in $\mathcal{O}_i$ is selected, and:

1. *If $\sigma \in \mathcal{P}_i$ is selected then*

   (i) *if σ is an assumption, then*
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\}$
   $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{x\} \mid x \in \mathcal{C}on(\sigma)\}$
   *(***In the original AB-dispute derivation,** $\mathcal{O}_{i+1} = \mathcal{O}_i \cup \{\{\overline{\sigma}\}\}$.*)*

   (ii) *if σ is not an assumption, then there exists some inference rule $\sigma \leftarrow R \in \mathcal{R}$ such that $C_i \cap R = \{\}$ and*
   $\mathcal{P}_{i+1} = \mathcal{P}_i - \{\sigma\} \cup (R - A_i)$
   $A_{i+1} = A_i \cup (\mathcal{A} \cap R)$.

2. *If S is selected in $\mathcal{O}_i$ and σ is selected in S then*

   (i) *if σ is an assumption, then*
   (a) *either σ is ignored, i.e.*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \{S - \{\sigma\}\}$
   (b) *or $\sigma \notin A_i$ and $\sigma \in C_i$ and*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   (c) *or $\sigma \notin A_i$ and $\sigma \notin C_i$ and* **chosen some** *$x \in \mathcal{C}on(\sigma)$ (***in the original AB-dispute derivations,** $x = \overline{\sigma}$*)*
   (c.1) *if x is not an assumption, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $\mathcal{P}_{i+1} = \mathcal{P}_i \cup \{x\}$
   $C_{i+1} = C_i \cup \{\sigma\}$
   (c.2) *if x is an assumption* **and it does not belong to** *C_i, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\}$
   $A_{i+1} = A_i \cup \{x\}$
   $C_{i+1} = C_i \cup \{\sigma\}$

   (ii) *if σ is not an assumption, then*
   $\mathcal{O}_{i+1} = \mathcal{O}_i - \{S\} \cup \big\{S - \{\sigma\} \cup R \,\big|\, \sigma \leftarrow R \in \mathcal{R}$, *and* $R \cap C_i = \{\}\big\}$.

Note that in step 2(i)(c) the choice of counter-attack is based upon the choice of contrary of the selected culprit. This choice is made randomly, but can be customised if necessary.

The definitions of GB- and IB-dispute derivations can be modified in a similar fashion, by considering all contraries (of a given assumption) when extending $\mathcal{O}$ (to find attacks against the assumption) and by choosing one contrary (of a given "culprit" assumption) when extending $\mathcal{P}$ (to counter-attack the assumption).

4 System Description

In this section, we will describe the CaSAPI system, a Prolog implementation for credulous and sceptical argumentation based upon the computation of dispute derivations for grounded beliefs (GB-dispute derivations), admissible beliefs (AB-dispute derivations) and ideal beliefs (IB-dispute derivations) for the generalised assumption-based frameworks described in the previous section. The latest version of CaSAPI can be downloaded from `www.doc.ic.ac.uk/~dg00/casapi.html`. The system is developed in Sicstus Prolog but runs on most variants of Prolog[4].

4.1 How to use CaSAPI

After invoking a Prolog process and loading the CaSAPI program, users need to load the input assumption-based framework[5] and the beliefs to be proved. These are best specified in a separate file, prior to invoking Prolog.

Rules in $\mathcal{R}$ are represented as facts of a binary relation `myRule/2` consisting of a left- and right-hand side. The first argument holds the head of the rule and the second argument a list containing the body of the rule. Assumptions in A and beliefs to be proved are represented as unary predicates `myAsm/1` and `toBeProved/1` (respectively) using a list notation for their respective argument. The latter predicate allows queries about more than one belief to be expressed. The notion of contrary can also be customised using an binary relation `contrary/2`. In order to illustrate the representation of assumption-based frameworks, Example 1 from Section 2 is represented as follows:

```
myRule(p,[a]).
myRule(not(a),[b]).
myRule(not(b),[a]).

myAsm([a,b]).

toBeProved([p]).
```

[4] CaSAPI has been successfully tested using SWI Prolog, for example.
[5] The language $\mathcal{L}$ does not need to be specified explicitly.

```
contrary(a,not(a)).
contrary(b,not(b)).
```

The users can then control the kind of dispute derivation they want to employ (GB, AB or IB), the amount of output to the screen (silent, compact or noisy) and the number of supports computed (one or all). They specify their choices as arguments to the command `run/3` and CaSAPI will begin the argumentation process in a manner dictated by the users' choices. For example, in order to run AB-dispute derivations in silent mode and asking for only one answer, one needs to specify: `run(ab,s,1)`. Furthermore, for running GB-dispute derivations in noisy mode asking for all answers, one needs to execute: `run(gb,n,a)`. Note that *all answers* here refers to all answers that can be computed using the dispute derivation in question.

4.2 Design Choices

We have picked Sicstus Prolog as the implementation language of choice since we intend to employ some of its constraint solving features in future versions of CaSAPI. In the current version 2.0 we do not make use of any Sicstus specific code and hence it should run on most standard Prolog engines

One of the interesting properties of Prolog is its handling of variables. Instantiation takes place when a binding can be made, but backtracking allows new instantiations to override old ones where possible. We made use of this feature in that we allow variables in the definition of rules, assumptions and contraries in CaSAPI. This can be seen as a shortcut to writing out all the ground instances of the predicate in question.

A further design choice, that paves the way to interesting experimental research, is the fact that the selection strategies of the agent are not hard-wired into CaSAPI. Different selection strategies do not affect the result of the argumentation process, but have a significant impact on efficiency. Indeed, these strategies control how the dispute trees are generated and hence can lead to early pruning for certain trees. One simple example for a selection strategy is:

```
selFunc([HeadProponent|_],_,HeadProponent,[]).
selFunc([],[[OppHead|OppTail]|_],OppHead,[OppHead|OppTail]).
selFunc([],[],_,_).    % Finished.
```

The first two arguments are the beliefs held by the proponent and opponent, respectively. The third argument is used to return the chosen element and the fourth one returns the set of beliefs of the opponent that this element was chosen from – if applicable. If both the proponent and the opponent have no further beliefs to investigate, the argumentation process terminates. In this simple example, all beliefs of the proponent are handled before the opponent gets to reply. More sophisticated selection strategies can easily be imagined and have been used as defaults in the CaSAPI system.

As we have hinted to before, CaSAPI allows queries to involve sets of beliefs to be proved, rather than individual beliefs as in the original formulation of dispute derivations. But the biggest innovation is the extension of the argumentation framework to allow multiple contraries. The theoretical aspects of this extension have been discussed in the previous section. An example where this extension is employed will be given in Section 5.3 on practical reasoning.

4.3 Worked Example

We illustrate an exemplary execution trace of the CaSAPI system in the case of Example 2 from Section 2. Here and in the remainder of the paper, we represent negative literals $\neg p$ as `not(p)`. In this example, basically `a`, `b` and `c`, `d` are (pairwise) mutually exclusive. Intending to prove the belief `not(a)`, after feeding the following input program:

```
myRule(not(a),[a]).
myRule(not(a),[b]).
myRule(not(b),[a]).
myRule(not(c),[d]).
myRule(not(d),[c]).

myAsm([a,b,c,d]).

toBeProved([not(a)]).

contrary(X,not(X)) :- myAsm(L), member(X,L).
```

into CaSAPI, one needs to choose the execution options. Deciding to use admissible belief semantics, demanding verbose output and requesting only one solution, the following will happen: `not(a)` can only be proved by either the first or second of the rules given above.

```
Step 0:
- Content of this quadruple:
- PropNods: [not(a)]
- OppoNods: []
- DfnceAss: []
- Culprits: []

CASE 1ii
Step 1:
- Content of this quadruple:
- PropNods: [a]
- OppoNods: []
- DfnceAss: [a]
- Culprits: []
```

```
CASE 1i
Step 2:
- Content of this quadruple:
- PropNods: []
- OppoNods: [[not(a)]]
- DfnceAss: [a]
- Culprits: []
```

After some backtracking, the output ends with the final answer:

```
Step 5:
- Content of this quadruple:
- PropNods: []
- OppoNods: []
- DfnceAss: [b,b]
- Culprits: [a]

FINISHED, the defence set is: [b,b]
Without duplicates it is: [b]

yes
```

The defence set `[b]` indicates which assumption(s) need to be made and are sufficient to defend the belief `not(a)` against all possible attacks.

5 Applications

In this section we give examples of how assumption-based argumentation in general and CaSAPI in particular can be applied. First we consider logic programming as an instance of non-monotonic reasoning, and then look at legal, practical and agent reasoning. Note that non-monotonic reasoning using default logic could also be modelled, following [3].

5.1 Non-monotonic reasoning: Logic programming.

A logic program P can be seen as an assumption-based framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where $\mathcal{R}{=}P$, $\mathcal{L}$ is the Herbrand base of P together with the set of all negations of atoms in such Herbrand base, $\mathcal{A}$ is the set of all negative literals in $\mathcal{L}$, and $\overline{not\, p} = p$ for all negative literals $not\, p$ in $\mathcal{A}$. Logic programming queries correspond to sets (conjunctions) of beliefs for which we want to compute dispute derivations.

In this instance of assumption-based frameworks, the admissible, grounded, and ideal semantics correspond (see [3] and [9]) to partial stable models [17], well-founded model [13], and ideal semantics [1], respectively. Although the theoretical framework is propositional, our Prolog implementation allows us to deal with variables, both in the rules of the deductive system and in the beliefs to be proved, as shown below.

Example 3. $P = \{p(X) \leftarrow not\, p(X);$
$p(X) \leftarrow not\, q(X);$
$q(X) \leftarrow not\, p(X);$
$r(X) \leftarrow not\, t(X);$
$t(X) \leftarrow not\, r(X)\}$

Given the logic program from Example 3 and queries $Q_1 = p(a)$, $Q_2 = q(a)$, $Q_3 = r(a)$, $Q_4 = t(a)$, the system computes the following answers, respectively:
GB-dispute derivations: loops, `no`, loops, loops
AB-dispute derivations: $\{not\, q(a)\}$, `no`, $\{not\, t(a)\}$, $\{not\, r(a)\}$
IB-dispute derivations: $\{not\, q(a)\}$, `no`, `no`, `no`

Example 4. $P = \{p \leftarrow not\, q;$
$q \leftarrow not\, r;$
$r \leftarrow not\, s;$
$s \leftarrow not\, q\}$

Given the logic program from Example 4 — with an odd-loop via negation — and query $Q_1 = p$, the system computes `no` for all three kinds of derivations.

5.2 Legal reasoning.

This kind of reasoning often requires dealing with defeasible rules and facts (possibly under dispute), strict rules and facts (beyond dispute) and preferences amongst defeasible rules and facts (possibly under dispute). We show here how a concrete example of legal reasoning from [15] can be dealt with by means of our CaSAPI system, following the formalisation of the problem given in [14].

Example 5. Consider the following set of defeasible rules, including rules defining preferences between rules[6]:

$r_1(X)$: *X 's exterior may not be modified if X is a protected building.*
$r_2(X)$: *X 's exterior may be modified if X needs restructuring.*
$r_3(X, Y)$: $R_1(X) > R_2(Y)$ *if* $R_1(X)$ *concerns artistic buildings and* $R_2(Y)$ *concerns town planning.*
$t(X, Y)$: $R_1(X) > R_2(Y)$ *if* $R_1(X)$ *is later than* $R_2(Y)$.

and the following six facts/strict rules:

$r_1(X)$ *concerns artistic buildings.*
$r_2(X)$ *concerns town planning.*
$r_2(X)$ *is later than* $r_1(X)$.
$r_3(X, Y)$ *is later than* $t(X, Y)$.
villa is a protected building.

[6] For all kinds of rules, we adopt a representation in pseudo-natural language, with variables implicitly universally quantified with scope the rules.

villa needs restructuring.

Intuitively, the conclusion that the exterior of the villa may not be modified should be drawn. Both rules r_1 and r_2 apply and the *meta-rules* r_3 and t deciding the priorities between r_1 and r_2 also apply both, but according to meta-rule t, the importance of r_3 is higher than its own importance. Hence, r_3 should be applied which gives r_1 priority over r_2. Following [14], this problem can be represented as a logic program (and thus as an assumption-based framework, as explained above):

$$
\begin{array}{l}
\textit{villa's exterior may not be modified} \leftarrow \textit{not } defeated(r_1(villa)) \\
\textit{villa's exterior may be modified} \leftarrow \textit{not } defeated(r_2(villa)) \\
defeated(r_1(villa)) \leftarrow \textit{not } defeated(t(villa, villa)),\ \textit{not } defeated(r_2(villa)) \\
defeated(r_2(villa)) \leftarrow \textit{not } defeated(r_3(villa, villa)), \textit{not } defeated(r_1(villa)) \\
defeated(t(villa, villa)) \leftarrow \textit{not } defeated(t((villa, villa), (villa, villa))), \\
\qquad\qquad\qquad\qquad\qquad \textit{not } defeated(r_3(villa, villa))
\end{array}
$$

GB-, AB- and IB-dispute derivations for the belief *villa's exterior not modified* all give the following defence set as an answer: $\{\textit{not } defeated(r_1(villa)), \textit{not } defeated(r_3(villa, villa)), \textit{not } defeated(t((villa, villa), (villa, villa)))\}$.

This can be understood as follows: the villa's exterior should not be modified since rule r_1 is not defeated (stating that artistic buildings should not be modified) and rule r_3 is not defeated (stating that rules concerning artistic buildings override rules concerning town planning) and the temporal ordering rule t is not defeated either.

5.3 Practical reasoning.

This form of reasoning requires making decisions in order to achieve certain properties/objectives, having only partial information. We show how to deal with the concrete example in [2], requiring multiple contraries.

Example 6. A judge needs to decide how best to punish a criminal found guilty, while deterring the general public, rehabilitating the offender, and protecting society from further crime. The judge can choose amongst three forms of punishment: (i) imprisonment, (ii) a fine, or (iii) community service. The judge believes that: (i) promotes deterrence and protection to society, but it demotes rehabilitation; (ii) promotes deterrence but has no effect on rehabilitation and protection of society; (iii) promotes rehabilitation but demotes deterrence.

We can represent the problem as a generalised assumption-based framework:

- $\mathcal{A} = \{prison, fine, service, \alpha, \beta, \gamma, \delta\}$,
- $\mathcal{C}on(prison) = \{fine, service\}$, $\mathcal{C}on(fine) = \{prison, service\}$, $\mathcal{C}on(service) = \{prison, fine\}$, $\mathcal{C}on(\alpha) = \{\neg deter\}$, $\mathcal{C}on(\beta) = \{deter\}$, $\mathcal{C}on(\gamma) = \{\neg rehabilitate\}$, $\mathcal{C}on(\delta) = \{rehabilitate\}$,

– $\mathcal{R}$ consists of nine rules:

$punish \leftarrow prison$	$deter \leftarrow prison, \alpha$	$rehabilitate \leftarrow service, \gamma$
$punish \leftarrow fine$	$deter \leftarrow fine, \alpha$	$\neg rehabilitate \leftarrow prison, \delta$
$punish \leftarrow service$	$\neg deter \leftarrow service, \beta$	$protect \leftarrow prison$

Then, given the goal (belief) *punish*, AB dispute derivations compute the defence set $\{prison\}$, for example. Given also goal *rehab* the defence set $\{service\}$ is computed. One cannot have all goals *punish*, *deter*, *rehabilitate* and *protect* provable (AB dispute derivations return `no`) and it would be interesting to give preferences amongst these, as suggested in [2]. We leave this for future research.

5.4 Agent reasoning

Finally, we will give an example involving a traditional BDI agent [16] that reasons about its beliefs, desires and intentions. We chose the ballroom scenario from [11] and the following setup: picture a traditional ballroom with several male and female dancers; the rules of etiquette state among other things that two dancers agreeing to dance together should be of opposite sex and that female dancers should wait to be approached by a male dancer (with the exception of *ladies' choice night*).

Imagine a female dancer called *anna*, who considers both *bob* and *charlie* to be pretty and who generally intends whatever she desires. This information can be expressed with the following rules in an assumption-based argumentation framework[7]:

$$intend(X) \leftarrow desire(X)$$
$$desire(danceWith(X)) \leftarrow belief(pretty(X)),\ \beta(X)$$
$$\neg desire(danceWith(X)) \leftarrow belief(sameSex(self, X)),\ \alpha(X)$$
$$intend(danceWith(X)) \leftarrow belief(approachedBy(X))$$

$$belief(pretty(bob))$$
$$belief(pretty(charlie))$$

Note that the first rule is domain-independent, whereas the other rules are domain-dependent.

Let $\mathcal{A}$ be the set of (all ground instances of) $\neg belief(X)$ together with $\alpha(X)$ and $\beta(X)$. The latter two assumptions are needed to relate $desire(danceWith(X))$ and $\neg desire(danceWith(X))$ as opposite notions. One cannot directly make them contraries of one another, since neither of them is an assumption.

The $\mathcal{C}on$ relation defines the contrary of any $\neg belief(X)$ as $belief(X)$ and the contrary of $\alpha(X)$ as $desire(danceWith(X))$ and finally, the contrary of $\beta(X)$ as $\neg desire(danceWith(X))$.

Then, asking CaSAPI whether *anna* should intend to dance with *charlie*, the system returns that **y**es, she should intend to dance with that person, provided

[7] We ignore here nested beliefs, desires and intentions for simplicity's sake.

that *anna* believes that *charlie* is not of the same gender. Thus CaSAPI replies with the following defence set: `beta(charlie)`. This is the assumption needed to defend the belief in question and it ensures that $\neg desire(danceWith(charlie))$ does not hold.

The reasoning goes roughly as follows: using the fourth rule, *anna* should intend to dance with *charlie* if she beliefs to have been approached by *charlie*. However, this is not the case. Using the first rule, *anna* should intend to dance with *charlie* if she desires it. She does desire it, since she believes *charlie* is pretty. Now, the fictional opponent who plays devil's advocate in *anna*'s mind may argue that she should not desire (and hence not intend) to dance with *charlie* because *charlie* may be female, too. Therefore, the fictional proponent who defends the query needs to make the additional assumption that *anna* and *charlie* are of opposite gender in order to render the third rule inapplicable.

Note that this is just one simple example of agent reasoning, and more complex and sophisticated forms of reasoning may be afforded by CaSAPI. For example, in case conflicts may arise, *e.g.* due to intending and not intending the same action, the use of preferences, as modelled in legal reasoning, can provide an effective means of conflict resolution. We leave this for future work.

6 Conclusions

In this paper, we have presented a generalisation of computational mechanisms for assumption-based argumentation that allows multiple contraries of assumptions to be expressed. This generalisation enables this kind of argumentation to handle a broader class of applications. Furthermore, we have described the CaSAPI system which implements credulous and (two forms of) sceptical argumentation for this generalisation of assumption-based argumentation and shown how to use the system in some application areas.

Two of these application areas (legal and practical reasoning) assumed a translation (by-hand) from a given formalism into assumption-based argumentation [18]. Future work includes providing appropriate front-ends to our system in order to automate this translation.

We have implemented a number of extensions to theoretical assumption-based argumentation (*e.g.* variables in rules) that would also be worthwhile to formalise in the future.

A number of other argumentation systems exist, for example GORGIAS [6], for credulous argumentation in argumentation frameworks with preferences amongst defeasible rules, the ASPIC system (`http://aspic.acl.icnet.uk/`) [5] dealing with quantitative uncertainty, DeLP [12] for defeasible logic programming, and the system by Krause et al. [4]. These systems are defined for different frameworks for argumentation than ours. It would be interesting to provide a mapping from these various frameworks onto assumption-based argumentation (possibly extended) in order to carry out a full comparison.

Acknowledgements

This research was partially funded by the EC-funded ARGUGRID project. The second author has also been supported by a UK Royal Academy of Engineering/Leverhulme Trust senior fellowship.

References

1. José Júlio Alferes, Phan Minh Dung, and Luís Moniz Pereira. Scenario semantics of extended logic programs. In *LPNMR*, 1993.
2. T. Bench-Capon and H. Prakken. Justifying actions by accruing arguments. In *COMMA*, 2006.
3. A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic framework for default reasoning. *Artificial Intelligence*, 93(1-2), 1997.
4. D. Bryant and P. Krause. An implementation of a lightweight argumentation engine for agent applications. In *JELIA*, 2006.
5. M. Caminada, S. Doutre, S. Modgil, H. Prakken, and G.A.W. Vreeswijk. Implementations of argument-based inference. In *Review of Argumentation Tech.*, 2004.
6. N. Demetriou and A. C. Kakas. Argumentation with abduction. In *Proceedings of the fourth Panhellenic Symposium on Logic*, 2003.
7. Y. Dimopoulos, B. Nebel, and F. Toni. On the computational complexity of assumption-based argumentation for default reasoning. *Artificial Intelligence*, 141, 2002.
8. P.M. Dung, R.A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170, 2006.
9. P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. Technical report, Imperial College London, 2006.
10. P.M. Dung, P. Mancarella, and F. Toni. A dialectic procedure for sceptical, assumption-based argumentation. In *COMMA*, 2006.
11. Dorian Gaertner, Keith Clark, and Marek Sergot. Ballroom etiquette: a case study for norm-governed multi-agent systems. In *1st International Workshop on Coordination, Organisation, Institutions and Norms*, 2006.
12. A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4(1-2), 2004.
13. A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3), 1991.
14. R. A. Kowalski and F. Toni. Abstract argumentation. *Journal of AI and Law, Special Issue on Logical Models of Argumentation*, 4(3-4), 1996.
15. H. Prakken and G. Sartor. On the relation between legal language and legal argument: assumptions, applicability and dynamic priorities. In *ICAIL*, 1995.
16. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
17. Domenico Saccà and Carlo Zaniolo. Partial models and three-valued models in logic programs with negation. In *LPNMR*, 1991.
18. F. Toni. Assumption-based argumentation for epistemic and practical reasoning. Technical report, Imperial College London, 2007.

Reductio ad Absurdum Argumentation in Normal Logic Programs

Luís Moniz Pereira and Alexandre Miguel Pinto
{lmp|amp}@di.fct.unl.pt

Centro de Inteligência Artificial (CENTRIA)
Universidade Nova de Lisboa
Quinta da Torre
2829-516 Caparica, Portugal

Abstract. This paper introduces a new method for defining the argumentative semantics of Normal Logic Programs. In doing so, our single and unified approach allows one to obtain the Stable Models [11] as a special case, or the more general Revision Complete Scenarios here defined.
Normal Logic Programs are approached as assumption-based argumentation systems. We generalize this setting by allowing both negative and positive assumptions. Negative assumptions are made maximal, consistent with existence of a semantics, and positive assumptions are adopted only insofar as they guarantee such existence. Our argumentation semantics thus extends the classical one of [7], and guarantees existence of semantics for any Normal Logic Program, whilst providing all the scenarios corresponding to Stable Models semantics.
Additionally, we provide equivalent and correct algorithms for incrementally computing our scenarios, with three variants. One starts by assuming all atoms as positive assumptions; another assumes them all negative; a third rests on a combination of the first two, and may start with any choice of assumptions. The latter may be employed to address the problem of finding those complete scenarios most compatible with an initial collection of complete scenarios. Consequently, argumentation can be put to collaborative use, not just an antagonistic one. Our results are achieved by generalizing the definitions of the classical approach, which allows only for negative hypotheses, and our definitions fall back on the classical ones when specialized to disallow positive hypotheses.
Finally, integrity constraints are introduced to prune undesired scenarios, whilst permitting these to be produced nevertheless.
Keywords: Argumentation, *Reductio ad Absurdum*, Logic Programs, Argument Revision

1 Introduction

After introducing in [15] and [14] the new Revised Stable Models semantics for Normal Logic Programs further work using the *Reductio ad Absurdum* (RAA) principle has been developed, namely the Revised Well-Founded Semantics [16]. Considering an argument-based view of Logic Programs, we define a new semantics which inherits the RAA principle studied in [15, 14] and apply it to argumentation.

Logic Programs can be viewed as a collection of argumentative statements (rules) based on arguments (default negated literals) [5, 2, 6, 17, 3, 13, 9, 8, 7]. In the quest for finding a Consistent and Complete argumentative scenario one can guess it and check its compliance with these properties; or, innovatively, start with an arbitrary scenario, calculate its consequences, and make revisions to the initial assumptions if necessary in order to achieve 2-valued Completeness and Consistency. This is the road we propose now, revision of assumptions justified by means of *Reductio ad Absurdum* reasoning.

This paper introduces a new method for defining the argumentative semantics of Normal Logic Programs. In doing so, our single and unified approach allows one to get the Stable Models [11] as a special case, or the more general Revision Complete Scenarios here defined.

Normal Logic Programs are approached as assumption-based argumentation systems. We generalize this setting by allowing both negative and positive assumptions. Negative assumptions are made maximal, consistent with existence of a semantics, and positive assumptions are adopted only insofar as they guarantee such existence. The justification of positive assumptions rests on the use of *reductio ad absurdum*, to the effect that replacing any one positive hypothesis (or assumption) by its negative counterpart, in a complete scenario, would result in its inconsistency. Hence, that complete 2-valued scenario must retain its positive assumptions. Our argumentation semantics thus extends the classical one of [7], and guarantees existence of semantics for any Normal Logic Program, whilst providing all the scenarios corresponding to Stable Models semantics.

Additionally, we provide equivalent and correct algorithms for incrementally computing our scenarios, with three variants. One starts by assuming all atoms as positive assumptions; another assumes them all negative; a third rests on a combination of the first two, and may start with any choice of assumptions. The latter may be employed to address the problem of finding those complete scenarios most compatible with an initial collection of complete scenarios. Consequently, argumentation can be put to collaborative use, not just an antagonistic one. Our results are achieved by generalizing the definitions of the classical approach, which allow only for negative hypotheses, and our definitions fall back on the classical ones when specialized to disallow positive hypotheses.

Finally, integrity constraints are introduced to prune undesired scenarios, whilst permitting these to be produced nevertheless.

In essence, our approach caters for the treatment of loops over an odd number of default negated literals, in that it assigns and justifies complete 2-valued models to any Normal Logic Program.

We start by presenting the general Motivation of this paper and, after introducing some needed Background Notation and Definitions, the more detailed Problem Description. We proceed by setting forth our proposal — the Revision Complete Scenarios— and show how it extends previous known results.

Before the Conclusions and Future Work, we show how our approach can enable Collaborative Argumentation, complementing the classical Competitive view of Argumentation.

1.1 Motivation

Ever since the beginning of Logic Programming the scientific community has formally define, in several ways, the meaning, the semantics of a Logic Program. Several semantics were defined, some 2-valued, some 3-valued, and even multi-valued semantics. The current standard 2-valued semantics for Normal Logic Programs— the Stable Models Semantics [11] — has been around for almost 20 years now, and it is generally accepted as the *de facto* standard 2-valued semantics for NLPs. This thoroughly studied semantics, however, lacks some important properties among which the guarantee of Existence of a Model for every NLP.

In [14] we defined a 2-valued semantics— the Revised Stable Models— which extends the Stable Models Semantics, guarantees Existence of a Model for every Normal Logic Program, enjoys Relevancy (allowing for top-down query-driven proof-procedures to be built) and Cumulativity (allowing the programmer to take advantage of tabling techniques for speeding up computations).

Aiming to find a general perspective to seamlessly unify the Stable Models Semantics and the Revised Stable Models Semantics we drew our attention to Argumentation as a means to achieve it. This is the main motivation of the work we present in this paper: by taking the Argumentation perspective we intend to show methods of identifying and finding a 2-valued complete Model for any NLP. The approach is unifying in the sense that it allows us to find the Stable Models and also some other Models needed to ensure guarantee of Existence of a Model. In the process we extend the argumentation stance itself with the ability to incorporate positive hypotheses as needed.

Example 1. **An invasion problem** Some political leader thinks that "If Iran will have Weapons of Mass Destruction then we intend to invade Iran", also "If we do not intend to invade then surely they will have Weapons of Mass Destruction".

$$\begin{aligned} intend\_we\_to\_invade &\leftarrow iran\_will\_have\_WMD \\ iran\_will\_have\_WMD &\leftarrow not\ intend\_we\_to\_invade \end{aligned}$$

If we assume that "we do not intend to invade Iran" then, according to this program we will conclude that "Iran will have Weapons of Mass Destruction" and "we intend to invade Iran". These conclusions, in particular "we intend to invade Iran", contradict the initial hypothesis "we do not intend to invade Iran". So, reasoning by *Reductio ad Absurdum* in a 2-valued setting, we should "intend to invade Iran" in the first place.

This example gives a hint on how we resolve inconsistent scenarios in the rest of the paper.

Example 2. **A vacation problem** Another example puts together three friends that are discussing where they will spend their next joint vacations. John says "If I cannot go the mountains I'd rather go traveling". Mary says "Well, I want to go to the beach, but if that's not possible then I'd rather go to the mountains". Finally, Michael says "I want to go traveling, and if that's not possible then I want to go to the beach".

We put together the three friends' statements formalized into a Normal Logic Program:

$$travel \leftarrow not\ mountain \quad mountain \leftarrow not\ beach \quad beach \leftarrow not\ travel$$

Now, because the three friends need to save money, they must minimize the number of places they will go to on vacation. So they start by assuming they are going nowhere — the cheapest solution. That is, they assume $\{not\ mountain, not\ beach, not\ travel\}$ as true. According to the program above, with these initial hypotheses the friends will conclude they will go traveling, to the beach and to the mountains; and this contradicts the initial hypotheses. They need to revise some of their initial assumptions. If they revise $not\ mountain$ to $mountain$ they will now conclude $\{mountain, beach\}$ and if we put it together with the new set of hypotheses $\{not\ beach, not\ travel, mountain\}$ we get the resulting set $\{mountain, beach, not\ beach, not\ travel\}$. We still have a contradiction on $beach$ and $not\ beach$, which we can easily remove by transforming the hypotheses set into $\{mountain, beach, not\ travel\}$.

There are two more alternative solutions — $\{beach, travel, not\ mountain\}$ and $\{travel, mountain, not\ beach\}$ — which are symmetric to this one.

Example 3. **A time-out problem** John likes Mary a lot so he asked her out: he said "We could go to the movies". Mary is more of a sports girl, so she replies "Either that, or we could go to the swimming pool". "Now, that's an interesting idea", John thought. The problem is that John cannot swim because he hasn't started learning to. He now thinks "Well, if I'm going to the swimming pool with Mary, and I haven't learned how to swim, I'm might risk drowning! And if I'm risking drowning then I really should want to start learning to swim".

Here is the Normal Logic Program corresponding to these sentences:

$$\begin{array}{ll} start\_learning\_to\_swim & \leftarrow risk\_drowning \\ risk\_drowning & \leftarrow go\_to\_pool, not\ start\_learning\_to\_swim \\ go\_to\_pool & \leftarrow not\ go\_to\_movies \\ go\_to\_movies & \leftarrow not\ go\_to\_pool \end{array}$$

If John is not willing to go to the swimming pool — assuming $not\ go\_to\_pool$ — he just concludes go_to_movies and maybe he can convince Mary to join him.

On the other hand, if the possibility of having a nice swim with Mary is more tempting, John assumes he is not going to the movies $not\ go\_to\_movies$ and therefore he concludes go_to_pool. In this case, since John does not know how to swim he could also assume $not\ start\_learning\_to\_swim$. But since John is going to the swimming pool, he concludes $risk\_drowning$. And because of $risk\_drowning$ he also concludes $start\_learning\_to\_swim$. That is, he must give up the hypothesis of $not\ start\_learning\_to\_swim$ in favor of $start\_learning\_to\_swim$ because he wants to go to the swimming pool with Mary. As a nice side-effect he no longer risks drowning.

Example 4. **Middle Region Politics** In a Middle Region two factions are at odds. One believes that if terrorism does not stop then oppression will do it and hence become

unnecessary.

$$oppression \leftarrow not\ end\_of\_terrorism \quad end\_of\_terrorism \leftarrow oppression$$

The other faction believes that if oppression does not stop then terrorism will do it and hence become unnecessary.

$$terrorism \leftarrow not\ end\_of\_oppression \quad end\_of\_oppression \leftarrow terrorism$$

According to these rules, if we assume the $not\ end\_of\_terrorism$ we conclude that there is $oppression$ which in turn will cause the $end\_of\_terrorism$. So, the $end\_of\_terrorism$ should be true in the first place, instead of $not\ end\_of\_terrorism$. The same happens with $end\_of\_oppression$. In spite of the peaceful resulting scenario we propose, $\{end\_of\_oppression, end\_of\_terrorism\}$, there is no Stable Model for this program.

1.2 Background Notation and Definitions

Definition 1. ***Logic Rule*** *A Logic Rule r has the general form $L \leftarrow b_1, b_2, \ldots, b_n, not\ c_1, not\ c_2, \ldots, not\ c_m$ where L is a literal, i.e., an atom h or its default negation $not\ h$, and $n, m \geq 0$.*

We call L the head of the rule — also denoted by $head(r)$. And $body(r)$ denotes the set $\{b_1, b_2, \ldots, b_n, not\ c_1, not\ c_2, \ldots, not\ c_m\}$ of all the literals in the body of r. Throughout this paper we will use 'not ' to denote the default negation.

When the body of the rule is empty, we say the head of rule is a fact and we write the rule as just h or $not\ h$. □

Definition 2. ***Logic Program*** *A Logic Program (LP for short) P is a (possibly infinite) set of ground Logic Rules of the form presented in definition 1. If the heads of all the rules in P are positive literals, i.e., they are simple atoms, and not default negated literal, we say we have a Normal Logic Program (NLP). If at least one of the heads of a rule of P is a default negated literal, and there is no explicit negation in the program — we say we have a Generalized Logic Program (GLP). If there is explicit negation, besides default negation, in the program we say we have an Extended Logic Program (ELP).* □

Definition 3. ***Atoms of a Logic Program*** *P — $Atoms(P)$ $Atoms(P)$ denotes the set of all atoms of P. Formally,*

$Atoms(P) = \{a : \exists_{r \in P}\big(head(r) = a \vee head(r) = not\ a \vee a \in body(r) \vee not\ a \in body(r)\big)\}$ □

Throughout the rest of this paper we will focus solely on Normal Logic Programs hence, when we write just a Program or a Logic Program we mean a Normal Logic Program.

Definition 4. ***Default negation of a set S of literals —*** *$not\ S$ Throughout this paper we will sometimes use the $not\ S$ default negation of a set S notation, where S is a set of literals, in order to denote the set resulting from default negating every literal of S. Formally, $not\ S = \{not\ a : a \in S\} \cup \{b : not\ b \in S\}$* □

Definition 5. ***Scenario*** *A scenario of a NLP P is the Horn theory $P \cup H$, where $H = H^+ \cup H^-$, $H^+ \subseteq Atoms(P)$, $H^- \subseteq not\ Atoms(P)$, and $not\ H^+$ and H^- are disjoint. H is called a set of hypotheses, positive and negative.* □

Definition 6. $\vdash$ ***operator*** *Let P be a NLP and H a set of hypotheses. P' is the Horn theory obtained from P by replacing every default literal of the form $not\ L$ in P by the atom not_L. H' is likewise obtained from H using the same replacement rule. By definition, $P' \cup H'$ is a Horn theory, and so it has a least model M. We define $\vdash$ in the following way, where A is any atom of P:*

$P \cup H \vdash A$ *iff* $A \in M$ $\qquad$ $P \cup H \vdash not\ A$ *iff* $not\_A \in M$ □

Definition 7. ***Consistent scenario*** *A scenario $P \cup H$ is consistent iff for all literals L, if $P \cup H \vdash L$ then $P \cup \vdash H \nvdash not\ L$, where $not\ not\ L \equiv L$.* □

Definition 8. ***Consistent program*** *A Logic Program P is consistent iff $P \cup \emptyset$ is a consistent scenario. NLPs are of course consistent.* □

2 Revision Complete Scenarios

In [4] the author proves that every Stable Model (SM) of a NLP is a 2-valued complete (total), consistent, admissible scenario. The author considers a scenario as a set of default negated literals — the hypotheses. However, not every NLP has a consistent, 2-valued complete scenario when one considers as hypotheses just default negated literals.

Also in [4], the author shows that preferred maximal (with maximum default negated literals) scenarios are always guaranteed to exist for NLPs. However, preferred maximal scenarios are, in general, 3-valued.

The problem we address now is to find a way to render 2-valued total a preferred maximal scenario. In this paper we take a step further from what was previously achieved in [4], extending its results. We allow a set of hypotheses to contain also positive literals, but only those absolutely necessary to guarantee Existence of a Model. These positive hypotheses are those who are justified *true* by a specific *Reductio ad Absurdum* reasoning we accept.

Before presenting the formal Definition of a Revision Complete Scenario we give a general intuitive idea to help the reader grasp the concept. For the formal definition of Revision Complete Scenario we will also need some preliminary auxiliary definitions.

2.1 Intuition

In [3] the authors prove that every SM of a NLP corresponds to a stable set of hypotheses which correspond in turn to a 2-valued complete, consistent, admissible scenario.

In order to guarantee the Existence of a 2-valued total Model for every NLP we allow positive hypotheses to be considered besides the usual negative hypotheses. Under this setting, the easiest way to solve the problem would be to accept every atom of a program as a positive hypotheses. However, we want to our semantics to be the most skeptical possible while ensuring stratification compatibility among hypotheses.

To further keep the semantics skeptical we want to have the maximal possible negative hypotheses and the minimum non-redundant positive hypotheses. Intuitively, a positive hypothesis L is considered redundant if, by the rules of the program and the rest of the hypotheses, L is already determined *true*. The formal definition of this notion of non-redundancy of positive hypotheses is presented and explained below.

The formal notion of compatibility will also be depicted and explained below, but for now the intuitive idea is that one positive hypothesis L must not contradict other hypotheses.

2.2 Definition

Definition 9. ***Evidence for a literal*** *L A negative set of hypotheses $E \subseteq not\ Atoms(P)$ is evidence for a literal L in program P iff $P \cup E \vdash L$. If P is understood we write $E \rightsquigarrow L$. We also say E attacks $not\ L$. Notice that we do not require an evidence to be consistent.* □

Definition 10. ***Weakly Admissible set of hypotheses*** H^-

The notion of weakly admissible set presented here is in line with that of weak stability, first defined in [12].

Let P be a NLP, $H^- \subseteq not\ Atoms(P)$ a set of negative hypotheses, $not\ L$ a default negated literal in P and E an evidence for L. We say H^- is weakly admissible iff $\forall_{not\ L \in H^-} \forall_{E \rightsquigarrow L} \exists_{not\ A \in E} P \cup H^- \cup E \vdash A$ □

The classical notion of admissible set checks only if $P \cup H^- \vdash A$. By doing this test with $P \cup H^- \cup E$ we allow E to be inconsistent. It suffices to see that if $P \cup H^- \nvdash A$ and $P \cup H^- \cup E \vdash A$ it means that E is essential to derive A in the $P \cup H^-$ context. Since we know $not\ A \in E$ and $P \cup H^- \cup E \vdash A$ we conclude that E is inconsistent.

There are some sets of hypotheses H^- which were not admissible according to the classical definition (with just $P \cup H^-$) and are weakly admissible — according to the definition using $P \cup H^- \cup E$. These sets of hypotheses which are accepted as weakly admissible are just the ones where the adding of the evidence E was essential to derive A, that is, where E is inconsistent.

Since the $\vdash$ operator is monotonic, every admissible set of hypotheses according to the classical definition (using $P \cup H^-$) is also weakly admissible — according to the definition with $P \cup H^- \cup E$.

Example 5. **Weakly Admissible vs Non Weakly Admissible sets of negative hypotheses** Consider the following NLP:

$$k \leftarrow not\ t \quad t \leftarrow a, b \quad a \leftarrow not\ b \quad b \leftarrow not\ a$$

In this program we can easily see that the bottom Even Loop Over Negation (ELON, for short) over a and b allows only one of them to be true — when we demand minimality of positive information. Under this setting we will never have t true for it needs both a and b to be true simultaneously to support its truthfulness. Therefore, k will always be true, since t is always false.

Let us analyze the different possible sets of hypotheses from an admissibility point of view. Consider the following two sets of negative hypotheses $H_1 = \{not\ b, not\ t\}$ and $H_2 = \{not\ b, not\ k\}$. The other two sets of negative hypotheses H_3 and H_4 are just symmetric to H_1 and H_2, respectively, on $not\ a$ and $not\ b$; therefore we are going to focus solely on H_1 and H_2.

H_1 is weakly admissible whereas H_2 is not. Let us see why. Analyzing $not\ b$ we verify that there is only one possible evidence $E = \{not\ a\}$ for b and that $P \cup H_1 \cup E \vdash a$, i.e., $H_1 \cup E$ attacks (in the sense presented in definition 9) $not\ a$. In this particular case even just H_1 attacks $not\ a$.

Analyzing $not\ t$ we can see that there is only one evidence $E = \{not\ a, not\ b\}$ for t. $P \cup H_1 \cup E$ derives both a and b, i.e., $P \cup H_1 \cup E \vdash a$ and $P \cup H_1 \cup E \vdash b$; hence H_1 is weakly admissible.

Let us see what happens with H_2. We have already seen $not\ b$, we just need to test $not\ k$. The only evidence for k is $E = \{not\ t\}$. We can see however that $P \cup H_2 \cup E \nvdash t$, which leads us to conclude that H_2 is not weakly admissible.

Example 6. **Allowing Inconsistent Evidence** Consider the following NLP:

$$k \leftarrow not\ t \quad t \leftarrow not\ t$$

The hypotheses $H_1 = \{not\ t\}$ is admissible and weakly admissible. However, since $P \cup H_1$ is not a consistent scenario, no model exists with $not\ t$.

The only possible hypotheses left are the empty set and $H_2 = \{not\ k\}$. Considering the classical notion of admissible set (with $P \cup H^-$) H_2 is non-admissible; however, H_2 is weakly admissible. Notice that the evidence for k is $E = \{not\ t\}$ and that $P \cup H_2 \cup E \vdash t$. $P \cup H_2$ is a consistent scenario, but it is not complete. Since we already know that $not\ t$ cannot be in any consistent model, in a 2-valued setting we would like to "complete" the scenario $P \cup H_2$ with t in order to obtain a 2-valued complete and consistent model. In such case we say $\{t\}$ is our set of positive hypotheses.

Definition 11. ***Non-redundant set H^+ of positive hypotheses*** *Let P be a NLP, and $H = H^+ \cup H^-$ a set of positive and negative hypotheses, i.e., ($H^+ \subseteq Atoms(P)$) and ($H^- \subseteq not\ Atoms(P)$). We say H^+ is non-redundant iff* $\forall_{L \in H^+} P \cup H \setminus \{L\} \nvdash L$ □

As just explained, we wish to allow some positive hypotheses when they are absolutely needed in order to obtain 2-valued complete and consistent scenarios. However, we require the positive set of hypotheses to be non-redundant, that is, all positive hypotheses must not be already derived by other hypotheses. This is the purpose of definition 11 above.

Example 7. **Redundant positive hypotheses** Consider the following program P:

$$b \leftarrow a \quad a \leftarrow not\ a$$

In the previous example 6 we saw how a rule like $t \leftarrow not\ t$ forbids the negative hypothesis $not\ t$. By the same token, in this example's program, the hypothesis $not\ a$ is also forbidden. Also $\{not\ b\}$ is not a weakly admissible set of negative hypotheses.

Since we are looking for 2-valued complete (total) and consistent scenarios, we would like one including both a and b.

The question now is: should both a and b be considered positive hypotheses? Since we are looking for the minimum possible set of positive hypotheses (compatible with the negative ones), we answer *no* in this case, because assuming the positive hypothesis a is enough to automatically determine the truth of b. That is why we say the set $\{a, b\}$ of positive hypotheses is redundant, whereas $\{a\}$ is not.

Definition 12. ***Unavoidable set*** H^+ ***of positive hypotheses*** *Let P be a NLP, and $H = H^+ \cup H^-$ a set of positive and negative hypotheses. We say H^+ is unavoidable iff $\forall_{L \in H^+} P \cup \big(H \setminus \{L\}\big) \cup \{not\ L\}$ is an inconsistent scenario* □

In a nutshell, this definition imposes that every positive hypothesis must be accepted as true for the sake of consistency and completeness in the context of all other hypotheses. We ensure this by demanding that any if positive hypothesis L was to be considered false — i.e., $not\ L$ considered true — the whole scenario of P with all the hypotheses, except L, and including $not\ L$ instead (for the sake of 2-valued completeness) would be inconsistent. So, there is no consistent 2-valued way to avoid having L true in the context of the remaining hypotheses. Additionally, one may need the condition as stating that, if the scenario with $not\ L$ is consistent, then L is avoidable.

Example 8. **Unavoidable vs Avoidable sets of positive hypotheses** Let P be the following NLP:

$$d \leftarrow not\ c \quad c \leftarrow not\ b \quad b \leftarrow not\ a \quad a \leftarrow not\ a$$

In this example we consider $H_1 = H_1^+ \cup H_1^-$, where $H_1^+ = \{a\}$ and $H_1^- = \{not\ b, not\ d\}$; and $H_2 = H_2^+ \cup H_2^-$, where $H_2^+ = \{a, b\}$ and $H_2^- = \{not\ c\}$.

By the same reason as in example 7 $not\ a$ cannot be in any H^- and, in order to obtain a 2-valued total model with an H, a must be accepted as true — in that sense we say a is unavoidable.

Definition 13. ***Revision Complete Scenarios*** *Let P be a NLP and $H = H^+ \cup H^-$ a set of positive (H^+) and negative (H^-) hypotheses. We say H is a Revision Complete Scenario iff*

1. *$P \cup H$ is a consistent scenario and $least(P \cup H)$ is a 2-valued complete model of P*
2. *H^- is weakly admissible*
3. *H^+ is not redundant*
4. *H^+ is unavoidable*

□

2.3 The Exhaustive Model Generation Algorithm

Another method for finding the Revision Complete Scenarios is an iterative and incremental way.

Definition 14. ***Inconsistency avoidance algorithm for generating the Revision Complete Scenarios (RCSs)***

1. *Start with* $i = 0$, $H_i^+ = Atoms(P)$ *and* $H_i^- = \emptyset$.
2. *If* H_i^- *is not weakly admissible then* $H_i^+ \cup H_i^-$ *is not a Revision Complete Scenario and the algorithm terminates unsuccessfully.*
3. *If* H_i^- *is weakly admissible then:*
4. *If* $H_i^+ = \emptyset$ *then* $H_i^+ \cup H_i^-$ *is a RCS and the algorithm terminates successfully in this case.*
5. *If* $H_i^+ \neq \emptyset$ *then non-deterministically take one arbitrary* $L \in H_i^+$ *and check if* H_i^+ *is redundant on* L. *If it is then:*
6. $H_{i+1}^+ = H_i^+ \setminus \{L\}$ *and go back to step 3 (a).*
7. *If* H_i^+ *is non-redundant then:*
8. *Check if* H_i^+ *is unavoidable and, if so, then* $H_i^+ \cup H_i^-$ *is a RCS and the algorithm terminates successfully.*
9. *If* H_i^+ *is not unavoidable and* $L \in H_i^+$ *is one of the positive hypotheses rendering* H_i^+ *non-unavoidable then* $H_{i+1}^+ = H_i^+ \setminus \{L\}$ *and* $H_{i+1}^- = H_i^- \cup \{not\ L\}$ *and go on to step 2 again.*

□

This algorithm starts with all the possible positive hypotheses (all the atoms of the program) and no negative hypotheses. By construction, a scenario with such H^+ and H^- is necessarily consistent and 2-valued complete. Along the execution of the algorithm, at each time, we either just remove one positive hypothesis because redundant, or non-deterministically remove one positive hypothesis and add its correspondent default negation to the set of negative hypotheses. By construction, the algorithm guarantees that $H = H^+ \cup H^-$ is consistent. When we just remove one positive hypothesis $L \in H^+$ the 2-valued completeness of the resulting scenario is guaranteed because L was removed from H^+ only because L was rendering H^+ redundant. When we remove L from H^+ and add $not\ L$ to H^- 2-valued completeness is naturally assured.

The requirement for weak admissibility of H^- in step 3 ensures the resulting $H = H^+ \cup H^-$ corresponds to a consistent scenario. The different non-deterministic choices engender all the RCSs.

Example 9. **Generating RCSs by Inconsistency avoidance**

$$a \leftarrow not\ a, not\ b \qquad b \leftarrow not\ a, not\ b$$

We start the algorithm with all the possible positive hypotheses and no negative ones:

- $H_0^+ = \{a, b\}, H_0^- = \emptyset$.
- H_0^- is weakly admissible.
- $H_0^+ \neq \emptyset$ so we check if it is redundant. It is not, so we check if H_0^+ is unavoidable.
- H_0^+ is not unavoidable. We non-deterministically choose one atom from $H_0^+ = \{a, b\}$ which makes it non-unavoidable (in this case, both a and b are rendering H_0^+ non-unavoidable, so we can choose any one). Let us say we choose b. Then $H_1^+ = H_0^+ \setminus \{b\}$ and $H_1^- = H_0^- \cup \{not\ b\}$. And we go on to step 2 again.

- H_1^- is weakly admissible.
- $H_1^+ \neq \emptyset$.
- H_1^+ is not redundant on any $L \in H_1^+$.
- H_1^+ is unavoidable and so $H_1 = H_1^+ 1 \cup H_1^- = \{a, not\ b\}$ is a Revision Complete Scenario and the algorithm terminates successfully.

If we were to choose $not\ a$ instead of $not\ b$ in step 9, the resulting Revision Complete Scenario would be $\{not\ a, b\}$. There are no other Revision Complete Scenario for this program besides these two.

Theorem 1. ***The sets*** $H = H^+ \cup H^-$ ***resulting from the execution of algorithm of definition 14 are the Revision Complete Scenarios***

Proof. Trivial, by construction of the algorithm. □

Theorem 2. ***Existence of Model*** *For any given NLP P there is always at least one Revision Complete Scenario.*

Proof. In the algorithm described above, when we need to non-deterministically choose one atom L to remove from H_i^+, and eventually add $not\ L$ to H_i^-, if there are no repetitions in the choice, then the algorithm is necessarily guaranteed to terminate.

Moreover, if the first positive hypothesis to remove correspond to atoms upon which no other atoms depend, then removing that positive hypotheses has causes no inconsistency, nor does it compromise 2-valued completeness. If the next positive hypotheses in the sequence to be removed always guarantee that the consequences of its removal (and eventual adding of its default negated counterpart to the set of negative hypotheses) does not change the truth value of positive hypotheses already removed, then it is necessarily guaranteed that the algorithm will find a Revision Complete Scenario.

Finally, it is always possible to find such a sequence of positive hypotheses to remove: the sequence just needs to be in reverse order of the stratification of the program. I.e., the first positive hypotheses in the sequence must be from the top strata of the program, the second hypotheses from the second strata counting from the top, and so on. The notion of stratification we are unsing here can be intuitively explained as: (1) atoms in a loop are all in the same strata; (2) atoms which are not in a loop, and are in the head of a rule are in a strata which is always one directly above the atoms in the body of the rule. □

Theorem 3. M ***is a Stable Model of a NLP*** P ***iff there is some Revision Complete Scenario*** H ***such that*** $M = least(P \cup H)$ ***with*** $H^+ = \emptyset$

Proof. Let $H = H^+ \cup H^-$ a set of positive and negative hypotheses. Let us consider the particular case where $H^+ = \emptyset$, therefore $H = H^-$.

In [4], the author already proved that when $H = H^-$, $P \cup H$ is a consistent scenario and $M = least(P \cup H)$ is a 2-valued complete scenario iff M is a Stable Model of P.

Stable Models are just a particular case of Revision Complete Scenarios. □

A variation of this algorithm reversing the direction of the changes in H^+ and H^- can also be depicted. In such an algorithm we start with $H^- = not\ Atoms(P)$ and $H^+ = \emptyset$. 2-valued completeness is also assured at the starting point, although consistency of $P \cup H$ is not. The algorithm is:

Definition 15. ***Inconsistency removal algorithm for generating the Revision Complete Scenarios (RCSs)***

1. *Start with* $i = 0$, $H_i^- = not\ Atoms(P)$ *and* $H_i^+ = \emptyset$.
2. *If* $P \cup H_i$ *is a consistent scenario then* H_i *is a RCS and the algorithm terminates successfully.*
3. *Check if* H_i^+ *is redundant:*
4. *If it is redundant then non-deterministically take one arbitrary atom* $L \in H_i^+$ *such that* $P \cup H \setminus \{L\} \vdash L$ *and construct* $H_{i+1}^+ = H_i^+ \setminus \{L\}$.
5. *If* H_i^+ *is non-redundant construct* $H_{i+1}^+ = H_i^+$.
6. *Check if* H_{i+1}^+ *is unavoidable:*
7. *If* H_{i+1}^+ *is non-unavoidable then* $H_{i+1}^+ \cup H_{i+1}^-$ *is not a RCS and the algorithm terminates unsuccessfully.*
8. *If* H_{i+1}^+ *is unavoidable then check if* $P \cup H_{i+1}$ *is a consistent scenario:*
9. *If* $P \cup H_{i+1}$ *is a consistent scenario then:*
10. *Check if* $P \cup H_{i+1}$ *is also a 2-valued complete scenario and if it is then* H_{i+1} *is a RCS and the algorithm terminates successfully.*
11. *If* $P \cup H_{i+1}$ *is not a 2-valued complete scenario then construct* $H_{i+2}^+ = H_{i+1}^+ \cup \{L\}$, *where* $P \cup H_{i+1} \nvdash L$ *and* $P \cup H_{i+1} \nvdash not\ L$, *and* H_{i+2}^+ *is non-redundant. Go on to step 4 again.*
12. *If* $P \cup H_{i+1}$ *is not a consistent scenario, take one* $not\ L \in H_{i+1}^-$ *such that* $P \cup H_{i+1} \vdash L$ *and* $P \cup H_{i+1} \vdash not\ L$ *(i.e., there is a contradiction in* L *with* $P \cup H_{i+1}^-$*) and construct* $H_{i+2}^- = H_{i+1}^- \setminus \{not\ L\}$ *and* $H_{i+2}^+ = H_{i+1}^+ \cup \{L\}$, *i.e., we revise the assumption* $not\ L$ *to* L *making it a positive hypothesis. Go on to step 3 again.*

□

This algorithm starts with all the possible negative hypotheses (the default negation of all the atoms of the program) and no positive hypotheses. By construction, a scenario with such H^+ and H^- is necessarily consistent and 2-valued complete. Along the execution of the algorithm, at each time, we either just remove one positive hypothesis — because it is redundant — , or remove one negative hypothesis $not\ L$ and add its correspondent positive L to the set of positive hypotheses — i.e., we revise the assumption $not\ L$ to L, when the set of negative hypotheses with $not\ L$ is not consistent.

Also by construction the algorithm guarantees that $H = H^+ \cup H^-$ is consistent and, therefore, that H^- is weakly admissible. When we just remove one positive hypothesis $L \in H^+$ the 2-valued completeness of the resulting scenario is guaranteed because L was redundant in H^+. When we remove $not\ L$ from H^- and add L to H^+ 2-valued completeness is naturally assured. The different non-deterministic choices engender all the RCSs.

Example 10. **Generating RCSs by Inconsistency removal** Let us revisit the example 9 and see the Inconsistency removal version of it.

$$a \leftarrow not\ a, not\ b \qquad b \leftarrow not\ a, not\ b$$

We start the algorithm with all the possible negative hypotheses and no positive ones:

- $H_0^- = \{not\ a, not\ b\}, H_0^+ = \emptyset$.
- $P \cup H_0$ is not a consistent scenario.
- $H_0^+ = \emptyset$ is non-redundant.
- $H_1^+ = H_0^+$ is unavoidable.
- $P \cup H_1$ is not a consistent scenario.
- We non-deterministically choose one negative hypothesis $not\ L$ from $H_1^- = \{not\ a, not\ b\}$ such that $P \cup H_1 \vdash L$ and $P \cup H_1 \vdash not\ L$. In this case, both $not\ a$ and $not\ b$, so we can choose any one of them. Let us say we choose $not\ a$. Then $H_2^+ = H_1^+ \cup \{1\}$ and $H_2^- = H_1^- \setminus \{not\ a\}$. And we go on to step 3 again.
- H_2^+ is non-redundant.
- $H_3^+ = H_2^+$ is unavoidable.
- $P \cup H_3$ is a consistent scenario.
- $P \cup H_3$ is a 2-valued complete scenario, so $H_3 = H_3^+ \cup H_3^- = \{a\} \cup \{not\ b\} = \{a, not\ b\}$ is a Revision Complete Scenario and the algorithm terminates successfully.

If we were to choose $not\ b$ instead of $not\ a$ in step 12, the resulting Revision Complete Scenario would be $\{not\ a, b\}$. These Revision Complete Scenario coincide with those produced by the algorithm in definition 14.

Theorem 4. ***The sets*** $H = H^+ \cup H^-$ ***resulting from the execution of algorithm of definition 15 are the Revision Complete Scenarios***

Proof. Trivial, by construction of the algorithm. □

2.4 The Name of the Game

Why the name "Revision" Complete Scenarios? The "Revision" part of the name comes from the assumption revision we do when an assumption $not\ A \in H^-$ leads to a contradiction in P, i.e., $\big(P \cup H^- \vdash \{A, not\ A\}\big) \wedge \big(P \cup (H^- \setminus \{not\ A\}) \nvdash \{A, not\ A\}\big)$.

In such a case we accept to revise $not\ L$ to its positive counterpart L. This is the specific form of reasoning by *Reductio ad Absurdum* we take here: if adding $not\ A$ to P in the context of H^- leads to self inconsistency, then, by absurdity, we should assume A instead of $not\ A$. A becomes, thus, one of the positive hypotheses.

3 Syntactic Perspective of Revision Complete Scenarios over Normal Logic Programs

In [3] the authors proved that every Stable Model of a NLP corresponds to a 2-valued complete, consistent and admissible scenario. In [10] the author shows that when a NLP has no SMs it is because the Normal Logic Program has Odd Loops Over Negation (OLONs) and/or Infinite Chains Over Negation (ICONs), although the author does not employ these designations. These designations are taken from [14].

For the sake of readability and self-containment we briefly present some examples of OLONs and ICONs. Intuitively an OLON is a set of rules of a NLP which induce

a cycle over some literals in the dependency graph. The cycle of an OLON has the characteristic of having an Odd number of default Negated arcs around the cycle.

An example of an OLON is given in example 1. There we can see that the atom $intend\_we\_to\_invade$ is in a cycle across the dependency graph, and that along that cycle there is only 1 (an Odd number) default negation.

Another example of an OLON is present in example 2. There the atom $mountain$ is in a cycle with 3 default negations along the circular dependency graph. The same is true for $travel$ and $beach$.

The classical example of an ICON was first presented in [10]. It goes as follows:

$$p(X) \leftarrow p(s(X)) \qquad p(X) \leftarrow not\ p(s(X))$$

where X is a variable. The ground version of this program when there is only one constant 0 is the infinite program

$$\begin{array}{ll} p(0) \leftarrow p(s(0)) & p(0) \leftarrow not\ p(s(0)) \\ p(s(0)) \leftarrow p(s(s(0))) & p(s(0)) \leftarrow not\ p(s(s(0))) \\ p(s(s(0))) \leftarrow p(s(s(s(0)))) & p(s(s(0))) \leftarrow not\ p(s(s(s(0)))) \\ \vdots & \vdots \end{array}$$

This example in particular is the one to which every other possible variation of an ICON reduces to (proven in [10]). As it can be easily seen, there is an infinitely long chain of support for any $p(X)$ with an infinite number of default negations.

As we just said, in [10] the author proves that only OLONs and/or ICONs can prevent the existence of SMs in a NLP. Therefore, since our Revision Complete Scenario guarantee the Existence of a Model for any given NLP it follows that the Revision Complete Scenario deal with OLONs and ICONs in a way that the Stable Models semantics did not. This is achieved by means of the reasoning by *Reductio ad Absurdum* we explained in subsection 2.4.

4 Collaborative Argumentation

The classical perspective on Argumentation is typically of a competitive nature: there are arguments and counter-arguments, all of them attacking each other and struggling for admissibility. The ones which counter-attack all its attackers are admissible.

Typically, one takes one argument — a set of hypotheses H — and check if it is admissible, and if $P \cup H$ is a consistent scenario. If 2-valuedness is a requisite, then an extra test for 2-valued completeness is required.

We now generalize this approach in a constructive way, by building up a compromise Revision Complete Scenario starting from several conflicting 2-valued complete and consistent Models of P — each corresponding to an argument. This is what the algorithm below does.

First, we take all the conflicting models N_1, N_2, ..., N_n and calculate the set of all the possible positive hypotheses $M^+ = \bigcup_{i=1}^{n} N_i^+$; and the set of all the possible negative hypotheses $M^- = \bigcup_{i=1}^{n} N_i^-$. M^+ and M^- will now be used to guide the algorithm below in order to ensure consensus, i.e., the resulting Revision Complete

Scenario H will have no positive hypotheses outside M^+, nor will it have negative hypotheses outside M^-. The algorithm goes as follows:

Definition 16. ***Revision Complete Scenario*** H ***construction from conflicting models*** $N_1, N_2, \ldots N_n$

1. *Start with* $M = M^+ \cup M^-$. $M_0 = M$ *is inconsistent.*
2. $M_1^+ = M_0^+ \setminus \{L \in M_0^+ : not\ L \in M_0^-\}$, *and* $M_1^- = M_0^-$. M_1 *is now consistent.*
3. *If* M_i^- *is not weakly admissible then non-deterministically select one* L *such that* $not\ L \in M_i^-$, *there is an* E *such that* $E \rightsquigarrow L$, *and there is some* $not\ a \in E$ *such that* $P \cup M_i^- \cup E \nvdash a$. *Construct* $M_{i+1}^- = M_i^- \setminus \{not\ L\}$. *Repeat this step.*
4. *If* M_{i+1}^+ *is avoidable then* $M_{i+2}^+ = M_{i+1}^+ \setminus \{L\}$, *where* $P \cup (M_{i+1} \setminus \{L\}) \cup \{not\ L\}$ *is an inconsistent scenario.* $M_{i+2}^- = M_{i+1}^- \cup \{not\ L\}$ *only if* $L \in M^-$, *otherwise* $M_{i+2}^- = M_{i+1}^-$. *Go on to step 3 again.*
5. *If* $P \cup M_{i+2}$ *is not a consistent scenario then non-deterministically select one* L *such that* $P \cup M_{i+2} \vdash \{L, not\ L\}$, *and construct* $M_{i+3}^- = M_{i+2}^- \setminus \{not\ L\}$. *Go on to step 3 again.*
6. *If* $P \cup M_{i+2}$ *is not a 2-valued complete scenario then* $M_{i+3}^+ = M_{i+2}^+ \cup \{L\}$, *where* $P \cup M_{i+2} \nvdash L$ *and* $P \cup M_{i+2} \nvdash not\ L$ *and* $L \in M^+$, *and go on to step 4 again.*
7. $P \cup M_{i+2}$ *is a 2-valued complete and consistent scenario, where* M_{i+2}^+ *is non-redundant and unavoidable, and* M_{i+2}^- *is weakly admissible. By definition,* M_{i+2} *is a Revision Complete Scenario, therefore* $H = M_{i+2}$ *and the algorithm terminates successfully.*

□

In essence, this algorithm is a mixture of the Inconsistency Avoidance and Inconsistency Removal algorithms presented in subsection 2.3. We start with two sets M^+ and M^- containing, respectively, all the possible positive hypotheses that can be adopted in the final Revision Complete Scenario H, and all the possible negative hypotheses that can be adopted. Next, we remove from the set of positive hypotheses all those conflicting with the negative ones in order to ensure consistency. Now we need to ensure a weak admissibility of the current negative hypotheses M_i^-. For that we check if the M_i^- is weakly admissible, and if it is not, then we non-deterministically select and remove from M_i^- one of the negative hypotheses causing M_i^- failing to comply to this requirement. This step is repeated until weak admissibility is verified by M_i^-. Now we turn to the set of positive hypotheses M_i^+. If it is avoidable, then we non-deterministically select and remove from M_i^+ one positive hypothesis L which contributes to M_i^+ avoidability. We also add the correspondent default negation of that positive hypotheses $not\ L$ to M_i^-, but only if $not\ L$ was already in M^- — the initial set of all the adoptable negative hypotheses. This extra requirement ensures the final compromise Revision Complete Scenario H to be found is maximally compatible with all the initial models $N_1, N_2, \ldots, N_n$. When we add $not\ L$ to M_i^- we need to recheck its weak admissibility, so we go on to that step again. If M_i^+ was unavoidable, then we need to check it the whole $P \cup M_i$ is consistent. If this scenario fails consistency, then we remove from M_i^- one of the negative hypothesis whose positive counterpart was also being produced by $P \cup M_i$. Notice that when the resulting scenario is not consistent we

remove one inconsistency in favour of the positive hypotheses, since the presence of the correspondent negative produced the inconsistency. This is basically the mechanism of reasoning by *Reductio ad Absurdum* we use. Again we need to recheck the weak admissibility, so we go on to that step again. If the scenario $P \cup M_i$ was consistent, then we need to check if it is 2-valued complete. If it is not, then we non-deterministically select one adoptable positive hypothesis and add it to M_i^+. Now we need to recheck M_i^+'s unavoidability; so we go on to that step again. Finally, if $P \cup M_i$ was 2-valued complete then $H = M_i$ is a Revision Complete Scenario and the algorithm terminates successfully.

Example 11. **Example 2 revisited — A vacation problem** Recall the example 2 presented earlier. The program is:

$$travel \leftarrow not\ mountain \quad mountain \leftarrow not\ beach \quad beach \leftarrow not\ travel$$

Now assume that one of the friends going on vacation with the other two could not be present when they were getting together to decide their vacations' destinies. So, only John (the one who preferred going to the mountains, otherwise traveling it is), and Mary (she prefers going to the beach, otherwise going to the mountains is ok).

John's opinion is $J = \{mountain, not\ travel, not\ beach\}$, while Mary's choice is $Z = \{beach, not\ mountain, not\ travel\}$. We can already see that at least on one thing they agree: $not\ travel$. We now find the largest set of positive hypotheses we can consider $M^+ = J^+ \cup Z^+ = \{mountain, beach\}$ and the largest set of negative hypotheses we can consider $M^- = J^- \cup Z^- = \{not\ travel, not\ beach, not\ mountain\}$. And now the algorithm starts:

$M = M^+ \cup M^- = \{mountain, beach, not\ mountain, not\ beach, not\ travel\}$

Going through the steps of the algorithm we have:

- $M_0 = M$.
- $M_1^+ = M_0^+ \setminus \{mountain, beach\} = \emptyset$, $M_1^- = M_0^-$.
- M_1^- is not weakly admissible, so we non-deterministically select one L such that $not\ L \in M_1^-$ is one of the causes for M_1^- not complying to the weak admissibility condition: for example, $L = mountain$. $M_2^- = M_1^- \setminus \{not\ mountain\} = \{not\ beach, not\ travel\}$. We repeat this set and now we must remove $not\ beach$ from M_2^-. $M_3^- = M_2^- \setminus \{not\ beach\} = \{not\ travel\}$.
- $M_3^+ = M_2^+ = M_1^+ = \emptyset$ is unavoidable.
- $P \cup M_3$ is a consistent scenario.
- $P \cup M_3$ is not a 2-valued complete scenario. So $M_4^+ = M_3^+ \cup \{mountain\}$ because $mountain$ is the only literal which verifies $P \cup M_3 \nvdash mountain$ and $P \cup M_3 \nvdash not\ mountain$. Now we go on to step 4 of the algorithm again.
- M_4^+ is unavoidable.
- $P \cup M_4$ is consistent.
- $P \cup M_4$ is 2-valued complete, so $H = M_4^+ \cup M_4^- = \{mountain, not\ travel\}$ and the algorithm terminates successfully.

In the end, the resulting model is $least(P \cup H) = \{mountain, beach, not\ travel\}$. Notice that $beach$ is just a consequence of $not\ travel$ in P, it does not have to be a hypothesis. If other atoms were to be chosen at step 3 other alternative solutions would be found.

5 Integrity Constraints

Example 12. **Middle Region Politics Revisited** Recall the example 4 presented earlier. We are now going to add extra complexity to it.

We already know the two factions which are at odds and their thinking.

$$oppression \leftarrow not\ end\_of\_terrorism \quad end\_of\_terrorism \leftarrow oppression$$
$$terrorism \leftarrow not\ end\_of\_oppression \quad end\_of\_oppression \leftarrow terrorism$$

We now combine these two sets of rules with the two following Integrity Constraints (ICs) which guarantee that $oppression$ and $end\_of\_oppression$ are never simultaneously true; and the same happens with terror:

$$falsum \leftarrow oppression, end\_of\_oppression, not\ falsum$$
$$falsum \leftarrow terrorism, end\_of\_terrorism, not\ falsum$$

So far so good, there is still a single joint set of hypotheses resulting in a consistent scenario $\{end\_of\_oppression, end\_of\_terrorism\}$. Still, there is no SM for this program. But introducing either one or both of the next two rules, makes it impossible to satisfy the ICs:

$$oppression \leftarrow not\ terrorism \quad terrorism \leftarrow not\ oppression$$

In this case all the consistent and 2-valued complete scenarios contain the atom $falsum$. There are still no Stable Models for the resulting program. The semantics we propose allows two models for this program, which correspond to the 2-valued complete consistent scenarios, both containing $falsum$. We can discard them on this account or examine their failure to satisfy the ICs.

6 Conclusions and Future Work

We have managed to assign a complete 2-valued semantics to every Normal Logic Program, by employing an argumentation framework that readily extends the argumentation framework of Stable Models semantics. We also presented three algorithms for finding the Revision Complete Scenario of any Normal Logic Program. Every Stable Model of a Normal Logic Program corresponds to a Revision Complete Scenario and, in that sense, our algorithms allow for a different perspective on Stable Models semantics: any Stable Model can be seen as the result of an iterative process of Inconsistency Removal or Inconsistency Avoidance. In any case, Stable Models are the final result of such inconsistency removal/avoidance where any initial positive hypotheses remain in the end. In the process, we have extended argumentation with *Reductio ad Absurdum* reasoning for that purpose, and shown how Collaborative Argumentation can be defined in that context.

Future work concerns the extension to Generalized Logic Programs and Extended Logic Programs, and the seamless merging with more general belief revision in Logic Programs.

Some of the applications enabled by this improved semantics of Normal Logic Programs, concern the ability to guarantee that the meaning of diverse programs, e.g. arising from Semantic Web usage, always has a semantics. Similarly, we can also ensure this property whenever updating programs, including the case where an autonomous program evolves through self-updating [1]. Such applications will be enabled by the ongoing implementation.

Acknowledgments We deeply thank Robert A. Kowalski for his crucial help in clarifying our ideas and their presentation.

References

1. J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In S. Flesca et al., editor, *JELIA*, volume 2424 of *LNCS*, pages 50–61. Springer, 2002.
2. J. J. Alferes and L. M. Pereira. An argumentation theoretic semantics based on non-refutable falsity. In J. Dix et al., editor, *NMELP*, pages 3–22. Springer, 1994.
3. A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93:63–101, 1997.
4. P. M. Dung. Negations as hypotheses: An abductive foundation for logic programming. In *ICLP*, pages 3–17. MIT Press, 1991.
5. P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *ICLP*, pages 616–630. MIT Press, 1993.
6. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
7. P. M. Dung, R. A. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artif. Intell.*, 170(2):114–159, 2006.
8. P. M. Dung, P. Mancarella, and F. Toni. Argumentation-based proof procedures for credulous and sceptical non-monotonic reasoning. In *Computational Logic: Logic Programming and Beyond*, volume 2408 LNCS, pages 289–310. Springer, 2002.
9. P. M. Dung and T. C. Son. An argument-based approach to reasoning with specificity. *Artif. Intell.*, 133(1-2):35–85, 2001.
10. F. Fages. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science*, 1:51–60, 1994.
11. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, pages 1070–1080. MIT Press, 1988.
12. A. C. Kakas and P. Mancarella. Negation as stable hypotheses. In *LPNMR*, pages 275–288. MIT Press, 1991.
13. A. C. Kakas and F. Toni. Computing argumentation in logic programming. *J. Log. Comput.*, 9(4):515–562, 1999.
14. L. M. Pereira and A. M. Pinto. Revised stable models - a semantics for logic programs. In G. Dias et al., editor, *Progress in AI*, volume 3808 of *LNCS*, pages 29–42. Springer, 2005.
15. A. M. Pinto. Explorations in revised stable models — a new semantics for logic programs. Master's thesis, Universidade Nova de Lisboa, February 2005.
16. L. Soares. Revising undefinedness in the well-founded semantics of logic programs. Master's thesis, Universidade Nova de Lisboa, 2006.
17. F. Toni and R. A. Kowalski. An argumentation-theoretic approach to logic program transformation. In *LOPSTR*, volume 1048 of *LNCS*, pages 61–75. Springer, 1996.

Inferring Preferred Extensions by Minimal Models

Juan Carlos Nieves[1], Mauricio Osorio[2], and Ulises Cortés[1]

[1] Universitat Politècnica de Catalunya
Software Department (LSI)
c/Jordi Girona 1-3, E08034, Barcelona, Spain
{jcnieves,ia}@lsi.upc.edu
[2] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@googlemail.com

Abstract. We identify that there is a direct relationship between the minimal models of a propositional formula and the preferred extensions of an argumentation framework. Then we show how to infer the preferred extensions of an argumentation framework by using UNSAT algorithms and disjunctive answer set solvers.

1 Introduction

Although several approaches have proposed for argument theory, Dung's approach, presented in [11], is a unifying framework which has played an influential role on argumentation research and Artificial Intelligence (AI). In fact, Dung's approach has influenced subsequent proposals for argumentation systems, *e.g.,* [18, 3]. Besides, Dung's approach is mainly relevant in fields where conflict management plays a central role. For instance, Dung showed that his theory naturally captures the solutions of the theory of n-person game and the well-known stable marriage problem.

Dung defined four argumentation semantics: *stable semantics*, *preferred semantics*, *grounded semantics*, and *complete semantics*. The central notion of these semantics is the *acceptability of the arguments*. An argument is called *acceptable* if and only if it belongs to a set of arguments which is called *extension*. The main argumentation semantics for collective acceptability are the grounded semantics and the preferred semantics [16, 1]. The first one represents a skeptical approach, since for a given argumentation framework the grounded semantics always identifies a single extension, called grounded extension. The preferred semantics instead represents a credulous approach, since for a given argumentation framework it identifies a set of extensions which are called preferred extensions.

It is well-known that the implementation of the decision problem of the grounded semantics is quite straightforward. However, the decision problem of the preferred semantics is hard since it is co-NP-Complete [12]. In the literature, we can find different algorithms for computing the preferred semantics [4, 6, 9,

10, 2]. We have to point out that these algorithms are so specific; they are not really flexible for developing small prototypes.

From the point of view that a proper representation of a given problem is a major step in finding robust solutions to it, we explore a couple of representations of an argumentation framework in order to compute their preferred extensions. In general terms, we identify that there is a direct relationship between the minimal models of a propositional formula and the preferred extensions of an argumentation framework. We show how to infer the *preferred extensions* of an argumentation framework by using *UNSAT algorithms* and *disjunctive answer set solvers e.g.,* DLV [8]. UNSAT is the complement of Satisfiability (SAT), a problem for which very efficient systems have been developed in AI during the last decade. Nowadays, there are fast answer set solvers *e.g.,* DLV [8], SMODELS [17], which have contributed to extend the applications of Answer Set Programming (ASP).

The rest of the paper is divided as follows: In §2, we present some basic concepts of logic programs and argumentation theory. In §3, we present a characterization of the preferred semantics by minimal models. In §4, we present how to compute the preferred semantics by using the minimal models of a positive disjunctive logic program. Finally in the last section, we present our conclusions.

2 Background

In this section, we present the syntax of a valid logic program in ASP, the definition of an answer set, and the definition of the preferred semantics. We will use basic well-known definitions in complexity theory such as co-NP-complete problem. We suggest the reader to consult [7] if s/he needs to read more on such definitions.

2.1 Logic Programs: Syntax

The language of a propositional logic has an alphabet consisting of

(i) proposition symbols: $p_0, p_1, ...$
(ii) connectives : $\vee, \wedge, \leftarrow, \neg, \bot, \top$
(iii) auxiliary symbols : (,).

where $\vee, \wedge, \leftarrow$ are 2-place connectives, $\neg$ is 1-place connective and $\bot, \top$ are 0-place connectives. The proposition symbols and $\bot$ stand for the indecomposable propositions, which we call *atoms*, or *atomic propositions*. A literal is an atom, a, or the negation of an atom $\neg a$. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of literals $\{\neg a_1, ..., \neg a_n\}$.

A general clause, C, is denoted by $a_1 \vee \ldots \vee a_m \leftarrow l_1, \ldots, l_n$,[3] where $m \geq 0$, $n \geq 0$, each a_i is an atom, and each l_i is a literal. When $n = 0$ and $m > 0$ the clause is an abbreviation of $a_1 \vee \ldots \vee a_m \leftarrow \top$, where $\top$ is $\neg\bot$. When $m = 0$

[3] $l_1, \ldots, l_n$ represents the formula $l_1 \wedge \cdots \wedge l_n$.

the clause is an abbreviation of $\bot \leftarrow l_1, \ldots, l_n$. Clauses of this form are called constraints (the rest, non-constraint clauses). A general program, P, is a finite set of general clauses. By $\mathcal{L}_P$, we denote the set of atoms that occurs in P. Given a set S and $E \subseteq S$, $\widetilde{E}$ denotes the complement of E *w.r.t.* S.

We point out that whenever we consider logic programs our negation $\neg$ corresponds to the default negation *not* used in Logic Programming. Also, it is convenient to remark that in this paper we are not using at all the so called strong negation used in ASP.

2.2 Answer set semantics

First, to define the answer set semantics, let us define some relevant concepts. Let P be a general program. An interpretation I is a mapping from $\mathcal{L}_P$ to $\{0,1\}$, where the generalization of I to connectives is as follows: $I(a \wedge b) = min\{I(a), I(b)\}$, $I(a \vee b) = max\{I(a), I(b)\}$, $I(a \leftarrow b) = 0$ if and only if $I(b) = 1$ and $I(a) = 0$, $I(\neg a) = 1 - I(a)$, $I(\bot) = 0$. An interpretation I is called a model of P if and only if for each clause $c \in P$, $I(c) = 1$. Finally, I is a minimal model of P if it does not exist a model I' of P such that $I' \subset I$.

By using answer set programming, it is possible to describe a computational problem as a logic program whose answer sets correspond to the solutions of the given problem. The answer set semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [13] and it is usually studied in the context of syntax dependent transformations on programs. The following definition of an answer set for general programs generalizes the definition presented in [13] and it was presented in [14].

Let P be any general program. For any set $S \subseteq \mathcal{L}_P$, let P^S be the general program obtained from P by deleting

(i) each rule that has a formula $\neg l$ in its body with $l \in S$, and then
(ii) all formulæ of the form $\neg l$ in the bodies of the remaining rules.

Clearly P^S does not contain $\neg$, then S is an answer set of P if and only if S is a minimal model of P^S.

2.3 Argumentation theory

Now, we define some basic concepts of Dung's argumentation approach. The first one is an argumentation framework. An argumentation framework captures the relationships between the arguments (All the definitions of this subsection were taken from the seminal paper [11]).

Definition 1. *An argumentation framework is a pair* $AF := \langle AR, attacks \rangle$, *where* AR *is a finite set of arguments, and* attacks *is a binary relation on* AR, i.e. attacks $\subseteq AR \times AR$.

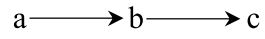

Fig. 1. A single argumentation framework

Any argumentation framework could be regarded as a directed graph. For instance, if $AF := \langle\{a, b, c\}, \{(a, b), (b, c)\}\rangle$, then AF is represented as shown in Fig. 1. We say that *a attacks b* (or *b* is attacked by *a*) if $attacks(a, b)$ holds. Similarly, we say that a set S of arguments attacks b (or b is attacked by S) if b is attacked by an argument in S. For instance in Fig. 1, $\{a\}$attacks b.

Definition 2. *A set S of arguments is said to be conflict-free if there are no arguments A, B in S such that A attacks B.*

By considering conflict-free sets of arguments, it is defined the concept of *admissible set*.

Definition 3. *(1) An argument $A \in AR$ is acceptable with respect to a set S of arguments if and only if for each argument $B \in AR$: If B attacks A then B is attacked by S. (2) A conflict-free set of arguments S is admissible if and only if each argument in S is acceptable w.r.t. S.*

For instance, the argumentation framework of Fig. 1 has two admissible sets: $\{a\}$ and $\{a, c\}$. The (credulous) semantics of an argumentation framework is defined by the notion of preferred extension.

Definition 4. *A preferred extension of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of AF.*

The only preferred extension of the argumentation framework of Fig. 1 is $\{a, b\}$.

3 Preferred extensions and UNSAT algorithms

In this section, we provide a method for computing preferred extensions. This method is based on model checking and Unsatisfiability (UNSAT). UNSAT is the complement of Satisfiability (SAT), a problem for which very efficient systems have been developed in AI during the last decade.

First of all, we introduce some notations which are used in the rest of the paper. Our representations of an argumentation framework use the predicate *d(X)*, where the intended meaning of *d(X)* is: "the argument X is defeated". Given an argumentation framework $AF := \langle AR, Attacks\rangle$ and $E \subseteq AR$, we define the set $s(E)$ as $\{d(a)|a \in AR \setminus E\}$. Essentially, $s(E)$ expresses the complement of E *w.r.t. AR*. Given $A \in AR$, we define $D(A)$ as $\{B|(B, A) \in Attacks\}$. Intuitively $D(A)$ denotes the set of arguments which attacks A.

Now we definite a mapping from an argumentation framework to a propositional formula.

Definition 5. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework, then $\alpha(AF)$ is defined as follows:*

$$\alpha(AF) := \bigwedge_{A\in AR}((\bigwedge_{B\in D(A)} d(A) \leftarrow \neg d(B)) \wedge (\bigwedge_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C)))$$

In the propositional formula $\alpha(AF)$, we can identify two parts for each argument $A \in AR$:

1. The first part $(\bigwedge_{B\in D(A)} d(A) \leftarrow \neg d(B))$ suggests that the argument A is defeated when one of its attackers is not defeated.
2. The last part $(\bigwedge_{B\in D(A)} d(A) \leftarrow \bigwedge_{C\in D(B)} d(C))$ suggests that the argument A is defeated when all the arguments that defend[4] A are defeated.

Notice that $\alpha(AF)$ is essentially a propositional formula (just considering the atoms like $d(a)$ as d_a). In order to illustrate the propositional formula $\alpha(AF)$, let us consider the following example.

Example 1. Let $AF := \langle AR, attacks\rangle$ be the argumentation framework of Fig. 1. We can see that $D(a) = \{\}$, $D(b) = \{a\}$ and $D(c) = \{b\}$. Hence if we consider the propositional formula *w.r.t.* argument a, we obtain (in order to be syntactically clear we use uppercase letters as variables and lowercase letters as constants):

$$(\bigwedge_{B\in\{\}} d(a) \leftarrow \neg d(B)) \wedge (\bigwedge_{B\in\{\}} d(a) \leftarrow \bigwedge_{C\in D(B)} d(C)) \equiv \top \wedge \top \equiv \top$$

It is important to remember that the conjunction of an empty set is the true value ($\top$). Now if one considers the propositional formula *w.r.t.* argument b, we get

$$(\bigwedge_{B\in\{a\}} d(b) \leftarrow \neg d(B)) \wedge (\bigwedge_{B\in\{a\}} d(b) \leftarrow \bigwedge_{C\in D(B)} d(C)) \equiv$$
$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \bigwedge_{C\in D(a)} d(C)) \equiv (d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top)$$

And the propositional formula *w.r.t.* argument c is

$$(\bigwedge_{B\in\{b\}} d(c) \leftarrow \neg d(B)) \wedge (\bigwedge_{B\in\{b\}} d(c) \leftarrow \bigwedge_{C\in D(B)} d(C)) \equiv$$
$$(d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))$$

Then, $\alpha(AF)$ is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))$$

Essentially $\alpha(AF)$ is a propositional representation of the argumentation framework AF. However $\alpha(AF)$ has the property that its minimal models characterize AF's preferred extensions. In order to formalize this property, let us consider the following proposition which was proved by Besnard and Doutre in [4].

[4] We say that C defends A if B attacks A and C attacks B.

Proposition 1. *[4] Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. A set $S \subseteq AR$ is a preferred extension if and only if S is a maximal model of the formula*

$$\bigwedge_{A\in AR}((A \rightarrow \bigwedge_{B\in D(A)} \neg B) \wedge (A \rightarrow \bigwedge_{B\in D(A)} (\bigvee_{C\in D(B)} C)))$$

Notice that $\alpha(AF)$ is related to defeated arguments and the formula of Proposition 1 is related to acceptable arguments. It is not difficult to see that $\alpha(AF)$ is the dual formula of the formula of Proposition 1. For instance, let us consider the argumentation framework AF of Example 1. The formula related to AF, according to Proposition 1, is:

$$(\neg a \leftarrow b) \wedge (\bot \leftarrow b) \wedge (\neg b \leftarrow c) \wedge (a \leftarrow c)$$

If we replace each atom x by the expression $\neg d(x)$, we get:

$$(\neg\neg d(a) \leftarrow \neg d(b)) \wedge (\bot \leftarrow \neg d(b)) \wedge (\neg\neg d(b) \leftarrow \neg d(c)) \wedge (\neg d(a) \leftarrow \neg d(c))$$

Now, if we apply transposition to each implication

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a))$$

The latter formula corresponds to $\alpha(AF)$. The following theorem is a straightforward consequence of Proposition 1.

Theorem 1. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is a minimal model of $\alpha(AF)$.*

In order to illustrate Theorem 1, let us consider again $\alpha(AF)$ of Example 1. This formula has three models: $\{d(b)\}$, $\{d(b), d(c)\}$ and $\{d(a), d(b), d(c)\}$. Then, the only minimal model is $\{d(b)\}$, this implies that $\{a, c\}$ is the only preferred extension of AF. In fact, each model of $\alpha(AF)$ implies an admissible set of AF, this means that $\{a, c\}$, $\{a\}$ and $\{\}$ are the admissible sets of AF.

There are several approaches for inferring minimal models from a propositional formula. For instance, it is possible to use UNSAT's algorithms for inferring minimal models. Hence, it is clear that we can use UNSAT's algorithms for computing the preferred extensions of an argumentation framework. This idea is formalized with the following lemma. Let S be a set of well formed formulæ then we define $n(S) := \bigwedge_{c\in S} c$.

Lemma 1. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is a model of $\alpha(AF)$ and $\alpha(AF) \wedge n(\neg\widetilde{s(S)}) \wedge \neg n(s(S))$ is unsatisfiable.*

Proof. It is direct by Theorem 1.

In order to illustrate Lemma 1, let us consider again the argumentation framework AF of Example 1. Let $S = \{a\}$, then $s(S) = \{d(b), d(c)\}$. We have already seen that $\{d(b), d(c)\}$ is a model of $\alpha(AF)$, hence the formula to verify its unsatisfiability is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a)) \wedge \\ \neg d(a) \wedge (\neg d(b) \vee \neg d(c))$$

However, this formula is satisfiable by the model $\{d(b)\}$, then $\{a\}$ is not a preferred extension. Now, let $S = \{a, c\}$, then $s(S) = \{d(b)\}$. As seen before, $\{d(b)\}$ is also a model of $\alpha(AF)$, hence the formula to verify its unsatisfiability is:

$$(d(b) \leftarrow \neg d(a)) \wedge (d(b) \leftarrow \top) \wedge (d(c) \leftarrow \neg d(b)) \wedge (d(c) \leftarrow d(a)) \wedge \\ \neg d(a) \wedge \neg d(c) \wedge \neg d(b)$$

It is easy to see that this formula is unsatisfiable, therefore $\{a, c\}$ is a preferred extension.

The relevance of Lemma 1 is that UNSAT is the prototypical and best-researched co-NP-complete problem. Hence, Lemma 1 opens the possibilities for using a wide variety of algorithms for inferring the preferred semantics.

4 Preferred extensions and general programs

In Section 3, we presented a representation of an argumentation framework in terms of a propositional formula for inferring preferred extensions. Another option for computing the preferred semantics is by considering a straightforward mapping from an argumentation framework to a *general program*. This approach is an *elegant* and *short* form for inferring the preferred extensions of an argumentation framework. The only system that we need for inferring the preferred extensions of an argumentation framework is any disjunctive answer set solver *e.g.,* DLV [8].

We start this section by defining a simple mapping from an argumentation framework to a positive disjunctive logic program.

Definition 6. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:*

$$\Gamma(A) := (\bigwedge_{B \in D(A)} (d(A) \vee d(B))) \wedge (\bigwedge_{B \in D(A)} (d(A) \leftarrow \bigwedge_{C \in D(B)} d(C)))$$

The generalization of the function Γ is defined as follows:

Definition 7. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework. We define its associated general program as follows:*

$$\Gamma_{AF} := \bigwedge_{A \in AR} \Gamma(A)$$

Remark 1. Notice that $\alpha(AF)$ (see Definition 5) is similar to Γ_{AF}. The main syntactic difference of Γ_{AF} *w.r.t.* $\alpha(AF)$ is the first parte of Γ_{AF} which is $(\bigwedge_{B \in D(A)}(d(A) \vee d(B)))$; however this part is logical equivalent to the first part of $\alpha(AF)$ which is $(\bigwedge_{B \in D(A)} d(A) \leftarrow \neg d(B))$. In fact, the main difference is their behavior *w.r.t.* answer set semantics. In order to illustrate this difference, let us consider the argumentation framework $AF := \langle AR, attacks \rangle$, where $AR := \{a\}$ and $attacks := \{(a, a)\}$. Then we can see that

$$\Gamma_{AF} := (d(a) \vee d(a)) \wedge (d(a) \leftarrow d(a))$$

and

$$\alpha(AF) := (d(a) \leftarrow \neg d(a)) \wedge (d(a) \leftarrow d(a))$$

It is clear that both formulæ have a minimal model which is $\{d(a)\}$, however $\alpha(AF)$ has no answer sets. In fact both formulæ are logically equivalent in classic logic but not in answer set semantics.

In the following theorem we formalize a characterization of the preferred semantics in terms of positive disjunctive logic programs and answer set semantics.

Theorem 2. *Let $AF := \langle AR, attacks \rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if $s(S)$ is an answer set of Γ_{AF}.*

Proof. S is a preferred extension of AF if and only if *s(S)* is a minimal model of $\alpha(AF)$ (by Theorem 1) if and only if $s(S)$ is a minimal model of Γ_{AF} (since Γ_{AF} is logically equivalent to $\alpha(AF)$ in classical logic) if and only if *s(S)* is an answer set of Γ_{AF} (since Γ_{AF} is a positive disjunctive program and for every positive disjunctive program P, M is an answer set of P if and only if M is a minimal model of P).

Let us consider the following example.

Example 2. Let $AF := \langle AR, attacks \rangle$ be an argumentation framework, where $AR := \{a, b, c, d, e\}$ and $attacks := \{(a, b), (b, a), (b, c), (c, d), (d, e), (e, c)\}$ (see Fig. 2). Then, Γ_{AF} is

$$\begin{array}{ll} d(a) \vee d(b). & d(a) \leftarrow d(a). \\ d(b) \vee d(a). & d(b) \leftarrow d(b). \\ d(c) \vee d(b). & d(c) \vee d(e). \\ d(c) \leftarrow d(a). & d(c) \leftarrow d(d). \\ d(d) \vee d(c). & d(d) \leftarrow d(b), d(e). \\ d(e) \vee d(d). & d(e) \leftarrow d(c). \end{array}$$

Γ_{AF} has two answer sets which are $\{d(a), d(c), d(e)\}$ and $\{d(b), d(c), d(e), d(d))\}$, therefore $\{b, d\}$ and $\{a\}$ are the preferred extensions of AF.

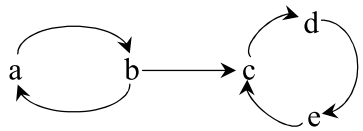

Fig. 2. An argumentation framework.

An alternative form for computing the preferred extensions of an argumentation framework, without considering the predicate *d(X)*, is taking advantage of *default negation*. It is possible by considering a new dual symbol for each argument of the argumentation framework. This means that we can infer the acceptable arguments directly from the answers sets of the logic program.

This idea is formalized with the following lemma. First, let us present some definitions.

Definition 8. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define the function η as $\eta : AR \to AR'$. Where AR' has the same cardinality to AR such that $AR \cap AR' = \emptyset$.*

η is a bijective function which assigns a new symbol to each argument of AR. Notice that the new symbol does not occurs in AR. We are going to denote the image of $A \in AR$ under η as A'.

Definition 9. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $A \in AR$. We define the transformation function $\Gamma(A)$ as follows:*

$$\Lambda(A) := (\bigwedge_{B \in D(A)} (A' \vee B')) \wedge (\bigwedge_{B \in D(A)} (A' \leftarrow \bigwedge_{C \in D(B)} C'))$$

Definition 10. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework. We define its associated general program as follows:*

$$\Lambda_{AF} := \bigwedge_{A \in AR} (\Lambda(A) \wedge (A \leftarrow \neg A'))$$

Notice that $\Gamma(A)$ and $\Lambda(A)$ are equivalent (module notation) and the main difference between Γ_{AF} and Λ_{AF} is the rule $A \leftarrow \neg A'$ for each argument.

Lemma 2. *Let $AF := \langle AR, attacks\rangle$ be an argumentation framework and $S \subseteq AR$. S is a preferred extension of AF if and only if there is an answer set M of Λ_{AF} such that $S = M \cap AR$.*

Proof. The proof is straightforward from Theorem 2 and the semantics of default negation.

In order to illustrate this lemma let us consider the following example.

Example 3. Let $AF := \langle AR, attacks \rangle$ be the argumentation framework of Example 2. So Λ_{AF} is

$$
\begin{array}{ll}
a' \vee b'. & a' \leftarrow a'. \\
b' \vee a'. & b' \leftarrow b'. \\
c' \vee b'. & c' \vee e'. \\
c' \leftarrow a'. & c' \leftarrow d'. \\
d' \vee c'. & d' \leftarrow b', e'. \\
e' \vee d'. & e' \leftarrow c'. \\
a \leftarrow \neg a'. & b \leftarrow \neg b'. \\
c \leftarrow \neg c'. & d \leftarrow \neg d'. \\
e \leftarrow \neg e'. &
\end{array}
$$

Γ_{AF} has two answer sets which are $\{a', c', e', b, d\}$ and $\{b', c', e', d', a\}$, hence $\{b, d\}$ and $\{a\}$ are the preferred extensions of AF.

5 Conclusions

The preferred semantics is regarded as the most satisfactory argumentation semantics of Dung's argumentation approach. For instance, John Pollock made preferred semantics one of the key ingredients of his revised formalism [15]. Also, it has been shown that some non-monotonic logic programming semantics can be viewed as a special form of this abstract argumentation semantics [5, 11].

It is well-known that the decision problem of the preferred semantics is co-NP-Complete. Then, to have different approaches for inferring this semantics could help to develop argumentation systems based on the preferred semantics. The inference of minimal models from a propositional formula and a logic program is a widely explored problem. Therefore, by defining a relationship between the preferred semantics and minimal models, we identify a wide family of algorithms for inferring the preferred semantics. In particular in this paper, we show how to infer the preferred extensions of an argumentation framework by using UNSAT algorithms and disjunctive answer set solvers.

Acknowledgement

We are grateful to anonymous referees for their useful comments. J.C. Nieves thanks to CONACyT for his PhD Grant. J.C. Nieves and U. Cortés were partially supported by the grant FP6-IST-002307 (ASPIC). The views expressed in this paper are not necessarily those of ASPIC consortium.

References

1. ASPIC:Project. *Deliverable D2.2:Formal semantics for inference and decision-making.* Argumentation Service Plarform with Integrated Components, 2005.

2. ASPIC:Project. ASPIC: Argumentation engine demo. `http://aspic.acl.icnet.uk/`, 2006.
3. T. Bench-Capon. Value-based argumentation frameworks. In *Proceedings of Non Monotonic Reasoning*, pages 444–453, 2002.
4. P. Besnard and S. Doutre. Checking the acceptability of a set of arguments. In *Tenth International Workshop on Non-Monotonic Reasoning (NMR 2004),*, pages 59–64, June 2004.
5. A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.
6. C. Cayrol, S. Doutre, and J. Mengin. On Decision Problems related to the preferred semantics for argumentation frameworks. *Journal of Logic and Computation*, 13(3):377–403, 2003.
7. T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.
8. S. DLV. Vienna University of Technology. `http://www.dbai.tuwien.ac.at/proj/dlv/`, 1996.
9. S. Doutre and J. Mengin. An algorithm that computes the preferred extensions of argumentation frameworks. In *ECAI'2000, Third International Workshop on Computational Dialectics (CD'2000)*, pages 55–62, Aug. 2000.
10. S. Doutre and J. Mengin. Preferred Extensions of Argumentation Frameworks: Computation and Query Answering. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 272–288. Springer-Verlag, 2001.
11. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.
12. P. E. Dunne and T. J. M. Bench-Capon. Complexity in value-based argument systems. In *JELIA*, volume 3229 of *LNCS*, pages 360–371. Springer, 2004.
13. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
14. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
15. J. L. Pollock. *Cognitive Carpentry: a blueprint for how to build a person*. The MIT Press, May 4, 1995.
16. H. Prakken and G. A. W. Vreeswijk. Logics for defeasible argumentation. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume 4, pages 219–318. Kluwer Academic Publishers, Dordrecht/Boston/London, second edition, 2002.
17. S. SMODELS. Helsinki University of Technology. `http://www.tcs.hut.fi/Software/smodels/`, 1995.
18. G. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90(1-2):225–279, 1997.

Formal Properties of the $\mathcal{S}$CIFF-AF Multiagent Argumentation Framework

Paolo Torroni

DEIS, University of Bologna
V.le Risorgimento 2, 40136 Bologna, Italy
Paolo.Torroni@UniBO.it

Abstract. Argumentation theories have recently emerged and gained popularity in the agents community, since argumentation represents a natural and intuitive way to model non-monotonic reasoning. In a multiagent context, argumentation has recently been proposed as a component of dialogue frameworks. However, despite the large interest in argumentation theories in multiagent domains, most proposed frameworks stay at a general though abstract level, and operational counterparts to abstract frameworks are not many. The aim of this work is to present the main formal properties of the $\mathcal{S}$CIFF-AF: an operational argumentation-based multiagent dialogue framework.

1 Introduction

Argumentation theories have recently emerged and gained popularity in the agents community, since argumentation represents a natural and intuitive way to model non-monotonic reasoning. In a multiagent context, argumentation has recently been proposed as a component of dialogue frameworks. A typical setting is that of collaborative problem solving, for example to tackle resource allocation and achievement [17]. In such a context, multiple agents have to coordinate in order to take joint decisions about possible allocations of resources.

In general, argumentative reasoning can be utilized by agents intending to decide about possible future courses of action. Typically, in collaborative problem solving domains, individual agents have own goals to achieve and own constraints to satisfy, but they are situated in a common environment in which there are resources they need to share. Thus when agents take actions they need to ensure that their activity does not clash with other agents' actions and constraints.

We then have to consider two aspects of collaborative problem solving: from an individual's perspective, an agent should be able to reason about what is the most appropriate course of action to take in a given situation. We believe that the theories and logics of argumentation are a very promising approach to this problem. From a "social" perspective, instead, agents can use argumentation in order to engage in dialogues, and use their arguments to make their decisions accepted by other agents.

The first aspect is related to decision making and practical reasoning, central issues in agent architectures and reasoning since the early days of BDI agent

models and implementations [16, 8] and further pursued in recent work such as [14, 10, 5]. The aspect of argumentation in agent dialogue has also been addressed by conspicuous work [4, 17, 13], and considered by many as the natural evolution of agent dialogue in domains such as negotiation [15]. Among others, Atkinson *et al.* explore the issue of multiagent argument over proposals for action [5].

Despite the large interest in argumentation theories in multiagent domains, most proposed frameworks stay at a general though abstract level, and seldom there exist operational counterparts to much of the existing proposals. One important contribution in this direction is work by Kakas and Toni [11] on mapping Dung's abstract argumentation framework [6] onto the Abductive Logic Programming (ALP) framework [9].

Drawing inspiration from Atkinsons *et al.*'s work about the PARMA action persuasion protocol [5], we have proposed an Argumentation Framework [19] based on the ALP $\mathcal{S}$CIFF framework ($\mathcal{S}$CIFF-AF) for multiagent argumentation, aimed at addressing explicitly this aspect. $\mathcal{S}$CIFF-AF encompasses multiagent dialogues over proposals for action, and it is equipped with a declarative and operational model with an ALP semantics.

The formal foundations of this framework rely on previous results from ALP, and from Dung's studies on argumentation. Basically, $\mathcal{S}$CIFF-AF is a casting of Dung's abstract argumentation framework in $\mathcal{S}$CIFF, augmented with a notion of 2-party agent dialogue and agreement over proposals for actions. In fact, agent dialogues in $\mathcal{S}$CIFF-AF can be used by the interacting parties to reach a consensus on a possible future course of action and consequent state, and ultimately such actions may be adopted by agents as future internal goals.

The aim of this work is to present the main formal properties of $\mathcal{S}$CIFF-AF, which insure a consistent and meaningful system evolution. We will start by showing semantic properties of the argumentation framework in relation with Dung's argumentation semantics. Later, we will refine the definition of multiagent dialogue proposed in [19] and we will show what properties multiagent agreements exhibit.

2 Background

$\mathcal{S}$CIFF-AF is built on three main ingredients: Dung's abstract argumentation framework [6], the $\mathcal{S}$CIFF language and Abductive Logic Programming (ALP) proof-procedure [2], and the PARMA action persuasion protocol and its locutions [5].

ALP is a computational paradigm aimed to introduce hypothetical reasoning in the context of Logic Programming (see [12] for an introduction to LP and [9] for a survey on ALP). A logic program $\mathcal{P}$ is a collection of clauses, with an associated notion of entailment, usually indicated by $\models$. In ALP, some predicates ("*abducibles*"), belonging to a special set $\mathcal{A}$, can be assumed to be true, if need be. In order to prevent unconstrained hypothesis-making, $\mathcal{P}$ is typically augmented with expressions which *must be true at all times*, called *integrity constraints* ($\mathcal{IC}$).

An *abductive logic program* is the triplet $\langle \mathcal{P}, \mathcal{A}, \mathcal{IC} \rangle$, with an associated notion of abductive entailment.

$\mathcal{S}$CIFF is an ALP proof-procedure defined by Alberti *et al.* [2] as an extension of Fung and Kowalski's IFF [7], and it is the reference ALP framework for this work. One distinguishing feature of $\mathcal{S}$CIFF is its notion of *expectations* about events. Expectations are abducibles denoted as $\mathbf{E}(X)$ (*positive* expectations) and $\mathbf{EN}(X)$ (*negative* expectations), where $\mathbf{E}(X)/\mathbf{EN}(X)$ stand for "X is expected/expected not to happen". Variables in events, expectations and in other atoms can be subject to CLP constraints and quantifier restrictions.

Two fundamental concepts in $\mathcal{S}$CIFF are those of *consistency* and *entailment*. We report their definition below.

Definition 1 (Consistent sets of hypotheses). *A set of hypotheses Δ is* consistent *if and only if* $\forall$ (ground) p,
$\{p, not\, p\} \nsubseteq \Delta$ *and* $\{\mathbf{E}(p), \mathbf{EN}(p)\} \nsubseteq \Delta$

Definition 2 (Entailment). *A ($\mathcal{S}$CIFF) ALP $\mathcal{S} = \langle \mathcal{P}, \mathcal{A}, \mathcal{IC} \rangle$* entails *a goal G (written $\mathcal{S} \models_\Delta G$), if and only if:*

$$\begin{cases} Comp(\mathcal{P} \cup \Delta) \cup CET \cup T_\chi \models G\sigma \\ Comp(\mathcal{P} \cup \Delta) \cup CET \cup T_\chi \models \mathcal{IC} \end{cases}$$

where Comp is the symbol of completion, CET is Clark's equality theory, $\models$ is Kunen's logical consequence relation for three-valued logic, σ is a substitution of ground terms for the variables in G, T_χ the theory of constraints, and Δ a consistent *subset of $\mathcal{A}$.*

$\mathcal{S}$CIFF operates by considering G together with $\mathcal{IC}$ as the initial goal, and by calculating a *frontier* as a disjunction of conjunctions of formulae, using at each step one among the inference rules defined in [2]. Given the frontier, at any step a selection function can be used to pick one among all the equally true disjuncts in the frontier. When no more inference rule applies (*quiescence*), if there exists at least one disjunct which is not false, then $\mathcal{S}$CIFF has succeeded, and Δ contains an answer to G. The $\mathcal{S}$CIFF proof-procedure is sound, and under reasonable restrictions it is also complete [2]. $\mathcal{S}$CIFF has been implemented and instantiated into a large number of scenarios involving agent communication, and it can be downloaded from its web site.[1]

Following Kakas and Toni [11], in $\mathcal{S}$CIFF-AF arguments are mapped onto abducibles. For example, an assumption $\mathbf{E}(p)$, "p is expected", could be considered as a argument which possibly supports some goal g. Arguments can be circumstances (in the sense of [5]), actions, and related constraints. Thus an agent may justify a goal g by saying, e.g., "in order to achieve a goal g, under the circumstances c and the constraints x, actions a_1 and a_2 should be carried out." In order to take this kind of position, an agent will utter the various elements of it (the circumstances, the goal, the actions, the constraints) via a suitable

[1] `http://lia.deis.unibo.it/research/sciff/`.

argumentation language and using the appropriate locutions. Argumentation dialogues will provide implicit links among such uttered elements.

Our proposed argumentation framework is an instantiation of Dung's work [6] and of the abstract computational framework developed by Kakas and Toni [11]. In particular, Dung's notion of *attack* is rephrased in the following way:

Definition 3. *A set of arguments A attacks another set Δ if and only if at least one of the following expressions is true:*

(1) $\mathcal{S} \models_A not\, p$, *for some* $p \in \Delta$;
(2) $\mathcal{S} \models_A \mathbf{E}(p)$, *for some* $\mathbf{EN}(p) \in \Delta$;
(3) $\mathcal{S} \models_A \mathbf{EN}(p)$, *for some* $\mathbf{E}(p) \in \Delta$;

Definition 4. *An Argumentation Framework (AF) is the pair* $\langle \mathcal{S}, attacks \rangle$.

In a multiagent context, agents can locally reason about circumstances, constraints, and actions (not) to be taken, based on the $\mathcal{S}$CIFF-AF, and produce – at the social level – dialogues in the style of PARMA dialogues.

PARMA considers a general argument schema for a rational position proposing an action, and handles possible attacks on one or more elements of a general argument schema. Attacks arise from disagreements originating from different sources. PARMA uses four categories of *locutions*, for dialogue control (C), action proposal (P), inquiry (A), and denial (D) of existence/validity of elements of a position. Such elements could be goals, circumstances, and actions (not) to be taken. While Atkinson *et al.* focus on addressing divergences on all elements of a position, $\mathcal{S}$CIFF-AF focusses instead on a more restricted number of issues, and adopts only a small set of locutions. In particular, it only considers some control locutions (C) and some proposal/denial locutions about circumstances and actions (P/D).[2]

Definition 5 (Agent system). *An* agent system *is a finite set Σ, where each $x \in \Sigma$ is a ground term, representing the name of an agent, equipped with a $\mathcal{S}$CIFF program $\mathcal{S} = \{\mathcal{P}, \mathcal{A}, \mathcal{IC}\}$.*

Definition 6 (Performative or dialogue move). *A* performative *or* dialogue move *p is an instance of a schema tell$(a, b, L[, Arg])$, where a is the* utterer*, b is the* receiver*, L is the* locution *and (optionally) Arg is the* argument *of the performative. For a given p, utterer$(p) = a$, receiver$(p) = b$, locution$(p) = L$ and argument$(p) = Arg$ (if present). The set of all possible performatives is called* argumentation language.

[2] A characteristic of PARMA is that it mixes elements of different levels, like turn-taking. We consider this as a feature rather than a limitation, since it makes it possible for agents to reason at different levels, and to implement high-level strategic decisions about which course a dialogue should follow. However, we will not cover this aspect in this work.

Note that Arg is optional, since a dialogue move not necessarily contains arguments all the time. In general, dialogue control (C) locutions will not need it. For instance, at start, an agent may simply want to declare that he is listening. In the definition below, we gear $\mathcal{S}$CIFF-AF with a concrete argumentation language inspired to PARMA.

Definition 7 (The argumentation language $\mathcal{L}_{arg}$). *The* argumentation language $\mathcal{L}_{arg}$ *is the set of all performatives p, such that:*

- *locution*$(p) \in \{$*'enter dialogue'*, *'leave dialogue'*, *'term finished'*, *'accept denial'*, *'state circumstances'*, *'deny circumstances'*, *'state actions'*, *'deny actions'*, $\}$, *and*
- *argument*(p) *is a conjunction of abducible atoms (possibly including* **E/EN** *expectations) and CLP constraints.*

$\mathcal{S}$CIFF-AF thus defines a concrete language for argumentation, $\mathcal{L}_{arg}$, which includes four dialogue control locutions (type C), two proposal locutions (P) and two denial locutions (D). Agents conversing in $\mathcal{L}_{arg}$ will not exchange formulae stating e.g. consequences of actions, such as implications, but only conjunctions of atoms.

Definition 8 (MAS argumentation framework). *A* MAS argumentation framework $\mathcal{M}$ *is a pair* $\langle \Sigma, Actions \rangle$ *where* Σ *is a multiagent system of agents with the same* $\mathcal{A}$ *which communicate using* $\mathcal{L}_{arg}$*, and Actions is a finite set, where each element is a ground term, representing the name of an action.*

Beside assuming a common language, $\mathcal{S}$CIFF-AF also assumes a common ontology (thus in Definition 8 $\mathcal{A}$ is the same for all agents in Σ). Otherwise some ontological middleware may be used so that, for example, in a position involving a sales, "buy" and "purchase" converge down to the same meaning. This is most necessary in open systems, to prevent misunderstandings arising from the use of terminology. Note that the presence of an argumentation framework based on ALP does not prevent agents from having and reasoning upon their private knowledge, and especially it does not prevent them from having private abducibles. However, for the sake of simplicity, in this article we will focus only on those abducibles which are functional to agent dialogue, and we assume that such abducibles are common to all agents for the reasons above.

In [19] argumentation dialogues are defined between two agents, and their evolution is modelled as a sequence of states. Each state contains a set of arguments modelling stated/agreed circumstances and actions, and possibly agreements reached by the agents.

3 Properties of SCIFF-AF

In this section we refine the original $\mathcal{S}$CIFF-AF framework. The idea is to define a notion of agent agreement about actions, and focus on the fundamental properties of multiagent agreements in $\mathcal{S}$CIFF-AF. Before doing so, we also discuss some important semantic properties of the $\mathcal{S}$CIFF-AF framework.

3.1 Admissible sets and grounded semantics of SCIFF-AF

Let us consider the *attacks* relation taken from [19] and reported in Section 2. From now on, if not explicitly mentioned otherwise, we will always refer to an arbitrary but fixed instance $\mathcal{S} = \langle \mathcal{P}, \mathcal{A}, \mathcal{IC} \rangle$ of a $\mathcal{S}$CIFF abductive framework. We will also use the terms "argument" and "hypothesis" interchangeably.

Lemma 1 *The following propositions are true:*

- *No set of arguments attacks the empty set of arguments* $\emptyset$*;*
- *attacks is* monotonic*, i.e. for all (consistent)* A*,* A'*,* Δ*,* $\Delta' \subseteq \mathcal{A}$*, if* A *attacks* Δ *then*
  - (i) *if* $A \subseteq A'$ *then* A' *attacks* Δ*, and*
  - (ii) *if* $\Delta \subseteq \Delta'$ *then* A *attacks* Δ'*;*
- *attacks is* compact*, i.e. for all* A*,* $\Delta \subseteq \mathcal{A}$*, if* A *attacks* Δ *then there exists a finite* $A' \subseteq A$ *such that* A' *attacks* Δ*;*

Proof. The first proposition follows from the definition of *attacks*. The second proposition follows from the fact that if $\mathcal{S} \models_A not\, p$, for some $p \in \Delta$, then $\forall A' \supseteq A, p \in A'$, therefore A' attacks Δ (i), and $\forall \Delta' \supseteq \Delta, not\, p \in \Delta'$, therefore A' attacks Δ (ii). The same holds if the attack is on some $\mathbf{E}(p)/\mathbf{EN}(p)$. The third proposition follows from the compactness of $\models_A$, by which finite expressions are always derivable from a finite set of antecedents.

These properties are considered by Kakas and Toni fundamental of an attacking relation [11, pag.518].

Remark 1. For an argument A such that $\mathcal{S} \models_A p$, it follows from the declarative semantics of $\mathcal{S}$CIFF that A is consistent, and that if an argument Δ is attacked by A, $A \cup \Delta$ is not consistent (in the sense of $\mathcal{S}$CIFF).

The definitions that follow are taken from Dung's abstract argumentation framework [6]. Corollaries 1 and 2 show the results of its instantiation in the $\mathcal{S}$CIFF framework.

Definition 9. *A set* Δ *of arguments is said to be* conflict-free *if there are no arguments* A *and* B *in* Δ *such that* A *attacks* B*.*

Corollary 1. *All consistent sets of arguments (in the sense of* $\mathcal{S}$*CIFF) are conflict-free.*

Proof. Let A be one among $\{not\, p, \mathbf{E}(p), \mathbf{EN}(p)\}$, and let $\hat{A}$ be the corresponding "attacked" hypothesis ($p, \mathbf{EN}(p)$, or $\mathbf{E}(p)$, respectively). Let Δ be a consistent set of arguments, and A, B two arguments in Δ. A *attacks* B means $\mathcal{S} \models_{\Delta'} A$ for some Δ', and $B = \hat{A}$; but this would imply that $\{A, \hat{A}\} \subseteq \Delta$. Contradiction!

As a consequence of of Remark 1 and Corollary 1, we have:

Corollary 2. *All arguments* A *such that* $\mathcal{S} \models_A p$ *are conflict-free.*

Finally, admissible sets of arguments are defined following Dung [6, Definition 6] and Kakas & Toni [11, Definition 2.3].

Definition 10. *A (conflict-free) set of arguments Δ is* admissible *iff for all sets of arguments A, if A attacks Δ, then Δ attacks $A \setminus \Delta$.*

Dung's Fundamental Lemma [6, pag. 327], together with the fact that the empty set is always admissible, implies the following corollary:

Corollary 3. *All arguments A such that $\mathcal{S} \models_A p$ are admissible sets of arguments for $\mathcal{S}$.*

Dung defines *preferred extensions* as maximal sets of admissible sets of arguments [6, Definition 7], but we will focus on admissible sets of arguments rather than on preferred extensions. In fact, as stressed by Kakas and Toni [11], since every admissible set of arguments is contained in some preferred extension, in order to determine whether a given query holds with respect to the preferred extension and partial stable model semantics, it is sufficient to determine whether the query holds with respect to the semantics of admissible sets.

Finally, the IFF proof-procedure upon which $\mathcal{S}$CIFF is built has a grounded argumentation semantics. Therefore we can conclude this section with a last important semantic property of the $\mathcal{S}$CIFF-AF framework.

Corollary 4. *All arguments A such that $\mathcal{S} \models_A p$ are grounded sets of arguments for $\mathcal{S}$.*

3.2 Properties of SCIFF-AF dialogues and agreements

In this section, we specialize the $\mathcal{S}$CIFF-AF dialogue framework, to define precisely what multiagent agreements are, and to show what properties they exhibit. The following definitions are based on the notions of agent system, performative, argumentation language and MAS argumentation framework given in Section 2.

Definition 11 (Dialogue). *Given an agent system Σ, a* dialogue $\mathcal{D}$ *in a language $\mathcal{L}$, between two agents $x, y \in \Sigma$, is an ordered set of performatives $\{p_0, p_1, \ldots\} \subseteq \mathcal{L}$, such that $\forall p_j = tell(a_j, b_j, L_j, A_j) \in \mathcal{D}, (a_j, b_j) \in \{(x, y), (y, x)\}$*

An example of dialogue will be provided later on (Example 1). The one above is a general definition, and it can be instantiated by choosing a concrete language, e.g. $\mathcal{L} = \mathcal{L}_{arg}$.

Definition 12 (State of a dialogue in $\mathcal{L}_{arg}$). *Given a dialogue $\mathcal{D}$ in $\mathcal{L}_{arg}$, for each j, $1 < j < |\mathcal{D}|$ the* state of the dialogue*, state($\mathcal{D}, j$) is a tuple*

$$\langle \Psi_j^{sc}, \Psi_j^{dc}, \Psi_j^{sa}, \Psi_j^{da}, \Psi_j^{aa} \rangle,$$

defined based on the dialogue history $\mathcal{D}_j = \{p_0, p_1, \ldots, p_{j-1}\}$ *as follows:*

– Ψ_j^{sc} *is the* set of stated circumstances*, defined as:*
$\Psi_j^{sc} = \{\ circ$ such that $\exists p_k \in \mathcal{D}_j \ \wedge\ k < j$
$\wedge$ *locution*(p_k) = *'state circumstances'*
$\wedge\ circ \in$ *argument*(p_k)
$\wedge\ \nexists p_l \in \mathcal{D}_j \ \wedge\ k < l < j$ such that (
locution(p_l) = *'state circumstances'*
$\wedge$ *argument*$(p_k) \neq$ *argument*(p_l)) $\}$
– Ψ_j^{dc} *is the* set of denied circumstances*, defined as:*
$\Psi_j^{dc} = \{\ circ$ such that $\exists p_k \in \mathcal{D}_j \ \wedge\ k < j$
$\wedge$ *locution*(p_k) = *'deny circumstances'*
$\wedge\ circ \in$ *argument*(p_k)
$\wedge\ \nexists p_l \in \mathcal{D}_j \ \wedge\ k < l < j$ such that
locution(p_l) = *'state circumstances'* $\}$
– Ψ_j^{sa} *is the* set of stated actions*, defined as:*
$\Psi_j^{sa} = \{\ \mathbf{E}(act)$ such that $\exists p_k \in \mathcal{D}_j \ \wedge\ k < j$
$\wedge$ *locution*(p_k) = *'state actions'*
$\wedge\ \mathbf{E}(act) \in$ *argument*(p_k)
$\wedge\ \nexists p_l \in \mathcal{D}_j \ \wedge\ k < l < j$ such that (
locution(p_l) = *'state actions'*
$\wedge$ *argument*$(p_k) \neq$ *argument*(p_l)) $\}$
– Ψ_j^{da} *is the* set of denied actions*, defined as:*
$\Psi_j^{da} = \{\ \mathbf{E}(act)$ such that $\exists p_k \in \mathcal{D}_j \ \wedge\ k < j$
$\wedge$ *locution*(p_k) = *'deny actions'*
$\wedge\ \mathbf{E}(act) \in$ *argument*(p_k)
$\wedge\ \nexists p_l \in \mathcal{D}_j \ \wedge\ k < l < j$ such that
locution(p_l) = *'state actions'* $\}$
– Ψ_j^{aa} *is the* set of agreed actions*, defined as:*
$\Psi_j^{aa} = \{\ \mathbf{E}(act)$ such that $\exists p_k, p_l \in \mathcal{D}_j$
$\wedge\ k < j \ \wedge\ l < j$
$\wedge$ *locution*(p_k) = *locution*(p_l) = *'state actions'*
$\wedge$ *argument*(p_k) = *argument*(p_l)
$\wedge\ \mathbf{E}(act) \in$ *argument*(p_k) $\}$

By Definition 12, the state of the dialogue at a step j with respect to circumstances/actions is determined by the last relevant move made.

Note that $state(D, j)$ is defined independently of control locutions, and that locutions *'state circumstances'* and *'state actions'* operate some sort of reset of the current state: if an agent utters *'state circumstances'* at step j, the set of stated circumstances will only contain the new circumstances Ψ_j^{sc}, until some agent again states *'state circumstances'*, and *'deny circumstances'* becomes the empty set, since the previously denied circumstances become obsolete. A similar semantics is that of *'state actions'* and *'deny actions'*. Note that memory of past moves is not necessarily lost, since agents may reason based on the previous states.

This definition of state is a specialization of the one given in [19]. We can immediately see what structural properties it exhibits:

Corollary 5. *Given a dialogue $\mathcal{D}$ in $\mathcal{L}_{arg}$, the state of $\mathcal{D}$ at step j, state$(\mathcal{D}, j) = \langle \Psi_j^{sc}, \Psi_j^{dc}, \Psi_j^{sa}, \Psi_j^{da}, \Psi_j^{aa} \rangle$, enjoys the following structural properties:*

1. *$\Psi_j^{dc} \subseteq \Psi_j^{sc}$ ("coherence" between the set of denied circumstances and the set of stated circumstances)*
2. *$\Psi_j^{da} \subseteq \Psi_j^{sa}$ ("coherence" between the set of denied actions and the set of stated actions)*
3. *$\Psi_j^{aa} = \Psi_j^{sa} \vee \Psi_j^{aa} = \emptyset$ ("coherence" between the set of agreed actions and the set of stated actions)*

Proof. The proof follows from Definition 12.

We can now proceed with defining the central concept of argumentation dialogue, which is as well a specialization of the one proposed in [19].

Definition 13 (Argumentation Dialogue). *Given a multiagent argumentation framework $\mathcal{M} = \langle \Sigma, Actions \rangle$, an* argumentation dialogue *$\mathcal{D}$ between $x, y \in \Sigma$, respectively equipped with $\mathcal{S}^x/\mathcal{S}^y$,* about a goal *$G_x$ is a dialogue in $\mathcal{L}_{arg}$ such that:*

1. *$p_0 = tell(x, y, \text{'enter dialogue'}, G_x)$;*
2. *$\forall p_j = tell(a_j, b_j, L_j, A_j) \in \mathcal{D}$:*
   (*i*) *if $L_j =$ 'state circumstances' then*

$$\mathcal{S}^{a_j} \models_{\Delta} G_x \cup \Psi_k^{sc} \cup \Psi_k^{sa}$$

   for some $k \leq j$, and argument$(p_j) = \Delta \setminus actions(\Delta)$;
   (*ii*) *if $L_j =$ 'state actions' then*

$$\mathcal{S}^{a_j} \models_{\Delta} G_x \cup \Psi_j^{sc} \cup \Psi_j^{sa}$$

   and argument$(p_j) = actions(\Delta)$;
   (*iii*) *if $L_j =$ 'deny circumstances' then*

$$\exists \Psi^{sc} \subseteq \Psi_j^{sc}, \Psi^{sa} \subseteq \Psi_j^{sa}, h \in \Psi_j^{sc} \setminus \Psi^{sc}$$

   such that $\mathcal{S}^{a_j} \cup \Psi^{sc} \cup \Psi^{sa} \models_{\Delta} h' \cup G_x$ and h' attacks h, and argument$(p_j) = h$;
   (*iv*) *if $L_j =$ 'deny actions' then*

$$\exists \Psi^{sc} \subseteq \Psi_j^{sc}, \Psi^{sa} \subseteq \Psi_j^{sa}, h \in \Psi_j^{sa} \setminus \Psi^{sa}$$

   such that $\mathcal{S}^{a_j} \cup \Psi^{sc} \cup \Psi^{sa} \models_{\Delta} h' \cup G_x$ and h' attacks h, and argument$(p_j) = h$;
   (*v*) *in all other cases, except for $L_j =$ 'enter dialogue', argument$(p_j) = \emptyset$.*
3. *$\nexists p_j, p_k \in \mathcal{D}$ such that $p_j = p_k \ \wedge \ j \neq k$,*

where for a given set Δ, $actions(\Delta) = \{\mathbf{E}(a) \in \Delta$ such that $a \in Actions\}$. We will call x the initiator *of $\mathcal{D}$.*

Thus, in an argumentation dialogue, the agents focus on a specific goal (1). They do not exchange purely "dialogical" arguments, but genuine products of their own reasoning based on the knowledge available to them. In particular, we require that circumstances/actions stated are supported by the uttering agent (2-*i*/*ii*), and for those denied the agent is able to produce an attacking argument based on the goal subject of the dialogue (2-*iii*/*iv*). Finally, we require that an agent does not utter the same performative twice (3). In this way, at each step j, the dialogue develops by an agent reasoning on the state at step k, for some $k < j$, to propose a new state to the receiver. Dialogue moves need not directly address the previous move, but are free to refer to moves uttered in the past, in the course of the same dialogue. This leaves agents free to try several alternative arguments, so that the dialogue can proceed even if an agent does not have an answer to the last move.

One can easily see that, given a finite number of ground arguments, dialogues will always finite length [18]. However, we are interested here not only in dialogues that terminate, but especially we want to be able to define what dialogues are "fruitful." We will then focus on the notion of agreement:

Definition 14 (Agreement between two agents). *Given a multiagent argumentation framework $\mathcal{M}$, an* agreement *between two agents $x, y \in \mathcal{M}$ about a goal G_x is a set $\mathcal{C}$ such that there exists an* argumentation dialogue $\mathcal{D} = \{p_0, p_1, \ldots\}$ *between x and y about G_x, whose state(D, j) is such that $\Psi_j^{aa} = \mathcal{C}$ for some j.*

In other words, we say that two agents reach an agreement when they come up in the course of the same dialogue with a set $\mathcal{C}$ which contains the same actions. By definition of argumentation dialogue, they are supported by the same arguments (circumstances) from both sides.

This formulation of argumentation dialogue makes it possible to prove some important properties of the framework, which to the best of our knowledge are not to be found in other multiagent argumentation frameworks.

Proposition 1. *Given an argumentation dialogue $\mathcal{D}$ and a performative $p \in \mathcal{D}$, argument(p) is a conflict-free set of arguments.*

Proof. By Definition 13, $\forall p \; \exists \; \Delta, \mathcal{S},$ and $\mathcal{G}$ such that $\mathcal{S} \vDash_{\Delta} \mathcal{G}$ and *argument*$(p) \subseteq \Delta$. Thus Proposition 1 follows from Corollary 2.

Proposition 2. *Given an argumentation dialogue $\mathcal{D}$ and a performative $p \in \mathcal{D}$:*

1. *if locution$(p) \in \{$'state circumstances', 'state actions'$\}$, then argument(p) is an admissible set of arguments for utterer(p);*
2. *if locution$(p) \in \{$'deny circumstances', 'deny actions'$\}$, then argument(p) is an admissible set of arguments for receiver(p);*

Proof. The proof follows from Corollary 3 and from Definition 13.

Proposition 3. *Every agreement $\mathcal{C}$ between two agents x and y about a goal G_x, is an admissible set of arguments for both x and y.*

Proof. If $\mathcal{C} = \emptyset$, Proposition 3 follows from Definition 10, which implies that the empty set is always admissible. If $\mathcal{C} \neq \emptyset$, by Definition 14 there exists an *argumentation dialogue* $\mathcal{D} = \{p_0, p_1, \ldots\}$ between x and y about G_x, whose $state(D, j)$ is such that $\Psi_j^{aa} = \mathcal{C}$ for some j. By Corollary 5 it is a structural property of $state(\mathcal{D}, j)$ that $\Psi_j^{aa} = \Psi_j^{sa}$, and by Definition 12 $\exists l, k$ such that $\Psi_j^{sa} = argument(p_l) = argument(p_k)$ and $locution(p_l) = locution(p_k) = `state\,actions'$ for some $l, k \leq j$. It thus follows from Proposition 2 that Ψ_j^{sa} is admissible for both x and y.

We believe that this is a very important property of $\mathcal{S}$CIFF-AF. If two agents reach what we call an agreement during a dialogue, it is important that such an agreement identifies a possible future system development which is admissible by both – which is the case here. In this way, agents can step through agreements and thus develop plans for future courses of action which ensure a consistent system evolution.

4 An example of an argumentation dialogue leading to an agreement using SCIFF-AF

In order to illustrate the usage of the $\mathcal{S}$CIFF-AF framework and its properties, we propose as a scenario an adaptation of Rahwan & Amgoud's conference example [14]:

Example 1. A scientist s (based in the UK) wishes to attend a $conf$erence. Prior to his departure, however, he needs to reach a preliminary agreement with his department d. s knows that $conf$ is in Liverpool, and that the fee can be 400 (on-site) or 200 (early), that a limo is a comfortable car, and that Liverpool is a far but domestic destination. s has some constraints: he knows that if he wishes to attend a conference, then he must reach the place of the conference, and pay the fee. If he wishes to reach a place, he must either fly or drive. In addition, if he wishes to reach a place, either it is not a domestic destination, and he does not want to fly economy, nor he wants to drive; or it is a far destination, and in that case he does not want to drive; or else he wants to rent a comfortable car. s's department, d, has a number of constraints. If one wants to reach a destination and pay a conference fee, then he must attend the conference; the fee must be lower than 300, or else it is not permitted to rent a limo, nor to fly business, or else it is a domestic destination, and then it is not permitted to fly business.

Given such a scenario, a possible argumentation dialogue that we would like to obtain in this framework could be the following:

1. (s): I wish to attend a conference ($conf$).
2. (d): I am listening.

$\mathcal{A}$ $\{early, on\_site\} \cup \mathbf{EXP}$
$Actions$ $\{reach, fly, drive, pay, buy\_ticket, rent\_car\}$

$\mathcal{G}_s$ $\{\mathbf{E}(attend(conf))\}$
$\mathcal{IC}_s$ $\mathbf{E}(attend(Conf)) \rightarrow conference(Conf, Venue, Fee) \ \wedge \ \mathbf{E}(reach(Venue))$
$\wedge \ \mathbf{E}(pay(Conf, Fee))$.
$\mathbf{E}(reach(Dest)) \rightarrow \mathbf{E}(fly(Dest)) \ \vee \ \mathbf{E}(drive(Dest))$.
$\mathbf{E}(reach(Dest)) \rightarrow non\,domestic(Dest) \ \wedge \ \mathbf{EN}(buy\_ticket(Dest, economy))$
$\wedge \ \mathbf{EN}(drive(Dest))$
$\vee \ far(Dest) \ \wedge \ \mathbf{EN}(drive(Dest))$
$\vee \ domestic(Dest) \ \wedge \ \mathbf{E}(rent\_car(Dest, Car)) \ \wedge \ comfortable(Car)$.
$\mathbf{E}(fly(Dest)) \rightarrow \mathbf{E}(buy\_ticket(Dest, economy)) \ \vee \ \mathbf{E}(buy\_ticket(Dest, business))$.
$\mathbf{E}(drive(Dest)) \rightarrow \mathbf{E}(rent\_car(Dest, sedan)) \ \vee \ \mathbf{E}(rent\_car(Dest, limo))$.
$\mathbf{E}(fly(Dest)) \ \wedge \ \mathbf{E}(drive(Dest)) \rightarrow \bot$.
$\mathbf{E}(fly(Dest)) \rightarrow \mathbf{EN}(rent\_car(Dest, Car))$.
$\mathbf{E}(drive(Dest)) \rightarrow \mathbf{EN}(buy\_ticket(Dest, Class))$.
$early \ \wedge \ on\_site \rightarrow \bot$.
$\mathcal{P}_s$ $conference(conf, lvp, Fee) \leftarrow (on\_site \ \wedge \ Fee = 400) \ \vee \ (early \ \wedge \ Fee = 200)$.
$comfortable(limo)$.
$far(lvp)$.
$domestic(lvp)$.

$\mathcal{IC}_d$ $\mathbf{E}(reach(Dest)) \ \wedge \ \mathbf{E}(pay(Conf, Fee)) \rightarrow \mathbf{E}(attend(Conf)) \ \wedge \ Fee < 300$
$\vee \ \mathbf{EN}(rent\_car(Dest, limo)) \ \wedge \ \mathbf{EN}(buy\_ticket(Dest, business))$
$\wedge \ domestic(Dest) \ \vee \ \mathbf{EN}(buy\_ticket(Dest, business))$.
$\mathbf{E}(buy\_ticket(Dest, business)) \ \wedge \ \mathbf{E}(buy\_ticket(Dest, economy)) \rightarrow \bot$.
$\mathcal{P}_d$ $domestic(lvp)$.

Fig. 1. $\mathcal{S}$CIFF programs of scientist (s) and department (d)

3. (s): There are some circumstances I wish to bring to your attention. I do not want to drive there. So I will not rent a car. Also, I think I have to pay on-site registration.
4. (s): I was thinking I can do the following: buy a business plane ticket, fly and reach Liverpool, pay 400 as a fee.
5. (d): You are not allowed to fly business class!
6. (s): I take your point.
7. (s): I can fly economy.
8. (d): Agreed. Go ahead.

Figure 1 shows its possible implementation in the $\mathcal{S}$CIFF-AF. Note that, in addition to its formulation given above, some additional domain-specific constraints are specified: in order to fly one must either buy an economy ticket or a business ticket, one does never want to fly and drive at the same time, etc. Note also that s does not know which one is the fee he has to pay, so it considers *early* and *on_site* to be an abducible atoms belonging to $\mathcal{A}$. s's goal, ICs and goal are denoted by $\mathcal{G}_s$, $\mathcal{IC}_s$ and $\mathcal{P}_s$; similarly for d.

$p_0 : tell(s, d, `enter\ dialogue', \{\mathbf{E}(attend(conf))\})$.
$p_1 : tell(s, d, `turn\ finished')$.
$p_2 : tell(d, s, `enter\ dialogue', \{\mathbf{E}(attend(conf))\})$.
$p_3 : tell(d, s, `turn\ finished')$.
$p_4 : tell(s, d, `state\ circumstances', Terms_1)$.
$p_5 : tell(s, d, `state\ actions', Actions_1)$.
$p_6 : tell(s, d, `turn\ finished')$.
$p_7 : tell(d, s,' deny\ actions', \mathbf{E}(buy\_ticket(lvp, business)))$
$p_8 : tell(d, s, `turn\ finished')$
$p_9 : tell(s, d,' accept\ denial')$
$p_{10} : tell(s, d, `state\ circumstances', Terms_2)$.
$p_{11} : tell(s, d, `state\ actions', Actions_2)$.
$p_{12} : tell(s, d, `turn\ finished')$
$p_{13} : tell(d, s,' state\ actions', Action_2)$
$p_{14} : tell(d, s, `turn\ finished')$
$p_{15} : tell(s, d,' leave\ dialogue')$
$p_{16} : tell(d, s,' leave\ dialogue')$
$Actions_1 = \{\ \mathbf{E}(reach(lvp)), \mathbf{E}(pay(conf, 400)), \mathbf{E}(buy\_ticket(lvp, business)), \mathbf{E}(fly(lvp))\ \}$
$Terms_1 = \{\ on\_site, not\ early, \mathbf{EN}(rent\_car(lvp, Car)), \mathbf{EN}(drive(lvp)), not\ \mathbf{E}(drive(lvp))\ \}$
$Actions_2 = \{\ \mathbf{E}(reach(lvp)), \mathbf{E}(pay(conf, 400)), \mathbf{E}(buy\_ticket(lvp, economy)), \mathbf{E}(fly(lvp))\ \}$
$Terms_2 = \{\ on\_site, not\ early, \mathbf{EN}(buy\_ticket(lvp, business)), \mathbf{EN}(rent\_car(lvp, Car)),$
$\mathbf{EN}(drive(lvp)), not\ \mathbf{E}(drive(lvp)), \mathbf{EN}(drive(lvp)), not\ \mathbf{E}(drive(lvp))\ \}$

Fig. 2. Sample argumentation dialogue between s and d about $G_s = \mathbf{E}(attend(conf))$

The agents can engage in argumentation dialogues to find a possible future evolution upon which both agree. One such dialogue is shown in Figure 2. At each step, the dialogue complies with Definition 13, and it therefore produces a result which (i) is consistent with both constraints, and (ii) is such that in the end both agree on the present/future circumstances. In fact, the dialogue ends with an agreement ($Actions_2$).

Thanks to the result enunciated in Proposition 3, we know that $Actions_2$ is indeed an admissible set of arguments for both s and d.

5 Conclusion and Future Work

The main contribution of this paper is the illustration of some fundamental properties of a declarative framework for multiagent reasoning and dialogue-based argumentation about actions ($\mathcal{S}$CIFF-AF), initially proposed in [19].

$\mathcal{S}$CIFF-AF is equipped with a sound operational model, an admissible sets semantics, a notion of (argumentation) dialogue and a notion of agreement about actions. Thanks to these properties, it is possible to accommodate in $\mathcal{S}$CIFF-AF a declarative representation of the agent knowledge, upon which agents can reason, and interact by argumentation dialogues.

Although agent reasoning is not covered by this work, Alberti *et al.* have proposed in [1] an agent architecture in which the reasoning activity of agents is

type of goal	type of expectation
positive	**E**
negative	**EN**
in abeyance	*not* **E** $\wedge$ *not* **EN**

Fig. 3. Mapping between types of goals and types of expectations

based on $\mathcal{S}$CIFF, so we have ground to believe that $\mathcal{S}$CIFF-AF can be actually used as a concrete, operational multiagent argumentation framework.

The operational nature of $\mathcal{S}$CIFF-AF is maybe one of its main distinguishing features, compare to other existing work. Argumentation dialogues are useful because through them agents may eventually reach mutual agreements, which they can directly use, for example by adopting them as possible future internal goals. Importantly, in this article we have demonstrated that $\mathcal{S}$CIFF-AF is grounded on a solid formal basis, which includes a number of results about its relation with Dung's abstract argumentation framework.

This work builds on previous results on abstract argumentation frameworks [6], on the $\mathcal{S}$CIFF proof-procedure [2], on computing arguments in ALP [11], and on multi-agent dialogue framework [17, 5], as cited in the text. In the future, we intend to investigate more thoroughly the formal relations between $\mathcal{S}$CIFF-AF and other argumentation frameworks. investigate other forms of argumentation and the enrichment of $\mathcal{S}$CIFF-AF by introducing a notion of value. We also intend to investigate the relation between $\mathcal{S}$CIFF-AF's notion of positive/negative expectations and Amgoud & Kaci's work about generation of bipolar goals [3]. In [3] goals are partitioned into three categories: *positive* goals, *negative* goals, and goals in *abeyance*. If positive goals reward the agent that satisfies them, negative goals are on the contrary those considered unacceptable, while goals in abeyance just mirror what is not rejected, although they do not really reward the agent that adopts them. We think that the $\mathcal{S}$CIFF-AF metaphor of expectations applies smoothly to this understanding of goals. One obvious relation among the two paradigms is shown in Figure 3.

Beside the very similar understanding of goals/expectations, Amgoud & Kaci's framework and its recent refinement by Rahwan & Amgoud [14] do have many motivations in common with this work. We plan to investigate these aspects in depth in the future. Some possible interesting extensions of the $\mathcal{S}$CIFF-AF framework could be a notion of attack that accommodates a *priority degree*, and a more comprehensive argumentation setting in which agents argue using not only atomic entities, but also implications (i.e., integrity constraints, or conditional rules).

Another aspect worth investigating is that of knowledge representation, for example to distinguish between *explanatory arguments*, used to provide reasons of adopting goals, beliefs or disbeliefs, and *instrumental arguments*, used to present plans to achieve goals [3, 14].

Acknowledgements

The author thanks Federico Chesani, Marco Gavanelli and Francesca Toni for their valuable comments on earlier versions of this work, and the anonymous referees for their valuable suggestions. This research has been partially supported by the MIUR PRIN 2005 project No 2005-011293, *Specification and verification of agent interaction protocols*,[3] and by the National FIRB project *TOCAI.IT*,[4].

References

1. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, and P. Mello. A verifiable logic-based agent architecture. In *Foundations of Intelligent Systems. 16th International Symposium, ISMIS 2006, Bari, Italy, September 27-29, 2006. Proceedings*, volume 2870 of *Lecture Notes in Computer Science*, pages 338–343. Springer-Verlag, 2006.
2. M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Verifiable agent interaction in abductive logic programming: the SCIFF framework. *ACM Transactions On Computational Logic (TOCL)*, 8, 2007.
3. L. Amgoud and S. Kaci. On the generation of bipolar goals in argumentation-based negotiation. In I. Rahwan, P. Moraitis, and C. Reed, editors, *ArgMAS 2004*, volume 3366 of *Lecture Notes in Artificial Intelligence*, pages 192–207. Springer-Verlag, 2005.
4. L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue and negotiation. In W. Horn, editor, *Proceedings of the Fourteenth European Conference on Artificial Intelligence, Berlin, Germany (ECAI 2000)*. IOS Press, Aug. 2000.
5. K. Atkinson, T. Bench-Capon, and P. McBurney. A dialogue game protocol for multi-agent argument over proposals for action. *Journal of Autonomous Agents and Multi-Agent Systems*, 11:153–171, 2005.
6. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
7. T. H. Fung and R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, Nov. 1997.
8. M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the 6th National Conference on Artificial Intelligence, AAAI'87*, pages 677–682, Seattle, WA, USA, July 1987. Morgan Kaufmann Publishers.
9. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.
10. A. C. Kakas and P. Moraïtis. Argumentation based decision making for autonomous agents. Technical report, Department of Computer Science, University of Cyprus, 2002.
11. A. C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9(4):515–562, 1999.
12. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 2nd extended edition, 1987.

[3] `http://www.ricercaitaliana.it/prin/dettaglio_completo_prin_en-2005011293.htm`

[4] `http://www.dis.uniroma1.it/~tocai/`

13. P. McBurney, S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part I*, pages 402–409, Bologna, Italy, July 15–19 2002. ACM Press.
14. I. Rahwan and L. Amgoud. An argumentation-based approach for practical reasoning. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006), Hakodate, Hokkaido, Japan*. ACM Press, 2006.
15. I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. *The Knowledge Engineering Review*, 18:343–375, 2003.
16. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers, Apr. 1991.
17. F. Sadri, F. Toni, and P. Torroni. Logic agents, dialogues and negotiation: an abductive approach. In *Proceedings AISB'01 Convention, York, UK*, Mar. 2001.
18. P. Torroni. A study on the termination of negotiation dialogues. In C. Castelfranchi and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002), Part III*, pages 1223–1230, Bologna, Italy, July 15–19 2002. ACM Press.
19. P. Torroni. Multi-agent agreements about actions through argumentation. In P. Dunne and T. Bench-Capon, editors, *Computational Models of Argument*, volume 144 of *Frontiers in Artificial Intelligence and Application*, pages 323–328. IOS Press, 2006.

Author Index

Alferes, José Júlio 64
Almeida, Iara 64
Atkinson, Katie 1

Baroni, Pietro 33
Bench-Capon, Trevor J.M. 1

Chesñevar, Carlos Iván 17
Cortés, Ulises 114

Dung, Phan Minh 49

Gaertner, Dorian 80
García, Alejandro 17
Giacomin, Massimiliano 33

Nieves, Juan Carlos 114

Osorio Galindo, Mauricio 114

Pereira, Luís Moniz 96
Pinto, Alexandre Miguel 96

Rotstein, Nicolás Daniel 17

Simari, Guillermo R. 17

Thang, Phan Minh 49
Toni, Francesca 80
Torroni, Paolo 125

Wyner, Adam 1