Chapter 6
Proof Theories and Algorithms for Abstract Argumentation Frameworks

Sanjay Modgil and Martin Caminada

1 Introduction

Previous chapters have focussed on abstract argumentation frameworks and properties of sets of arguments defined under various extension-based semantics. The main focus of this chapter is on more procedural, proof-theoretic and algorithmic aspects of argumentation. In particular, Chapter 2 describes properties of extensions of a Dung argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ under various semantics. In this context a number of questions naturally arise:

1. For a given semantics s, "global" questions concerning the existence and construction of extensions can be addressed:
   a. Does an extension exist?
   b. Give an extension (it does not matter which, just give one)
   c. Give all extensions.
2. For a given semantics s, "local" questions concerning the existence and construction of extensions, relative to a set $A \subseteq \mathcal{A}$ can be addressed. Note that it is often the case that $|A| = 1$, in which case the member of A is called the query argument.
   a. Is A contained in an extension ? (Credulous membership question.)
   b. Is A contained in all extensions ? (Sceptical membership question.)
   c. Is A attacked by an extension?
   d. Is A attacked by all extensions?
   e. Give an extension containing A.

Sanjay Modgil
Department of Computer Science, King's College London, e-mail: sanjay.modgil@kcl.ac.uk

Martin Caminada
Interdisciplinary Lab for Intelligent and Adaptive Systems, University of Luxembourg e-mail: martin.caminada@uni.lu

I. Rahwan, G. R. Simari (eds.), *Argumentation in Artificial Intelligence,*
DOI 10.1007/978-0-387-98197-0_6,

f. Give all extensions containing A.
g. Give an extension that attacks A.
h. Give all extensions that attack A.

In this chapter, procedures will be described for answering a selection of the above questions with respect to finite argumentation frameworks $\langle \mathcal{A}, \mathcal{R} \rangle$ (in which $\mathcal{A}$ is finite). Notice that for some semantics, such as the grounded and preferred semantics, extensions always exist, so that 1a will be answered in the positive for any framework. Also, for the grounded semantics, at most one extension exists, so that questions distinguished by reference to 'an' or 'all' extensions are equivalent (e.g., questions 2a and 2b).

Sections 2 and 3 will introduce some key concepts underpinning the approaches that we will use in the description of proof theories and algorithms. Sections 4 - 6 will then focus on application of these approaches to the core semantics defined by Dung [13]; namely grounded, preferred and stable.

Broadly speaking, two approaches will be presented. Firstly, Section 2 formally describes the argument graph *labelling* approach that was originally proposed by Pollock [23], and has more recently been the subject of renewed analysis and investigation [5, 6, 25, 27]. The basic idea is that the status assignment to arguments defined by the extension-based approach (see Chapter 2), can be directly defined through assignment of labels to the arguments (nodes) in the framework's corresponding argument graph. Section 2 provides formal underpinnings for the definition of argument graph labelling algorithms that are used to address a selection of the above questions in Sections 4 - 6.

Section 3 then describes a framework for *argument game* based proof theories [9, 16, 17, 28]. The inherently dialectical nature of argumentation lends itself to formulation of argument games in which a proponent starts with an initial argument to be tested, and then an opponent and the proponent successively attack each other's arguments. The initial argument provably has a certain status if the proponent has a winning strategy whereby he can win irrespective of the moves made by the opponent. In Sections 4 - 6 we describe specific games, emphasising the way in which the rules of each specific game correspond to the semantics they are meant to capture.

2 Labellings

In this section the labelling approach (based on its formulation in [5, 6]) is briefly reviewed. Given an argumentation framework $AF = \langle \mathcal{A}, \mathcal{R} \rangle$, a labelling assigns to each argument exactly one label, which can be either `IN`, `OUT` or `UNDEC`. The label `IN` indicates that the argument is justified, `OUT` indicates that the argument is overruled, and `UNDEC` indicates that the status of the argument is undecided.

Definition 6.1. Let $\langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework.

- A labelling is a total function $\mathcal{L} : \mathcal{A} \mapsto \{\texttt{IN}, \texttt{OUT}, \texttt{UNDEC}\}$

- We define: $\texttt{in}(\mathcal{L}) = \{x | \mathcal{L}(x) = \texttt{IN}\}$; $\texttt{out}(\mathcal{L}) = \{x | \mathcal{L}(x) = \texttt{OUT}\}$; $\texttt{undec}(\mathcal{L}) = \{x | \mathcal{L}(x) = \texttt{UNDEC}\}$

Notice that from hereon, we may represent a labelling $\mathcal{L}$ as a triple of the form $(\texttt{in}(\mathcal{L}), \texttt{out}(\mathcal{L}), \texttt{undec}(\mathcal{L}))$.

We now define what it is for an argument to be assigned a legal labelling:

Definition 6.2. Let $\mathcal{L}$ be a labelling for $\langle \mathcal{A}, \mathcal{R} \rangle$ and $x \in \mathcal{A}$

- x is legally IN iff x is labelled IN and every y that attacks x ($y\mathcal{R}x$) is labelled OUT
- x is legally OUT iff x is labelled OUT and there is at least one y that attacks x and y is labelled IN
- x is legally UNDEC iff x is labelled UNDEC, there is no y that attacks x such that y is labelled IN, and it is not the case that: for all y, y attacks x implies y is labelled OUT.

The rules defining legal labelling assignments encode one's intuitive understanding of the status assignments defined by the extension-based semantics and their use of the reinstatement principle, as described in Chapter 2. An argument x is IN only if all its attackers are OUT, and each attacker is OUT only if it is itself attacked by an argument that is IN. Thus, the arguments that are IN in a legal labelling correspond to a single extension. It is sometimes not possible to obtain a labelling where each argument is either legally IN or legally OUT; consider for example an argumentation framework with just a single argument that attacks itself. This is why we need a third label UNDEC, which basically means that there is insufficient ground to explicitly justify the argument and insufficient ground to explicitly overrule the argument. Notice that from Definition 6.2 it follows that x is legally UNDEC iff it is labelled UNDEC, and at least one y that attacks x is labelled UNDEC, and no y attacking x is labelled IN.

Definition 6.3. For $\texttt{l} \in \{\texttt{IN}, \texttt{OUT}, \texttt{UNDEC}\}$ an argument x is said to be illegally l iff x is labelled l, and it is not legally l.

- An admissible labelling $\mathcal{L}$ is a labelling without arguments that are illegally IN and without arguments that are illegally OUT.
- A complete labelling $\mathcal{L}$ is an admissible labelling without arguments that are illegally UNDEC

Notice that the additional requirement on complete labellings corresponds intuitively to Chapter 2's characterisation of a complete extension as a fixed point of a framework AF's characteristic function $\mathcal{F}_{AF}$. Since the grounded and preferred extensions of a framework are the minimal, respectively maximal, fixed points (complete extensions) of a framework, then as one would expect, grounded and preferred labellings are given by complete labellings that minimise, respectively maximise, the arguments that are made legally IN. A stable labelling is a complete labelling in which all arguments are either legally IN or legally OUT, and hence no argument is UNDEC.

Definition 6.4. Let $\mathcal{L}$ be a complete labelling. Then:

- $\mathcal{L}$ is a grounded labelling iff there there does not exist a complete labelling $\mathcal{L}'$ such that $\texttt{in}(\mathcal{L}') \subset \texttt{in}(\mathcal{L})$ [1]
- $\mathcal{L}$ is a preferred labelling iff there there does not exist a complete labelling $\mathcal{L}'$ such that $\texttt{in}(\mathcal{L}') \supset \texttt{in}(\mathcal{L})$
- $\mathcal{L}$ is a stable labelling iff $\texttt{undec}(\mathcal{L}) = \emptyset$

In [6], the following theorem is shown to hold:

Theorem 6.1. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework, and $E \subseteq \mathcal{A}$. For $s \in \{$admissible, complete, grounded, preferred, stable$\}$:*

E is an s extension of AF iff there exists an s labelling $\mathcal{L}$ with $\texttt{in}(\mathcal{L}) = E$ [2]

In Sections 4 - 6 we will describe algorithms that compute labellings and so address a subset of the questions enumerated in Section 1. We conclude this section with an example:

Example 6.1. Consider the framework in Figure 6.1. There exists three complete labellings: 1. ($\emptyset$, $\emptyset$, $\{a,b,c,d,e\}$); 2. ($\{a\}$, $\{b\}$, $\{c,d,e\}$); and 3. ($\{b,d\}$, $\{a,c,e\}$, $\emptyset$). 1 is the grounded labelling, 2 and 3 are preferred, and 3 is also stable.

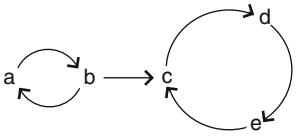

Fig. 6.1 An argumentation framework

3 Argument Games

In general, proof theories license the way in which pieces of information can be articulated in order to prove a fact. They therefore provide a basis for algorithm development, and proofs constructed according to these theories provide explanations as to why a given fact is believed to be true. For example, a proof that argument x is in an admissible extension, would consist of showing *how* one can establish the existence of such an extension, rather than simply identifying the extension. Intuitively,

[1] Since every framework has a unique minimal fixed point, one could alternatively define $\mathcal{L}$ to be a grounded labelling iff for each complete labelling $\mathcal{L}'$ it holds that $\texttt{in}(\mathcal{L}) \subset \texttt{in}(\mathcal{L}')$

[2] Note that for $s \neq$ admissible there is a 1-1 mapping between s extensions and s labellings. An admissible extension may have more than one admissible labelling. For example, the admissible extension $\{c\}$, of $c \rightarrow \underline{b}$, $\underline{c} \rightarrow a$, has two admissible labellings: $(\{c\},\{b\},\{a\})$ and $(\{c\},\{b,a\},\emptyset)$.

one would need to show how to *defend* x by showing that for every argument y that is put forward (moved) as an attacker of x, one must move an argument z that attacks y, and then subsequently show how any such z can be reinstated against attacks (in the same way that z reinstates x). The arguments moved can thus be organised into a graph of attacking arguments that constitutes an explanation as to why x is in an admissible extension.

The *process* of moving arguments and counter-arguments can be implemented as an algorithm [27]. In this chapter we follow the approach of [9, 14, 16, 17, 26, 28] and present the moving of arguments as 2-person dialogue games that provide a natural way in which to lay out and understand the algorithms that implement them. To be sure, the actual algorithms themselves, should, except for didactic purposes, not be implemented as dialogue games, but rather as monological procedures (or *methods* in OO-languages) that are called recursively.

A dialogue game is played by two players, PRO (for "proponent") and OPP (for "opponent"), each of which are referred to as the other's 'counterpart'. A game begins with PRO moving an initial argument x that it wants to put to the test. OPP and PRO then take turns in moving arguments that attack their counterpart's last move. From hereon:

> a sequence of moves in which each player moves against its counterpart's last move is referred to as a *dispute*.

If the last move in a dispute is by player Pl, and Pl's counterpart cannot respond to this last move, then Pl is said to win the dispute. If a dispute with initial argument x is won by PRO, we call the dispute a *line of defense* for x.

The rules of the game encode restrictions on the legality of moves in a dispute, and different sets of rules capture the different semantics under which justification of the initial argument x is to be shown, by effectively establishing when OPP or PRO run out of legal moves. In general, however, a player can backtrack to a counterpart's previous move and initiate a new dispute. Consider the dispute $a_{PRO} - b_{OPP} - c_{PRO} - d_{OPP} - e_{PRO} - f_{OPP}$ won by OPP ($x_{Pl} - y_{Pl'}$ denotes player Pl' moving argument y against counterpart Pl's argument x). PRO must then try and backtrack to move an argument against either b_{OPP} or d_{OPP} and establish an alternative line of defense for a. Suppose such a line of defense $a_{PRO} - b_{OPP} - g_{PRO}$. Then OPP can backtrack and try an alternative line of attack moving h against a, so that PRO must now try and win the newly initiated dispute $a_{PRO} - h_{OPP}$. Thus, the 'playing field' of a game — the data structure on the basis of which argument games are played — can be represented by an argumentation framework's induced *dispute tree*, in which every branch from root to leaf is a dispute:

Definition 6.5. Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be an argumentation framework, and let $a \in \mathcal{A}$. The dispute tree induced by a in AF is a tree T of arguments, such that T's root node is a, and $\forall x, y \in \mathcal{A}$: x is a child of y in T iff $x \mathcal{R} y$.

Figure 6.2i) shows an argumentation framework, and part of the tree induced by a is shown in Figure 6.2ii). Notice that multiple instances of arguments are individuated by numerical indicies. Any game played by PRO and OPP in which PRO

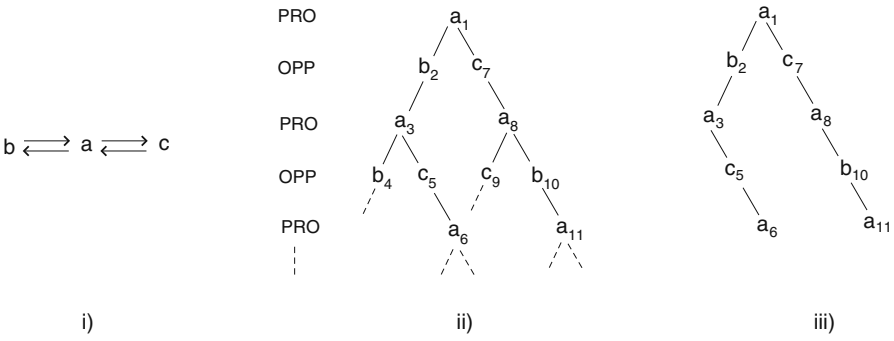

Fig. 6.2 i) shows an argumentation framework, and ii) shows the dispute tree induced in *a*. iii) shows the dispute tree induced under the assumption that OPP cannot repeat moves in the same dispute (branch of the tree)

attempts to show that *a* is justified, must necessarily involve the submission of attacking arguments conforming to some sub-tree of the induced tree in Figure 6.2ii). In particular, PRO must show that it fully fulfills its burden of proof, in response to OPP who fully fulfills its burden of attack. In other words, OPP moves *all* ys that attack an x moved by PRO, and each such y must in turn be responded to by PRO moving *at least one* x' that attacks y. This does of course capture the reinstatement principle used to define the extensions of an argumentation framework, and correlates with Section 2's definition of legal labellings and the extensions they define[3]. In the context of a game, this is captured by the notion of a *winning strategy* for an argument. Notice that in the following definition we refer to the notion of a *sub-dispute* d' of a dispute d, which, intuitively is any sub-sequence of d that starts with the same initial argument as d.

Definition 6.6. Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$, T the dispute tree induced by a in AF, and T' a sub-tree of T. Then T' is a winning strategy for a iff:

1. The set $D_{T'}$ of disputes in T' is a non-empty finite set such that each dispute $d \in D_{T'}$ is finite and is won by PRO (terminates in an argument moved by PRO)
2. $\forall d \in D_{T'}$, $\forall d'$ such that d' is some sub-dispute of d and the last move in d' is an argument x played by PRO, then for any y such that $y \mathcal{R} x$, there is a $d'' \in D_{T'}$ such that $d' - y_{OPP}$ is a sub-dispute of d''.

If PRO plays moves as described in a winning strategy sub-tree, then PRO is guaranteed to win.

As stated earlier, the rules of a game encode restrictions on the arguments a player can legally move in a dispute in order to attack its counterpart's argument. These restrictions vary according to the semantics of interest, and are encoded in a legal move function:

[3] Recall that x is legally `IN` iff *all* ys that attack an x are legally `OUT`, and each such y is legally `OUT` iff there is *at least one* x' attacking y that is legally `IN`

Definition 6.7. Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$, T the dispute tree induced by a in AF. Let D_T be the set of all disputes in T. Then ϕ is a legal move function such that $\phi : D_T \mapsto 2^{\mathcal{A}}$.

Given a dispute tree T induced by a, the legal move function ϕ for a semantics s, prunes T to obtain the sub-tree T' of T that we call the ϕ tree induced by a. T' is the playing field of the game for semantics s. Thus, we define a *ϕ-winning strategy for a* [9, 17] as a sub-tree of the ϕ dispute tree induced by a, in the same way as Definition 6.6, except that we replace 'for any x such that $x\mathcal{R}y$' in condition 2, with 'for any x that OPP can ϕ legally move against y'. Intuitively, ϕ is defined such that a is in an admissible extension that conforms to the semantics s iff there is a ϕ-winning strategy for a in the ϕ tree induced by a, where the arguments moved by PRO in the ϕ-winning strategy are conflict free (recall that an admissible extension must contain no arguments that attack each other).

For example, consider games whose legal move function ϕ prohibits OPP from repeating arguments in the same dispute. Figure 6.2iii) shows the *ϕ-dispute tree* that is a sub-tree of the *dispute tree* induced by a (Figure 6.2ii)). After PRO plays a_6, OPP cannot backtrack and extend the dispute $d = a_1 - b_2 - a_3$ by moving b against a_3, since b has already been moved by OPP in d. Similarly, OPP cannot backtrack to move c against a_8 in order to extend $d' = a_1 - c_7 - a_8$. Note also that both $D_{T_1} = \{d_1 = a_1 - b_2 - a_3 - c_5 - a_6\}$ and $D_{T_2} = \{d_2 = a_1 - c_7 - a_8 - b_{10} - a_{11}\}$ are winning strategies. In the former case, consider the sub-dispute $d'_1 = a_1 - b_2 - a_3$ of d_1. OPP can legally move c against a_3, but there is a dispute in D_{T_1} that extends d'_1 (d_1 itself) in which PRO moves against OPP's move of c.

To summarise, suppose PRO wishes to show that x is a member of an extension E under the semantics s. The associated legal move function ϕ for s defines some ϕ-dispute tree T that is a sub-tree of the dispute tree induced by x, and defines all possible disputes the players can play in a game. The ϕ-dispute tree T should be such that: $x \in E$ iff there is a ϕ-winning strategy T' in T, such that the arguments moved by PRO in T' do not attack each other (are conflict free). A ϕ-winning strategy is a set of disputes won by PRO in which PRO has fulfilled its burden of proof by countering all possible ϕ-legal moves of OPP.

4 Grounded Semantics

For any argumentation framework, there is guaranteed to be exactly one grounded extension. Hence, questions 1b, 1c and 2a - 2h can all be addressed by construction of a framework's grounded extension. In Section 4.1 we present an algorithm that generates the grounded labelling of an argumentation framework. Section 4.2 then describes an argument game for deciding whether a given argument is in the grounded extension, thus providing an alternative way for addressing the questions 2a and 2b.

The grounded semantics places the highest burden of proof on membership of the extension that it defines. This equates with Chapter 2's definition of the extension as the *least* fixed point of a framework AF's characteristic function $\mathcal{F}_{AF}$ (i.e., the smallest admissible E that contains exactly those arguments that are acceptable

w.r.t. E). The extra burden of proof is intuitively captured by the fact that in defending x's membership of the grounded extension E, one must 'appeal to' some argument other than x itself. That is to say, for any y such that y attacks x, y is attacked by at least one $z_1 \in E$ such that $z_1 \neq x$, and in turn, z_1 must be reinstated against any attack, by some $z_2 \in E$ such that $z_2 \neq x$, $z_2 \neq z_1$, and so on. This property is exploited by both the algorithm for generating the grounded labelling, and argument games for the grounded semantics. The property is relatively straightforward to show given Chapter 2's description of how, starting with the empty set, iteration of the characteristic function yields the grounded extension. We have that $x \in \mathcal{F}^i_{AF}$ iff for every attack on x, x is reinstated by some $z \in \mathcal{F}^j_{AF}$, where $z \neq x$ and $j < i$.

4.1 A labelling algorithm for the Grounded Semantics

An algorithm for generating the grounded labelling starts by assigning IN to all arguments that are not attacked, and then iteratively: OUT is assigned to any argument that is attacked by an argument that has just been made IN, and then IN to those arguments *all* of whose attackers are OUT. Thus, the arguments assigned IN on each iteration, are those that are reinstated by the arguments assigned IN on the previous iteration. The iteration continues until no more new arguments are made IN or OUT. Any arguments that remain unlabelled are then assigned UNDEC. One can straightforwardly show that the algorithm is sound and complete since it effectively mimics construction of the grounded extension through iteration of a framework's characteristic function. The algorithm for generating the grounded labelling $\mathcal{L}_G$ of a framework $\langle \mathcal{A}, \mathcal{R} \rangle$ is presented more formally below, in which we use Section 2's representation of a labelling $\mathcal{L}$ as a triple ($\mathtt{in}(\mathcal{L})$, $\mathtt{out}(\mathcal{L})$, $\mathtt{undec}(\mathcal{L})$).

Algorithm 6.1 Algorithm for Grounded Labelling

1: $\mathcal{L}_0 = (\emptyset,\emptyset,\emptyset)$
2: **repeat**
3: $\quad \mathtt{in}(\mathcal{L}_{i+1}) = \mathtt{in}(\mathcal{L}_i) \cup \{x \mid x$ is not labelled in $\mathcal{L}_i$, and $\forall y$: if $y \mathcal{R} x$ then $y \in \mathtt{out}(\mathcal{L}_i)\}$
4: $\quad \mathtt{out}(\mathcal{L}_{i+1}) = \mathtt{out}(\mathcal{L}_i) \cup \{x \mid x$ is not labelled in $\mathcal{L}_i$, and $\exists y$: $y \mathcal{R} x$ and $y \in \mathtt{in}(\mathcal{L}_{i+1})\}$
5: **until** $\mathcal{L}_{i+1} = \mathcal{L}_i$
6: $\mathcal{L}_G = (\mathtt{in}(\mathcal{L}_i), \mathtt{out}(\mathcal{L}_i), \mathcal{A} - (\mathtt{in}(\mathcal{L}_i) \cup \mathtt{out}(\mathcal{L}_i))$

Consider the following example framework:

$$a \rightarrow b \rightarrow c \ , \ d \rightleftarrows e$$

$\mathcal{L}_1 = (\{a\},\{b\},\emptyset)$, $\mathcal{L}_2 = (\{a,c\},\{b\},\emptyset)$, $\mathcal{L}_3 = \mathcal{L}_2$ and so $\mathcal{L}_G = (\{a,c\},\{b\},\{d,e\})$.

Finally, notice that the algorithm presented here can be made more efficient in a number of ways. For example, when assigning IN to arguments in line 3, checking whether all attackers are OUT can be made more efficient by giving each argument

a counter `attackers-out` that represents the number of attackers that are labelled `OUT`. Since all arguments are initially unlabelled, this counter is set to zero before the actual labelling begins. Every time that an argument is labelled `OUT`, it sends a message to each of the arguments that it attacks to increase its variable `attackers-out`. Evidently, if this variable equals the number of attackers, the attacked argument can be labelled `IN`.

4.2 Argument games for the Grounded Semantics

We have discussed how, in defending an argument x's membership of the grounded extension, one must not loop back to x itself, and how the same restriction applies to any argument moved in x's line of defence. Intuitively, this is captured by a legal move function ϕ_{G_1} that prohibits PRO from repeating arguments it has already moved in a dispute.

Definition 6.8. Given $\langle \mathcal{A}, \mathcal{R} \rangle$, a dispute d such that x is the last argument in d, and PRO(d) the arguments moved by PRO in d, then ϕ_{G_1} is a legal move function such that:

- If d is of odd length (next move is by OPP) then $\phi_{G_1}(d) = \{y \mid y \mathcal{R} x\}$
- If d is of even length (next move is by PRO) then:

$$\phi_{G_1}(d) = \{y \mid \quad 1.\ y \mathcal{R} x \quad 2.\ y \notin \text{PRO}(d)\}$$

Theorem 6.2. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be a finite argumentation framework. Then, there exists a ϕ_{G_1}-winning strategy T for x such that the set PRO(T) of arguments moved by PRO in T is conflict free, iff x is in the grounded extension of AF.*

One can give an intuitive proof of Theorem 6.2 by appealing to the correspondence between the grounded extension and grounded labelling of an argumentation framework (see Theorem 6.1). That is to say, by showing that:

1. Let T be a ϕ_{G_1}-winning strategy for x such that $PRO(T)$ is conflict free. Then there is a grounded labelling $\mathfrak{L}$ with $\mathfrak{L}(x) = \texttt{IN}$.
2. Let $\mathfrak{L}$ be a labelling with $\mathfrak{L}(x) = \texttt{IN}$. Then there exists a ϕ_{G_1}-winning strategy for x such that $PRO(T)$ is conflict free.

Proof of the above correspondences can be found in [21].

Consider the example framework in Figure 6.3i). Part of the dispute tree induced by a is shown in Figure 6.3ii), and the ϕ_{G_1} dispute tree induced by a is shown in Figure 6.3iii). Observe that $\{(a_1 - b_2 - c_3 - d_4 - e_6)\}$ is a ϕ_{G_1}-winning strategy for a (a is in the grounded extension $\{a, c, e\}$). Finally, consider the example framework in Figure 6.3iv). In this case the ϕ_{G_1}-winning strategy for a consists of two disputes:

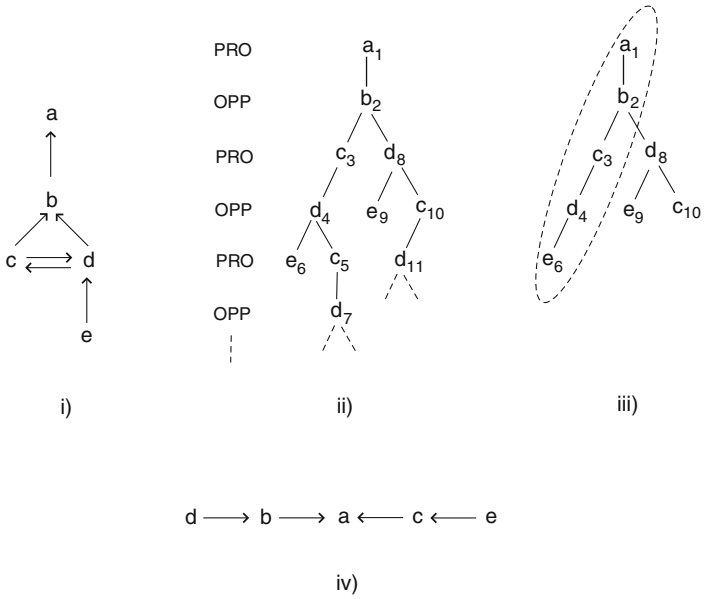

Fig. 6.3 i) shows an argumentation framework and ii) shows the dispute tree induced in a. iii) shows the ϕ_{G_1}-dispute tree induced by a and the ϕ_{G_1} winning strategy encircled. The ϕ_{G_1} winning strategy for a, in the framework in iv), consists of two disputes.

$$\{(a_{PRO} - b_{OPP} - d_{PRO}), (a_{PRO} - c_{OPP} - e_{PRO})\}.$$

Some gain in efficiency can be obtained by a legal move function ϕ_{G_2} that additionally prohibits PRO from moving a y that is itself attacked by the x that PRO moves against (i.e., augmenting 1 and 2 in Definition 6.8 with $\neg(x\mathcal{R}y)$). This is because if PRO moves such a y against x, then OPP can simply repeat x and move against y, and then PRO will be prevented from repeating y. The ϕ_{G_2} game is instantiated by Prakken and Sartor for their argument-based system of prioritized extended logic programming [24]. Amgoud and Cayrol [1] do the same for their argument based system for inconsistency handling in propositional logic. The following soundness and completeness result can be proved as a straightforward generalisation of proofs for the specific systems in [24, 1]. Such a generalised proof can be found in [4].

Theorem 6.3. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be a finite argumentation framework. Then, there exists a ϕ_{G_2}-winning strategy T for x such that the set PRO(T) of arguments moved by PRO in T is conflict free, iff x is in the grounded extension of AF*

Since the arguments moved by PRO in a winning strategy are required to be conflict free, it is obvious to see that shorter proofs may also be obtained by preventing PRO from moving arguments in a dispute d that attack themselves or attack or are attacked by arguments that PRO has already moved in d.

Definition 6.9. Let $\text{POSS}(d) = \{y \mid \neg(y\mathcal{R}y)$ and $\forall z \in \text{PRO}(d), \neg(z\mathcal{R}y)$ and $\neg(y\mathcal{R}z)\}$.

One can then further restrict PRO's moves in Definition 6.8, by adding the condition that $y \in \text{POSS}(d)$, thus obtaining the legal move function ϕ_{G_3}.

Finally, further gains in efficiency can be obtained by noticing that if T is a ϕ_{G_1}, ϕ_{G_2} or ϕ_{G_3} winning strategy, then $\text{PRO}(T)$ is conflict free. Thus, one need not instigate the conflict free check on winning strategies suggested by the above soundness and completeness results. To see why, notice that ϕ_{G_1}, ϕ_{G_2} and ϕ_{G_3} make no restrictions on moves by OPP, and one can show that the following theorem holds:

Theorem 6.4. *Let T be a ϕ winning strategy such that ϕ makes no restrictions on moves by OPP. Then PRO(T) is conflict free.*

We refer the reader to [21] for a proof of the above theorem.

5 Preferred Semantics

For any argumentation framework, existence of a preferred extension is guaranteed, and there can be more than one preferred extension. Hence, the decision questions 2a (credulous membership) and 2b (sceptical membership) are distinct. In Section 5.2 we describe argument games for addressing the credulous membership question. Section 5.3 then describes argument games for addressing the more difficult sceptical membership question. Solution-orientated questions 2c - 2h require procedures for identifying one or all preferred extensions. Such questions become relevant when end-users would like to be informed about the reasons as to how and why an argument is justified or overruled, and can be addressed by labelling algorithms that compute one or all preferred labellings. We describe labelling algorithms in the following section.

5.1 A Labelling Algorithm for the Preferred Semantics

In this Section we review Caminada's work on labelling algorithms [6]. Theorem 6.1 in Section 2 establishes an equivalence between an argumentation framework's preferred extensions and the framework's preferred labellings. In [5] it is shown that:

$\mathcal{L}$ is a preferred labelling iff $\mathcal{L}$ is an admissible labelling such that for no admissible labelling $\mathcal{L}'$ is it the case that $\texttt{in}(\mathcal{L}') \supset \texttt{in}(\mathcal{L})$. **(R1)**

Hence, a framework's preferred extensions can be identified by algorithms that compute admissible labellings that maximise the number of arguments that are legally IN. In [6], admissible labellings are generated by starting with a labelling that labels all arguments IN and then iteratively, selects arguments that are illegally IN and applies a *transition step* to obtain a new labelling, until a labelling is reached in which no argument is illegally IN.

Definition 6.10. Let $\mathcal{L}$ be a labelling for $\langle \mathcal{A}, \mathcal{R} \rangle$ and x an argument that is illegally IN in $\mathcal{L}$. A *transition step* on x in $\mathcal{L}$ consists of the following:

1. the label of x is changed from IN to OUT
2. for every $y \in \{x\} \cup \{z | x \mathcal{R} z\}$, if y is illegally OUT, then the label of y is changed from OUT to UNDEC (i.e., any argument made illegally OUT by 1 is changed to UNDEC)

In what follows, we assume a function *transition step* that takes as input x and $\mathcal{L}$, and applies the above operations to yield a labelling $\mathcal{L}'$. We then define a *transition sequence* as follows:

A *transition sequence* is a list $[\mathcal{L}_0, x_1, \mathcal{L}_1, x_2, \ldots, x_n, \mathcal{L}_n]$ ($n \geq 0$), where for $i = 1 \ldots n$, x_i is illegally IN in $\mathcal{L}_{i-1}$, and $\mathcal{L}_i$ = *transition step*$(\mathcal{L}_{i-1}, x_i)$.

A *transition sequence* is said to be terminated iff $\mathcal{L}_n$ does not contain any argument that is illegally IN.

Let us examine a transition sequence that starts with the initial labelling $\mathcal{L}_0$ in which all arguments are labelled IN (from hereon any such labelling is referred to as an 'all-in' labelling and we assume that any initial labelling $\mathcal{L}_0$ is an all-in labelling). Any labelling containing an argument x that is illegally IN cannot be a candidate admissible labelling (since not all of x's attackers are OUT and so x is not reinstated against all attackers), and so must be relabelled OUT. One might expect that the second part of the transition step relabels to IN, those arguments that are made illegally OUT by the first step. However this may not only result in a loop, but would also 'overcommit' arguments to membership of an admissible labelling (and so extension); just because an argument may be acceptable w.r.t. an admissible extension E does not mean that it must be in E.

For finite frameworks it can be shown that:

For any terminated transition sequence $[\mathcal{L}_0, x_1, \mathcal{L}_1, x_2, \ldots, x_n, \mathcal{L}_n]$, it holds that $\mathcal{L}_n$ is an admissible labelling. **(R2)**

To see why, observe that $\mathcal{L}_0$ contains no arguments that are illegally OUT, and it is straightforward to show that a transition step preserves the absence of arguments that are illegally OUT. Hence, since the terminated sequence contains no arguments that are illegally IN, then by Definition 6.3, $\mathcal{L}_n$ is admissible.

In [6], it is also shown that:

For any preferred labelling $\mathfrak{L}$, it holds that there exists a terminated transition sequence $[\mathfrak{L}_0, x_1, \mathfrak{L}_1, x_2, \ldots, x_n, \mathfrak{L}_n]$, where $\mathfrak{L}_n = \mathfrak{L}$. **(R3)**

The above results R1, R2 and R3, imply that terminated transition sequences whose final labellings maximise the arguments labelled *IN* are exactly the preferred labellings. Before presenting the algorithm for generating such sequences, let us consider how admissible labellings are generated for the argumentation framework in Figure 6.4i). Starting with the initial all-in labelling $\mathfrak{L}_0 = (\{a,b,c\},\emptyset,\emptyset)$, then selecting a on which to perform a transition step obtains $\mathfrak{L}_1 = (\{b,c\},\{a\},\emptyset)$. Now only c is illegally IN, and relabelling it to OUT results in a being illegally *OUT*, and so $\mathfrak{L}_2 = (\{b\},\{c\},\{a\})$. Now b is illegally IN, and relabelling b to OUT results in both b and c being illegally OUT, so that they are both labelled UNDEC. Thus, the transition sequence terminates with the labelling $\mathfrak{L}_3 = (\emptyset,\emptyset,\{a,b,c\})$ in which all arguments are UNDEC. It is easy to verify that irrespective of whether a, b or c is selected on the first transition step, every terminated transition sequence will result in $\mathfrak{L}_3$.

Consider now the framework in Figure 6.4ii). Starting with the initial all-in labelling $\mathfrak{L}_0 = (\{a,b,c\},\emptyset,\emptyset)$, we observe that *B* and *C* are illegally IN:

1. Selecting b for the first transition step obtains the terminated sequence $[\mathfrak{L}_0, b, \mathfrak{L}_1 = (\{a,c\},\{b\},\emptyset)]$. $\mathfrak{L}_1$ is an admissible and complete labelling, yielding the admissible and complete extension $\{a,c\}$.
2. Selecting c for the first transition step obtains the terminated sequence $[\mathfrak{L}_0, c, \mathfrak{L}_1 = (\{a,b\},\{c\},\emptyset), b, \mathfrak{L}_2 = (\{a\},\{b\},\{c\})]$, yielding the admissible extension $\{a\}$

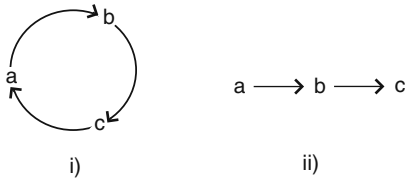

Fig. 6.4 Two argumentation frameworks

Notice that in the second sequence, the label of c is changed from OUT to UNDEC since c is made illegally OUT by the second transition step's assignment of OUT to the illegally IN b. $\mathfrak{L}_2$ is an admissible but not complete labelling, since c is illegally UNDEC. To help avoid non-complete labellings, one can guide the choice of arguments on which to perform transition steps: *choose an argument that is* super-illegally IN*, if such an argument is available*.

Definition 6.11. An argument x in $\mathfrak{L}$ that is illegally IN, is also *super-illegally* IN iff it is attacked by a y that is legally IN in $\mathfrak{L}$, or UNDEC in $\mathfrak{L}$.

Thus, b would preferentially be selected according to the above strategy, since b and not c is super-illegally IN in $(\{a,b,c\},\emptyset,\emptyset)$. As shown in [6], both the results R2 and R3 are preserved under such a strategy.

Algorithm 6.2 Algorithm for Preferred Labellings

```
 1: candidate-labellings := ∅;
 2: find_labellings(all-in);
 3: print candidate-labellings;
 4: end.
 5: .
 6: .
 7: procedure find_labellings(£)
 8: .
 9: # if £ is worse than an existing candidate labelling then prune the search tree
10: # and backtrack to select another argument for performing a transition step
11: if ∃£' ∈ candidate-labellings: in(£) ⊂ in(£') then return;
12: .
13: # if the transition sequence has terminated
14: if £ does not have an argument that is illegally IN then
15:    for each £' ∈ candidate-labellings do
16:       # if £''s IN arguments are a strict subset of £'s IN arguments
17:       # then remove £'
18:       if in(£') ⊂ in(£) then
19:          candidate-labellings :=
20:          candidate-labellings − {£'};
21:       end if
22:    end for
23:    # add £ as a new candidate
24:    candidate-labellings := candidate-labellings ∪ {£};
25:    return; # we are done, so try the next possibility
26: else
27:    if £ has an argument that is super-illegally IN then
28:       x := some argument that is super-illegally IN in £;
29:       find_labellings(transition_step(£, x));
30:    else
31:       for each x that is illegally IN in £ do
32:          find_labellings(transition_step(£, x))
33:       end for
34:    end if
35: end if
36: endproc
```

We now describe the above listed algorithm for generating preferred labellings. The main procedure `find_labellings` starts with the all-in labelling, and then iteratively applies transitions steps in an attempt to generate terminated transition sequences that update the global variable `candidate-labellings`. The algorithm preferentially selects from amongst super-illegal arguments for performing transition steps, if such arguments are available. If at any stage in the generation of a transition sequence, the arguments that are IN in the labelling $£_i$ thus far obtained

are a strict subset of $\texttt{in}(\mathcal{L}')$ for some $\mathcal{L}' \in \texttt{candidate-labellings}$, then no further transition steps on $\mathcal{L}_i$ can result in a preferred labelling (that maximises the arguments that are IN). This follows from the result that during the course of a transition sequence, the set of IN labelled arguments monotonically decreases (as shown in [6]). Thus, any further transition steps on $\mathcal{L}_i$ will only reduce the arguments that are IN. In such cases, the algorithm backtracks to $\mathcal{L}_{i-1}$ and, if possible, selects another argument on which to perform a transition step. In the case that a transition sequence terminates, the obtained labelling $\mathcal{L}$ is compared with all labellings $\mathcal{L}'$ in candidate-labellings. If for any $\mathcal{L}'$, $\texttt{in}(\mathcal{L}')$ is a strict subset of $\texttt{in}(\mathcal{L})$, then $\mathcal{L}'$ is removed from candidate-labellings. Thus, given a finite argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$, the algorithm calculates the preferred labellings and so preferred extensions.

5.2 Argument Games for the Credulous Preferred Semantics

Since the admissible extensions of a framework form a complete partial order with respect to set inclusion (and so every admissible extension is a subset of a preferred extension), then for argument games addressing the credulous membership question, it suffices to show an admissible extension containing the argument in question. In contrast with the grounded semantics, x's membership of an admissible extension E can now 'appeal to' x itself, in the sense that in defending x's membership of E, and membership of all subsequent defenders, one can loop back to x itself. This then means, that to prevent infinite disputes, it is now OPP, rather than PRO, that should not be allowed to repeat an argument y it has already moved in a dispute, since y can then be attacked by PRO repeating the argument it moved against OPP's first move of y.

Consider the framework in Figure 6.5i), and the dispute tree induced in a in Figure 6.5ii). In both disputes (branches) PRO is allowed to repeat its arguments (c_5 and d_9). OPP repeats its arguments, and the disputes continue with PRO repeatedly fulfilling its burden of proof w.r.t. c (d). It is of course sufficient that PRO fulfill its burden of proof only once. Hence, as well as preventing PRO from introducing a conflict into a dispute, the following legal move function prohibits OPP from repeating arguments.

Definition 6.12. Given $\langle \mathcal{A}, \mathcal{R} \rangle$, a dispute d such that x is the last argument in d, and OPP(d) the arguments moved by OPP in d, then ϕ_{PC_1} is a legal move function such that:

- If d is of odd length (next move is by OPP) then:
  $\phi_{PC_1}(d) = \{y \mid$
  1. $y \mathcal{R} x$
  2. $y \notin \text{OPP}(d)$
  $\}$

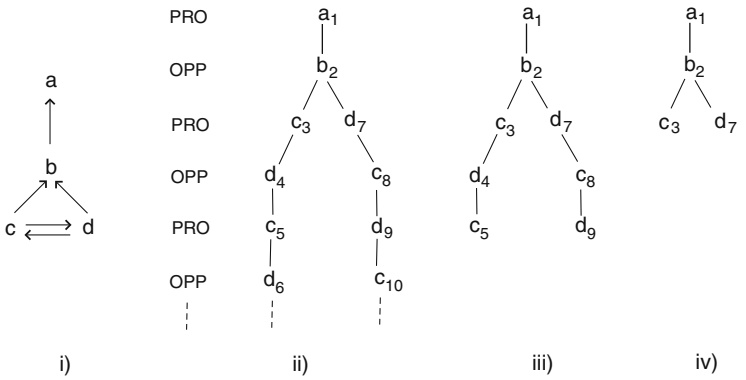

Fig. 6.5 i) shows an argumentation framework and ii) shows the dispute tree induced in a. iii) and iv) respectively shows the ϕ_{PC_1} and ϕ_{PC_2} dispute trees induced by a.

- If d is of even length (next move is by PRO) then:
  $\phi_{PC_1}(d) = \{y \mid$
  1. $y\mathcal{R}x$
  2. $y \in \text{POSS}(d)$
  $\}$

Notice that ϕ_{PC_1} mirrors Section 4.2's grounded game function ϕ_{G_3} (that augments ϕ_{G_1} to restrict PRO to moving arguments in POSS(d)). They differ only in that ϕ_{PC_1} prevents repetition by OPP, and ϕ_{G_3} prevents repetition by PRO.

Consider again the framework in Figure 6.5i). The ϕ_{PC_1} dispute tree induced by a is shown in Figure 6.5iii), and both disputes in the tree individually constitute ϕ_{PC_1} winning strategies. Notice that for the example framework in Figure 6.3iv), the ϕ_{PC_1}-winning strategy for a consists of two disputes: $\{(a_{PRO} - b_{OPP} - d_{PRO}), (a_{PRO} - c_{OPP} - e_{PRO})\}$.

The following theorem states the soundness and completeness result for ϕ_{PC_1} games:

Theorem 6.5. *Let $AF = \langle \mathcal{A}, \mathcal{R} \rangle$ be a finite argumentation framework. Then, there exists a ϕ_{PC_1}-winning strategy T for x such that the set PRO(T) of arguments moved by PRO in T is conflict free, iff x is in an admissible (and hence preferred) extension of AF.*

One can give an intuitive proof of the above by using the correspondence between admissible extensions and admissible labellings of an argumentation framework (see Theorem 6.1). That is, it suffices to prove that:

1. Let T be a ϕ_{PC_1}-winning strategy for x such that PRO(T) is conflict free. Then there exists an admissible labelling $\mathcal{L}$ with $\mathcal{L}(x) = \texttt{IN}$.

2. Let $\mathcal{L}$ be an admissible labelling with $\mathcal{L}(x) = \texttt{IN}$. Then there exists a ϕ_{PC_1}-winning strategy for x such that $PRO(T)$ is conflict free.

Proof of the above correspondences can be found in [21].

Observe that the spectrum of outcomes would not be changed by a function ϕ_{PC_2} that augments ϕ_{PC_1} by prohibiting OPP from moving *any* argument y (and not just a y already moved by OPP) that is attacked by an argument x in PRO(d). This is because PRO can then simply move x against y, and if $y\mathcal{R}x$, prohibiting repetition by OPP will mean that y cannot be moved against this second move of x by PRO. Notice that if this prohibition on OPP is in place, then one cannot have a dispute of the form $(\ldots y_{PRO}\ldots x_{OPP} - y_{PRO}\ldots)$ in which PRO repeats an argument, since the prohibition on OPP would prevent the move x_{OPP}. Hence, shorter proofs can be obtained by a function ϕ_{PC_2} that augments ϕ_{PC_1} by prohibiting OPP from moving any argument attacked by an argument in PRO(d), and prohibiting repetition by PRO. Indeed, [9] prove that the following theorem holds:

Theorem 6.6. *Let $AF = \langle \mathcal{A}, \mathcal{R}\rangle$ be a finite argumentation framework. Then, there exists a ϕ_{PC_2}-winning strategy for x such that the set PRO(T) of arguments moved by PRO in T is conflict free, iff a is in a preferred extension of AF.*

Figure 6.5iv) shows the ϕ_{PC_2} dispute tree induced by a for the framework in Figure 6.5i), where both disputes in the tree are ϕ_{PC_2} winning strategies. However, notice that neither dispute fully fulfills the remit of a proof to *explain* the credulous membership of a, since neither demonstrates the reinstatement of c, respectively d, against its attacker d, respectively c, and so provides an explanation for the admissibility of $\{a,c\}$, respectively $\{a,d\}$. This illustrates a more general point that efficiency gains often come at the expense of explanatory power.

Finally, note that unlike games for the grounded semantics, checking that the arguments moved by PRO in a ϕ_{PC_1} (or ϕ_{PC_2}) winning strategy are conflict free, is required. This is because ϕ_{PC_1} and ϕ_{PC_2} games place restrictions on moves by OPP (and hence the result concluding Section 4.2 does not hold). For example, consider that a is not in an admissible, and hence preferred, extension of the framework in Figure 6.6. Now, $\{(a_{PRO} - b_{OPP} - c_{PRO} - d_{OPP} - g_{PRO}), (a_{PRO} - e_{OPP} - f_{PRO} - g_{OPP} - d_{PRO})\}$ is a ϕ_{PC_1} winning strategy since OPP cannot legally extend either dispute. However, the arguments moved by PRO are not conflict free (PRO has moved g and d).

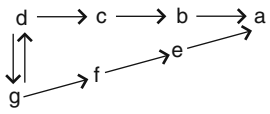

Fig. 6.6 Argument a is not in an admissible and so preferred extension of the above framework.

5.3 Argument Games for the Sceptically Preferred Semantics

The question of whether an argument is sceptically preferred is much harder to answer than the credulously preferred membership problem. To understand why, it may first help to realise that the credulous membership problem only requires us to point at one extension, while the sceptical membership problem requires us to prove something about all possible extensions. Thus, the credulously preferred membership problem is an existence problem while the sceptically preferred membership problem is a verification problem. To understand better why verification is hard in this case, we recall the definition of sceptically preferred membership: an argument a is *sceptically preferred* iff it is a member of all preferred extensions. The crux of the problem is that we have to verify whether there exists preferred extensions that do not contain a. In so doing, it is not immediately clear where to begin to search for such extensions.

The following result establishes a connection between a and preferred extensions that might possibly exclude a (we refer the reader to [21] for a proof of this result). It basically ensures that the search space for the sceptical decision problem is confined to elements that are indirectly connected to defense sets of a.

Theorem 6.7 (Complement lemma). *An argument a is sceptically preferred if and only if for every admissible extension B, there is an admissible extension A, containing a, that is consistent with B.*

Thus, conversely, a is not sceptically preferred if there exists an admissible extension B that conflicts with all admissible extensions around a. Because such an extension B blocks sceptically preferred membership, such an extension is called a *block*. With the help of Theorem 6.7 we may now formulate an abstract and inefficient, but conceptually correct proof procedure to determine sceptical membership. This procedure works by falsification, as follows. Try to construct a block B. If this attempt fails, we may, with the help of Theorem 6.7 conclude that a is sceptically preferred.

The procedure to block a can be described as an argument game that we informally describe here. The difference with the games described earlier, is that the players exchange entire admissible extensions rather than single arguments. The game works as follows. Suppose PRO's goal is to show that a is sceptically preferred. To this end, PRO starts by constructing an admissible extension, $A\{1\}$ around a. Since $A\{1\}$ is the only admissible extension known at this stage, it follows that at this stage a is sceptically preferred. To invalidate this temporary conclusion, the burden of proof shifts to OPP who must show that a is not sceptically preferred. By virtue of Theorem 6.7 it suffices for OPP to show that there exists an admissible extension that conflicts with $A\{1\}$. If OPP does not manage to construct such an extension, the procedure ends and OPP has lost. Suppose OPP manages to produce $A\{1,1\}$ as a response to $A\{1\}$. Thus, $A\{1,1\}$ is an admissible extension that conflicts with $A\{1\}$. Once $A\{1,1\}$ is advanced, a is no longer sceptically preferred, because $A\{1,1\}$ conflicts with every admissible extension around a constructed thus far, viz. $A\{1\}$. To invalidate this temporary conclusion, the burden of proof shifts back to PRO who must now show that there exists another admissible extension

around a that does not conflict with $A\{1,1\}$. If PRO fails to do so (and PRO's search was adequate and exhaustive), it follows that $A\{1,1\}$ conflicts with all admissible extensions around A, so that a is not sceptically preferred. Suppose otherwise, i.e, suppose that PRO is able to construct an admissible extension, $A\{1,1,1\}$, that does not conflict with $A\{1,1\}$. OPP must now either extend $A\{1,1\}$ such that it also conflicts with $A\{1,1,1\}$ or else drop $A\{1,1\}$ to start all over to attack another member of $A\{1\}$. Continuing this way (including backtracking), OPP is busy with extending an admissible extension until either PRO is unable to produce another admissible extension around a, or else until OPP's admissible extension cannot be further extended (on pain of becoming inconsistent).

More generally, we may suppose that $A\{1\},\ldots,A\{n\}$ are possible begin moves of PRO, and $A\{i_1,\ldots,i_k,m\}$, $k \geq 1$ is the mth possible response of either PRO or OPP to $A\{i_1,\ldots,i_k\}$. Naturally, all the $A\{\bar{i}\}$ are admissible extensions. The following constraints hold:

1. Every extension advanced by PRO must contain the main argument, a.
2. Every response of PRO must be consistent with the extension that is previously advanced by OPP.
3. Every response of OPP must attack PRO's immediately preceding extension.
4. Within one branch, every extension advanced by OPP must be an extension of OPP's previous extension in the same branch.
5. Both parties may backtrack and construct alternative replies.
6. OPP has won if it is able to move last; else PRO has won.

If OPP has won this means that OPP was able to create a block $B = A\{i_1,\ldots,i_{2k}\}$, where $k \geq 1$ (note that we have '$2k$' since all moves by OPP have an even number of indices). With B, OPP is able to move last in the particular branch where that block was created and all sub-branches emanating from the main branch. It must be noted that all this only works in finitary argument systems, i.e., argument systems where all arguments have a finite number of attackers. Algorithms for non-finitary argument systems require additional constraints such as *fairness* which must guarantee that every possibility is enumerated eventually.

The above ideas are taken from earlier work on the sceptically preferred membership problem, notably that of Doutre *et al.* [12] and Dung *et al.* [15]. In [12], the procedure to find a possible block is presented as a so-called meta-acceptance dialogue. As above, moves in this dialogue are extensions (hence the *meta*), and a dialogue is won by OPP if it is able to move last in at least one branch. In Dung *et al.* [15] the procedure to construct a "fan" of admissible extensions around A that together represent all preferred extensions is called *generating a complete base* for a. A *base* for a is a set of admissible extensions, $\mathcal{B}$, such that every preferred extension around a includes at least one element of $\mathcal{B}$. A *complete base* for a, then, is a set of admissible extensions, $\mathcal{B}$, such that *every* preferred extension includes at least one element of $\mathcal{B}$. In line with Theorem 6.7, Dung *et al.* proceed to show that a base $\mathcal{B}$ is incomplete if and only if there exists a preferred extension that attacks every element of $\mathcal{B}$. Their proof procedure is a combination of a so-called BG-derivation (base generation derivation) followed by a CB-verification (complete base verifica-

tion). With BG a base for a is generated, such that every preferred extension around a contains an element of $\mathcal{B}$. Such a base always exists, but not every base may serve as a representant of sceptical membership. To check whether $\mathcal{B}$ indeed represents sceptical membership, it is checked for completeness, which effectively means that it must hold out against every candidate block that might undermine $\mathcal{B}$. Again, all decision procedures only work in finitary argument systems.

6 Stable and Semi-Stable Semantics

Stable semantics are, what one might call 'xenophobic', since every argument outside of a stable extension is attacked by an argument in the stable extension. Unlike the preferred semantics, existence of a stable extension is not guaranteed; consider that a framework consisting of a single argument that attacks itself has no stable extension. However, as in the case of the preferred semantics, there may be more than one extension, and so decision questions 2a (credulous membership) and 2b (sceptical membership) are distinct. These questions, questions 1b, 1c, and the solution-orientated questions 2a - 2h can be addressed by an algorithm (taken from [6]) that generates all stable extensions of a framework. Since a stable labelling makes all arguments either `OUT` or `IN`, one can straightforwardly adapt the algorithm for preferred labellings in Section 5.1, so as to only yield labellings without `UNDEC` labelled arguments. Thus, line 11 in the algorithm is replaced by:
if $\texttt{undec}(\mathcal{L}) \neq \emptyset$ **then return**;
Furthermore, we do not have to compare the arguments made IN by other candidate labellings, and so we can remove lines 15 to 22. The result is an algorithm that calculates all stable extensions of a finite framework.

Argument games for stable semantics have only recently been studied. In [28], the authors study *coherent* argumentation frameworks, in which every preferred extension is also stable (meaning that the preferred and stable extensions coincide, since each stable extension is by definition also a preferred extension). Thus, for coherent argumentation frameworks, one can simply apply existing games for the preferred semantics to decide membership under stable semantics.

For the general case, where one is not restricted to coherent argumentation frameworks, the situation is more complex, but can still be expressed in terms of the credulous games defined in Section 5.2. Given a framework $\langle \mathcal{A}, \mathcal{R} \rangle$, and letting $PRO(T)$, respectively $OPP(T)$, denote the arguments moved by PRO, respectively OPP, in a dispute tree T, then an argument x is in a stable extension iff there exists a set S of ϕ_{PC_1} winning strategies such that:

1. at least one winning strategy in S is for x.
2. $\bigcup\{PRO(T)|T \in S\}$ is conflict free.
3. $\bigcup\{PRO(T) \cup OPP(T)|T \in S\} = \mathcal{A}$

This can be seen as follows. First of all, each ϕ_{PC_1} winning strategy corresponds to an admissible labelling. A set of winning strategies that do not attack each other (point 2) again corresponds to an admissible labelling. If this resulting admissible labelling spans the entire argumentation framework (each argument is either IN or OUT) then this labelling is also stable (point 3). Then, if x is IN in this labelling, then x is labelled IN in at least one stable labelling (point 1).

It is also possible to define a single dispute game that determines credulous acceptance w.r.t. stable semantics. Such a game has recently been stated by Caminada and Wu [7]. One particular feature of their approach, which builds on the work of Vreeswijk and Prakken [28], is that they do not use the concept of a winning strategy. Instead, for an argument x to be in a stable extension, it suffices to have at least one game for x that is won by PRO. Caminada and Wu are able to do this by first defining a game for credulous preferred in which PRO may repeat its own moves, but not the moves of OPP, and in which OPP may repeat PRO's moves but not its own moves. Moreover, PRO has to react to the directly preceding move of OPP, whereas OPP is free to react either to the directly preceding move of PRO, *or* to a previous PRO move. A dispute is won by PRO iff OPP cannot move. A dispute is won by OPP iff PRO cannot move, or if OPP managed to repeat one of PRO's moves.

Basically, the game can be understood in terms of PRO and OPP building an admissible labelling in which PRO makes IN moves, and OPP makes OUT moves. This game can be altered to implement stable semantics by introducing a third kind of move, which is called *QUESTION*. By uttering *QUESTION* x, OPP asks PRO for an explicit opinion on argument x. PRO is then obliged to reply with either IN x or with IN y, where y is an attacker of x. Caminada and Wu show that this game indeed models credulous acceptance under the stable semantics.

Once a procedure for credulous acceptance w.r.t. stable semantics has been defined, the issue of sceptical acceptance w.r.t. stable semantics becomes relatively straightforward: an argument x is in all stable extensions iff one fails to establish credulous membership of any attacker of x. For the left to right half, observe that if x is in all stable extensions, then all attackers of x are attacked by all such extensions (an argument y is *attacked by an extension* if it is attacked by an argument in that extension), and so no attacker of x can be in any such extension, since each such extension is conflict free. For the right to left half, observe that if any attacker of x does not belong to any stable extension, then it is attacked by all such extensions. Thus every extension contains an argument that reinstates x, and so contains x.

Caminada has recently proposed *semi-stable* semantics [5, 6], that unlike the stable semantics, guarantees that every (finite) framework has at least one semi-stable extension. In the case that there exists at least one stable extension for a framework, semi-stable semantics yield the same extensions as stable semantics. From the perspective of argument labellings, semi-stable semantics select those labellings in which the set of UNDEC arguments is minimal. Referring to Definition 6.4, this can be expressed as follows:

Let $\mathfrak{L}$ be a complete labelling. Then $\mathfrak{L}$ is a semi-stable labelling iff there does not exist a complete labelling $\mathfrak{L}'$ such that $\texttt{undec}(\mathfrak{L}') \subset \texttt{undec}(\mathfrak{L})$

For example, consider the framework in Figure 6.4ii) augmented by an additional argument d that attacks itself. The augmented framework has no stable extension, but $\{a,c\}$ is the single semi-stable extension equating with the semi-stable labelling $(\{a,c\},\{b\},\{d\})$. Notice that although $\{a,c\}$ is also the single preferred extension, in general not every preferred extension is a semi-stable extension since not every preferred extension minimises UNDEC. However, every semi-stable extension is a preferred extension, which suggests that we can adapt Section 5.1's algorithm for preferred labellings in order to compute semi-stable labellings.

In [6] it is also shown that:

$\mathfrak{L}$ is a semi-stable labelling iff $\mathfrak{L}$ is an admissible labelling such that for no admissible labelling $\mathfrak{L}'$ is it the case that $\texttt{undec}(\mathfrak{L}') \subset \texttt{undec}(\mathfrak{L})$. (**R1′**)

Since every semi-stable extension is a preferred extension then R3 in Section 5.1 also holds for semi-stable labellings $\mathfrak{L}$. This result, together with R1′ and R2 in Section 5.1, implies that terminated transition sequences whose final labellings minimise the arguments labelled *UNDEC* are exactly the semi-stable labellings. Hence, one can adapt Section 5.1's algorithm by replacing line 11 by:

if $\exists \mathfrak{L}' \in$ `candidate-labellings`: $\texttt{undec}(\mathfrak{L}') \subset \texttt{undec}(\mathfrak{L})$ **then return**;

In other words, if at any stage in the generation of a transition sequence, the UNDEC arguments of the labelling $\mathfrak{L}_i$ thus far obtained, are a strict superset of $\texttt{undec}(\mathfrak{L}')$ for some $\mathfrak{L}' \in$ `candidate-labellings`, then no further transition steps on $\mathfrak{L}_i$ can result in a semi-stable labelling, and so one can backtrack to perform a transition step on another choice of argument. This follows from the result that during the course of a transition sequence, the set of UNDEC labelled arguments monotonically increases (as shown in [6]). Finally, we replace line 18 with:

if $\texttt{undec}(\mathfrak{L}) \subset \texttt{undec}(\mathfrak{L}')$;

and we are done. We have an algorithm that calculates the semi-stable labellings of a finite argumentation framework.

7 Conclusions

In this chapter we have described labelling algorithms and argument game proof theories for various argumentation semantics. Labellings and argument games can be seen as alternatives to the *extension-based* approach to specifying argumentation

semantics described in Chapter 2. We conclude with some further reflections on these different ways of specifying argumentation semantics.

One of the original motivations for developing the labelling approach was to provide an easy and intuitive account of formal argumentation. After all, principles like "In order to accept an argument, one has to be able to reject all its counterarguments" and "In order to reject an argument, one has to be able to accept at least one counterargument" are easy to explain and have therefore been used as the basis of the labelling approach. Also, our teaching experiences indicate that students who are new to argumentation tend to find it easier to understand the labelling approach rather than the extension-based approach to argumentation. In fact, it is often easier for them to understand the extension-based approach after having been introduced to the labelling approach.

Another advantage of the labelling approach is that it allows one to specify a number of relatively small and simple properties, each of which can be individually satisfied or not, and that collectively define the argumentation semantics. This modular approach can be of assistance when constructing formal proofs. Also, by explicitly distinguishing between `IN`, `OUT` and `UNDEC` (instead of merely specifying the set of `IN`-labelled arguments as in the extension-based approach) one is provided with more detailed information. For instance, Section 5.1's algorithm for generating all preferred extensions, would be much more difficult to specify using the extension-based approach.

Finally, we note that the labelling approach essentially identifies a graph or 'network' labelling problem, suggesting that the approach more readily lends itself to extensions of argument frameworks that accommodate: different types of relation between arguments (e.g. support [10] and collective attack [22]); attacks on attacks [20]; multi-valued and quantitative valuations of arguments [2, 11], and so on. In essence, these extensions of Dung's abstract argumentation framework can be understood as instantiating a more general network reasoning model in which the valuations of nodes (arguments) is determined by propagating the valuations of the connected nodes, as mediated by the semantics of the connecting arcs. Algorithms for determining these valuations will thus generalise the three value labelling algorithms described in this chapter.

With regard to the argument game approach, we recall that Dung's abstract argumentation semantics can be understood as a semantics for a number of non-monotonic and defeasible logics [3, 13], in the sense that:

> α is an inference from a theory Δ in a logic $\mathcal{L}$, iff α is the conclusion of a justified argument of the argumentation framework $\langle \mathcal{A}, \mathcal{R} \rangle$ defined by Δ and $\mathcal{L}$.

The argument game approach places an emphasis on the dialectical nature of argumentation, in the sense that the approach appeals more directly to an inter-subjective notion of truth: truth becomes that which can be defended in a rational exchange and evaluation of interacting arguments. Thus, what accounts for the correctness of an inference is that it can be *shown* to rationally prevail in the face of

arguments for opposing inferences, where it is application of the reinstatement principle that encodes logic neutral, rational means for establishing such standards of correctness. This account of argumentation as a semantics, contrasts with model-based semantics for formal entailment that appeal to an objective notion of truth: true is that which holds in every possible model. Notice that dialectical semantics are not unique to formal argumentation. For instance, Lorenzen and Lorenz [18, 19] have proposed dialectical devices as a method of demonstration in formal logic.

An advantage of dialectical semantics is that they are able to relate formal entailment to something most people are familiar with in everyday life: debates and discussions. Argument games of the type described in this chapter are therefore useful not only for providing guidelines and principles for the design of algorithms, but also for bridging the gap between formal and informal reasoning.

Finally, we note that the dialectical view also accords with our understanding of reasoning as an incremental process. Rather than have all the arguments and their attacks defined from the outset, we incrementally acquire knowledge in order to construct arguments required to counter-argue existing arguments. At any stage in this incremental process we can evaluate the status of arguments, which in turn motivates acquisition of further knowledge for construction and submission of arguments. Argument games allow one to model such processes. Provided that there is a well understood notion of what constitutes an attack between any two arguments, one can then formalise the games described in this chapter, without reference to a pre-existing framework. This also allows one to acknowledge that reasoning agents are resource bounded, and suggests that bounds on reasoning resources may be characterised by bounds on the breadth and depth of the dispute trees constructed in order to prove the claim of the argument under test.

Acknowledgements The authors would like to thank Gerard Vreeswijk for his contributions to the contents of this chapter. Thanks also to Nir Oren for commenting on a draft of the chapter.

References

1. L. Amgoud and C. Cayrol. A Reasoning Model Based on the Production of Acceptable Arguments. *Annals of Mathematics and Artificial Intelligence*, 34(1–3),197–215, 2002.
2. H. Barringer, D. M. Gabbay and J. Woods. Temporal Dynamics of Support and Attack Networks: From Argumentation to Zoology. *Mechanizing Mathematical Reasoning*, 59–98, 2005.
3. A. Bondarenko and P.M. Dung and R.A. Kowalski and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.
4. M. Caminada. *For the sake of the Argument. Explorations into argument-based reasoning*. Doctoral dissertation Free University Amsterdam, 2004.
5. M. Caminada. On the Issue of Reinstatement in Argumentation. In *European Conference on Logic in Artificial Intelligence (JELIA)*, 111–123, 2006.
6. M. Caminada. An Algorithm for Computing Semi-stable Semantics. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 222–234, 2007.

7. M. Caminada and Y. Wu. Towards an Argument Game for Stable Semantics. In *Computational Models of Natural Argument*, to appear, 2008.
8. C. Cayrol, S. Doutre and J. Mengin. Dialectical Proof Theories for the Credulous Preferred Semantics of Argumentation Frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 668–679, 2001.
9. 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.
10. C. Cayrol and M. Lagasquie-Schiex. On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 378–389, 2005.
11. C. Cayrol and M.-Ch. Lagasquie-Schiex. Graduality in argumentation. *Journal of Artificial Intelligence Research*, 23:245–297, 2005.
12. S. Doutre and J. Mengin. On sceptical vs credulous acceptance for abstract argument systems. In *Ninth European Conference on Logics in Artificial Intelligence (JELIA 2004)*, 462–473, 2004.
13. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and *n*-person games. *Artificial Intelligence*, 77:321–357, 1995.
14. P.M. Dung, P. Mancarella and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence Journal*, 171(10–15):642–674, 2007.
15. P.M. Dung and P.M. Thang. A Sound and Complete Dialectical Proof Procedure for Sceptical Preferred Argumentation. In *Proc. of the LPNMR-Workshop on Argumentation and Nonmonotonic Reasoning (ArgNMR07)*, 49–63, 2007.
16. P.E. Dunne and T.J.M. Bench-Capon. Two Party Immediate Response Disputes: Properties and Efficiency. *Artificial Intelligence Journal*, 149(2),221–250, 2003.
17. H. Jakobovits and D. Vermeir. Dialectic Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 53–62, 1999.
18. P. Lorenzen. Dialectical foundations of logical calculi. *Constructive Philosophy*, Univ. of Massachusetts Press, 1987.
19. P. Lorenzen and K.Lorenz". Dialogische Logik. Wissenschaftliche Buchgesellschaft, Darmstadt, 1978.
20. S. Modgil. Reasoning About Preferences in Argumentation Frameworks. *Artificial Intelligence Journal*, 173(9–10), 901–934, 2009.
21. S. Modgil and M. Caminada. Proof Theories and Algorithms for Abstract Argumentation Frameworks. *Technical Report*, Department of Computer Science, King's College London, *www.dcs.kcl.ac.uk/staff/modgilsa/ProofTheoriesAlgorithms.pdf*, 2008.
22. S. Nielsen and S. Parsons. A generalization of Dung's abstract framework for argumentation: Arguing with sets of attacking arguments. In *Proc. Third International Workshop on Argumentation in Multiagent Systems (ArgMAS 2006)*, 54–73, 2006.
23. J. L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person*. MIT Press, Cambridge, MA, 1995.
24. H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.
25. B. Verheij. A Labeling Approach to the Computation of Credulous Acceptance in Argumentation. In *International Joint Conference on Aritificial Intelligence (IJCAI)*, 623–628, 2007.
26. G. A. W. Vreeswijk. Defeasible dialectics: A controversy-oriented approach towards defeasible argumentation. *Journal of Logic and Computation*, 3:3–27, 1993.
27. G. A. W. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *Proc. 1st International Conference on Computational Models of Argument*, 109–120, 2006.
28. G. A. W. Vreeswijk and H. Prakken. Credulous and sceptical argument games for preferred semantics. In *Proc. 7th European Workshop on Logic for Artificial Intelligence*, 239–253, 2000.