On the Responsibility for Undecisiveness in Preferred and Stable Labellings in Abstract Argumentation

Claudia Schulz[a,b], Francesca Toni[a]

[a] *Department of Computing, Imperial College London, UK*
[b] *UKP Lab, Technische Universität Darmstadt, Germany*

Abstract

Different semantics of abstract Argumentation Frameworks (AFs) provide different levels of decisiveness for reasoning about the acceptability of conflicting arguments. The stable semantics is useful for applications requiring a high level of decisiveness, as it assigns to each argument the label "accepted" or the label "rejected". Unfortunately, stable labellings are not guaranteed to exist, thus raising the question as to which parts of AFs are responsible for the non-existence. In this paper, we address this question by investigating a more general question concerning preferred labellings (which may be less decisive than stable labellings but are always guaranteed to exist), namely why a given preferred labelling may not be stable and thus undecided on some arguments. In particular, (1) we give various characterisations of parts of an AF, based on the given preferred labelling, and (2) we show that these parts are indeed responsible for the undecisiveness if the preferred labelling is not stable. We then use these characterisations to explain the non-existence of stable labellings. We present two types of characterisations, based on labellings that are more (or equally) committed than the given preferred labelling on the one hand, and based on the structure of the given AF on the other, and compare the respective AF parts deemed responsible. To prove that our characterisations indeed yield responsible parts, we use a notion of enforcement of labels through structural revision, by means of which the preferred labelling of the given AF can be turned into a stable labelling of the structurally revised AF. Rather than prescribing how this structural revision is carried out, we focus on the enforcement of labels and leave the engineering of the revision open to fulfil differing requirements of applications and information available to users.

Keywords: Abstract Argumentation, Stable Semantics, Preferred Semantics, Non-Existence, Undecidedness, Inconsistency, Dynamics, Debugging

Email addresses: schulz@ukp.informatik.tu-darmstadt.de (Claudia Schulz), f.toni@imperial.ac.uk (Francesca Toni)

1. Introduction

Argumentation formalisms have been widely studied for representing arguments and conflicts between these arguments, and for evaluating which sets of arguments should be accepted by resolving the conflicts. An important application area of such formalisms is in decision support, where decisions are made based on an exchange of arguments and an evaluation of their acceptability, see for example [1, 2, 3, 4, 5, 6, 7, 8, 9].

One of the most prominent formalisms is Dung's (abstract) *Argumentation Framework* (AF) [10], which assumes as given a set of arguments, i.e. abstract entities that can represent anything desired by users, and attacks between these arguments. AFs are equipped with different semantics, defining which arguments should be deemed acceptable. They can be defined in terms of acceptable sets of arguments, so called *extensions* [10], or equivalently in terms of *labellings* [11, 12], which assign one of the labels `in` (accepted), `out` (rejected), or `undec` (undecided) to each argument. Extensions coincide with the set of all arguments labelled `in` by a corresponding labelling. Different semantics impose different restrictions on labellings and extensions. Concerning labellings in particular, each argument needs to be *legally* labelled, where an `in`-labelled argument is legally labelled if all arguments attacking it are labelled `out`, an `out`-labelled argument is legally labelled if at least one argument attacking it is labelled `in`, and an `undec`-labelled argument is legally labelled if at least one argument attacking it is labelled `undec` and no argument attacking it is labelled `in` [11, 12].

In many applications, it is desirable to choose a highly decisive semantics, in other words, a semantics that assigns the label `in` or the label `out` to as many arguments as possible. Compared to less decisive semantics, this means greater certainty about the acceptance status of arguments for the user. In particular, the *preferred semantics* assigns the label `in` to a maximal set of arguments (w.r.t. set inclusion). If the union of all `in`- and `out`-labelled arguments in a preferred labelling is maximal among all preferred labellings (w.r.t. set inclusion), the labelling is semi-stable [13]. Even more decisively, if all arguments in a preferred labelling are labelled `in` or `out`, the labelling is *stable*. In applications requiring decisiveness, e.g. in medical or legal scenarios, it is desirable to have at least one stable labelling. Unfortunately, stable labellings are not guaranteed to exist, that is, in some cases all preferred labellings may comprise arguments labelled `undec`.

As an illustration, consider the following example from the medical domain[1], represented graphically as an AF in Figure 1, where nodes are arguments and directed edges are attacks.[2] A physician needs to decide which therapy amongst

[1] A similar example can be found, e.g., in the legal domain as presented in [14].

[2] The example is inspired by the framework of [15] for representing and synthesising knowledge from medical evidence. Whereas in [15] arguments are preferences over treatments, supported by evidence, here, for simplicity of presentation, they are in support or against the effectiveness of treatments.

five possible therapies to recommend to her patient. She first reads a study praising therapy A and concluding that therapy A is way more effective than therapy B. This study thus provides an argument for the effectiveness of therapy A and positions it as a counterargument against any argument stating that therapy B is effective and should be chosen. In Figure 1, this is indicated by the attack from argument "A is effective" to argument "B is effective", which the physician obtains reading a second article. This second article recommends therapy B, showing that it is more reliable than therapy C and much more effective than therapy D. The physician reviews a third study, which describes the enormous success of therapy C and the poor performance of therapy A compared to C. Another article advocates therapy D somewhat incoherently, providing within the same study evidence against the effectiveness of this therapy. Therefore, the argument "D is effective" in Figure 1 attacks itself. Finally, a fifth article discusses therapy E, providing evidence against its effectiveness.

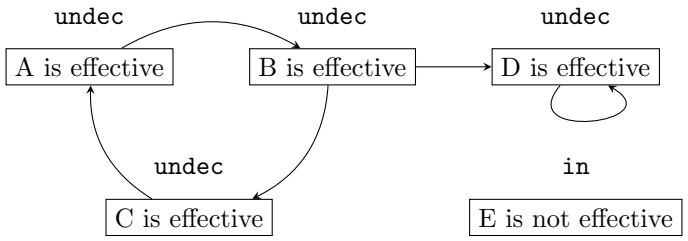

Figure 1: AF representing the physician's information about therapies according to information from scientific articles and the AF's only preferred (and semi-stable) labelling $LabArg_{pref}$.

The resulting AF, representing the physician's information on the effectiveness of the five therapies, has a single preferred (and semi-stable) labelling but no stable labelling. Thus, using the stable semantics, no therapy can be recommended. The preferred (and semi-stable) labelling, referred to as $LabArg_{pref}$ in the remainder of this introduction and illustrated in Figure 1, labels all arguments as `undec` except for argument "E is not effective", which is labelled `in`. Thus, using the preferred (or semi-stable) semantics, the physician can draw the conclusion that therapy E is definitely not effective but still cannot make any decision as to which therapy to prescribe. Thus, the non-existence of stable labellings and the undecisiveness of preferred labellings are closely connected problems.

In this paper we address the problem of non-existence of stable labellings as a by-product of identifying, for a chosen non-stable preferred labelling of a given AF, which parts of the AF can be deemed responsible that this preferred labelling is not stable. Our mechanisms for identifying parts responsible that a chosen preferred labelling is not stable can be seen as means to move from a "partially undecided" preferred labelling to a "fully decided" stable labelling in a rational way. Note that the problem of identifying parts responsible that a chosen preferred labelling is not stable, is interesting in its own right, even if the AF admits stable labellings. Indeed, preferred, non-stable labellings will differ

from stable labellings in their assignment of `in` and `out` labels, so an argument may be labelled `in` in a stable labelling and `out` in a non-stable preferred one. If the user has a preference for the assignment in the latter, but needs to be fully decisive, then understanding why the preferred labelling is not stable is important.

Naively, the set of all `undec` arguments may be deemed responsible if a preferred labelling is not stable, since there are no `undec` arguments in a stable labelling. We show that, in general, only a subset of `undec` arguments is in fact responsible in general. We propose two different characterisation approaches identifying such sets of responsible arguments: a labelling-based approach and a structural approach.

In the labelling-based approach, we give characterisations of responsible parts in terms of sets of arguments labelled `undec` by the chosen preferred labelling and *illegally* labelled if all `undec` labels are changed to `in` or `out`. As an example, consider the AF in Figure 1 and its only preferred labelling $LabArg_{pref}$. Figure 2 illustrates a re-labelling of the AF, where all arguments labelled `undec` by $LabArg_{pref}$ are re-labelled as `in` or `out`. This may reflect the physician's intuition about the effectiveness of the different therapies or her belief in the truth of the different studies. For example, she may know that the authors of the first study cannot be trusted, whereas those of the second study work in an exemplary scientific manner, leading to the labels illustrated in Figure 2. Only the argument "A is effective" is illegally labelled by the new labelling and is thus deemed responsible by our labelling-based approach that $LabArg_{pref}$ is not a stable labelling, and consequently that no stable labelling exists.

In contrast, in the structural approach we characterise responsible parts as initial strongly connected components (SCCs) [16] of the AF restricted to arguments labelled `undec` by the chosen preferred labelling. We call such parts *strongly connected* `undec` *parts* (SCUPs) and prove that they always comprise an odd-length cycle of attacking arguments. The only SCUP of the AF in Figure 1 is the cycle of arguments about therapies A, B, and C, so the set of these three arguments is deemed responsible by our structural approach that $LabArg_{pref}$ is not a stable labelling, and consequently that no stable labelling exists.

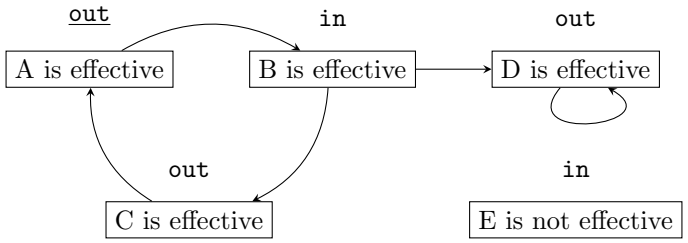

Figure 2: AF representing the physician's information, where `undec` labels from the preferred labelling are replaced by `in` or `out` labels (underlined labels are illegal).

We define *responsibility* that a chosen preferred labelling is not stable in

terms of parts of the AF that require changes in order to turn the `undec` labels into legal `in` or `out` labels. This necessarily requires some structural revision of those responsible parts. Importantly, the exact structural change is not of interest here; instead, the engineering of the revision is left open to fulfil differing requirements of applications and information available to users. We therefore focus on the change of labels from `undec` to `in` or `out` and the fact that *enforcing*[3] the label through some structural revision makes this label legal (in the structurally revised AF).

For instance, in our example we determined that one of the sets of arguments deemed responsible by our labelling-based characterisation comprises only the argument "A is effective", since it is illegally labelled `out` by the re-labelling in Figure 2. Enforcing the label `out` for this argument leads to "A is effective" being *legally* labelled `out`. This could be achieved, e.g., by adding a new argument attacking the argument about A, as illustrated in Figure 3. The new argument may be additional evidence found by the physician, concluding that therapy A is not effective at all. In this paper, however, we are interested in the existence of some structural revision rather than the exact nature of the structural change.

Using this notion of enforcement of labels, we prove that our labelling-based characterisations yield exactly those sets of arguments (a) that need to be enforced and (b) that are sufficient to enforce in order to ensure that all arguments are legally labelled (in the structurally revised AF). Since our labelling-based characterisations thus provide necessary and sufficient conditions for turning a non-stable into a stable labelling (through structural revision), the characterised parts of the AF can be deemed responsible that the given preferred labelling is not stable.

Within our structural approach, the characterisation in terms of SCUPs give a necessary condition for turning a non-stable into a stable labelling (through structural revision). We furthermore show that iteratively enforcing arguments in SCUPs gives a sufficient condition for turning a non-stable into a stable labelling (through structural revision). SCUPs can thus be deemed responsible that the given preferred labelling is not stable.

Note that our labelling-based characterisations are defined with respect to any preferred labelling. For preferred labellings that are stable, the empty set of arguments is the only "responsible" set identified by our characterisations. We can therefore show that an AF has no stable labelling if and only if, with respect to all preferred labellings, there exists a non-empty set of arguments identified as the responsible part of the AF.

The paper is organised as follows. We provide background on AFs in Section 2 and give some preliminary definitions used throughout this paper in Section 3. In Section 4, we define our labelling-based characterisations of parts of an AF and prove that two of them are indeed responsible if a given preferred

[3]Baumann and Brewka [17] introduce the term "enforcement" as a structural change of an AF that makes a desired set of arguments an extension. We here use the term differently, to refer to a structural change that makes desired *labels* of arguments *legal*.

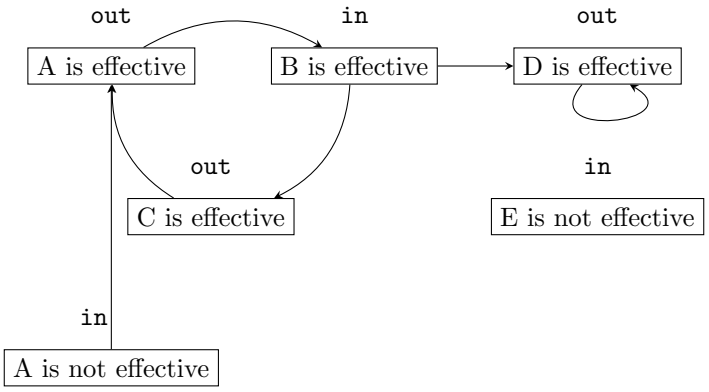

Figure 3: AF representing the physician's information, where the previously illegal `out`-label of the argument about therapy A is enforced by adding a new argument.

labelling is not stable since they provide necessary and sufficient conditions for turning the preferred labelling into a stable labelling through a structural revision. In Section 5, we introduce our structural characterisations of responsible parts: odd-length cycles of attacking arguments, specific strongly connected components (SCCs), and SCUPs. The latter are indeed responsible if the given preferred labelling is not stable as they provide a sufficient condition for turning a non-stable preferred labelling into a stable one (of the structural revision). We then investigate the relation between our labelling-based and structural characterisations in Section 6. In Section 7, we discuss how our definitions characterise the non-existence of stable labellings, explain some of the design choices underlying our approach, and compare our to related work. In Section 8, we conclude and discuss future work.

2. Background

An *argumentation framework* (AF) [10] is a pair $\mathcal{AF} = \langle Ar, Att \rangle$, where Ar is a (finite) set of *arguments* and $Att \subseteq (Ar \times Ar)$ is a set of *attacks* between them. We say that argument A *attacks* argument B, or equivalently that B *is attacked by* A, if and only if $(A, B) \in Att$. A set of arguments $Args \subseteq Ar$ attacks a set of arguments $Args' \subseteq Ar$ if and only if there exist arguments $A \in Args$ and $B \in Args'$ such that $(A, B) \in Att$. We denote by $parents(Args)$ the set of all arguments that are not contained in $Args$ and attack $Args$, i.e. $parents(Args) = \{A \in Ar \setminus Args \mid (A, B) \in Att, B \in Args\}$.

Example 1. Let $\mathcal{AF}_1 = \langle \{a, b, c\}, \{(a, b), (b, b), (b, c), (c, b)\} \rangle$, which is illustrated as a graph in Figure 4 (with arguments as nodes and attacks as directed edges). For the set of arguments $\{b\}$, $parents(\{b\}) = \{a, c\}$.

For the rest of this paper, we assume as given a fixed but arbitrary argumentation framework $\mathcal{AF} = \langle Ar, Att \rangle$, unless specified otherwise.

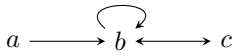

Figure 4: $\mathcal{AF}_1$ from Example 1.

The semantics of an AF are originally defined in terms of *extensions* [10], i.e. sets of "accepted" arguments, which are able to defend themselves against all attacking arguments. The semantics can be equivalently expressed in terms of labellings [11, 12], which we will use here.

A *labelling* of $\mathcal{AF}$ is a total function $LabArg : Ar \longrightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$. We will sometimes refer to a labelling regarding the set of arguments it labels rather than the AF, for example $LabArg$ is a *labelling of* Ar. We denote the set of all arguments labelled `in` by $LabArg$ as $\texttt{in}(LabArg)$, i.e. $\texttt{in}(LabArg) = \{A \in Ar \mid LabArg(A) = \texttt{in}\}$, and the sets of arguments labelled `out` and `undec` as $\texttt{out}(LabArg)$ and $\texttt{undec}(LabArg)$, respectively. We call a labelling with $\texttt{undec}(LabArg) = \emptyset$ an `in-out` *labelling*.

Given a labelling $LabArg$ of $\mathcal{AF}$ and an argument $A \in Ar$:

- A is *legally labelled* `in` by $LabArg$ if and only if $A \in \texttt{in}(LabArg)$ and $\forall B \in Ar$ attacking A it holds that $B \in \texttt{out}(LabArg)$;
- A is *legally labelled* `out` by $LabArg$ if and only if $A \in \texttt{out}(LabArg)$ and $\exists B \in Ar$ attacking A such that $B \in \texttt{in}(LabArg)$;
- A is *legally labelled* `undec` by $LabArg$ if and only if $A \in \texttt{undec}(LabArg)$ and $\exists B \in Ar$ attacking A such that $B \in \texttt{undec}(LabArg)$, and $\forall C \in Ar$ attacking A it holds that $C \notin \texttt{in}(LabArg)$.

A is *legally labelled* by $LabArg$ if and only if it is legally labelled `in`, `out`, or `undec` by $LabArg$; otherwise A is *illegally labelled* by $LabArg$. Equivalently we say that a label *is legal/illegal* w.r.t. $LabArg$.

A labelling $LabArg$ of $\mathcal{AF}$ is a *complete labelling* of $\mathcal{AF}$ if and only if all arguments $A \in Ar$ are legally labelled by $LabArg$. A complete labelling $LabArg$ of $\mathcal{AF}$ is

- a *stable labelling* of $\mathcal{AF}$ if and only if it is an `in-out` labelling;
- a *preferred labelling* of $\mathcal{AF}$ if and only if $\texttt{in}(LabArg)$ is maximal (w.r.t. $\subseteq$) among all complete labellings.

Example 2. $\mathcal{AF}_1$ (see Figure 4) has one complete labelling, namely $LabArg = \{(a, \texttt{in}), (b, \texttt{out}), (c, \texttt{in})\}$, which is also the only preferred and only stable labelling.

Given a set of arguments $Args \subseteq Ar$, $\mathcal{AF}{\downarrow_{Args}} = \langle Args, Att_{Args} \rangle$ denotes the *restriction of* $\mathcal{AF}$ *to* $Args$, where $Att_{Args} = Att \cap (Args \times Args)$. Furthermore, given a labelling $LabArg$ of $\mathcal{AF}$, $LabArg{\downarrow_{Args}} = LabArg \cap (Args \times \{\texttt{in}, \texttt{out}, \texttt{undec}\})$ denotes the *restriction of* $LabArg$ *to* $Args$ [18].

Example 3. Given the set of arguments $\{a, b\}$, $\mathcal{AF}_1{\downarrow}_{\{a,b\}}$ is depicted in Figure 5 along with the labelling $LabArg{\downarrow}_{\{a,b\}}$.

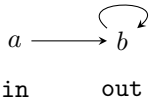

Figure 5: $\mathcal{AF}_1{\downarrow}_{\{a,b\}}$ and the labelling $LabArg{\downarrow}_{\{a,b\}}$.

A *path* from argument $A \in Ar$ to argument $B \in Ar$ is a sequence of arguments $A_0, A_1, \ldots, A_n$ ($n > 0$, $\forall i \in \{0, \ldots, n\} : A_i \in Ar$) with $A_0 = A$ and $A_n = B$ such that $\forall i \in \{0, \ldots, n-1\} : A_i$ attacks A_{i+1}. A *cycle* is a path $A_0, A_1, \ldots, A_n$ where $A_n = A_0$. It is an *odd-length* cycle if and only if n is odd. With an abuse of notation, we denote a cycle as a set of arguments $\mathscr{C}$, where $A_i \in \mathscr{C}$ if and only if A_i occurs in the cycle.

Path-equivalence between two arguments $A \in Ar$ and $B \in Ar$ holds if and only if $A = B$ or there exists a path both from A to B and from B to A. The equivalence classes of arguments under the relation of path-equivalence are called *strongly connected components* (SCCs) of $\mathcal{AF}$ [16]. Since SCCs are sets of arguments, the notion of attacks between sets of arguments can be straightforwardly lifted to a notion of attacks between SCCs. Given an SCC $Args \subseteq Ar$, the set of *parent SCCs* is $parentSCCs(Args) = \{Args' \subseteq Ar \mid Args' \text{ is an SCC of } \mathcal{AF}, Args' \cap parents(Args) \neq \emptyset\}$. If $parentSCCs(Args) = \emptyset$, then $Args$ is an *initial SCC*. Furthermore, the set of *ancestor SCCs* of $Args$ is

$$ancestorSCCs(Args) = parentSCCs(Args) \cup \bigcup_{Args' \in parentSCCs(Args)} ancestorSCCs(Args').$$

Example 4. $\mathcal{AF}_1$ (see Figure 4) has one odd-length cycle, namely $\{b\}$, and two SCCs, namely $\{a\}$ and $\{b, c\}$, where the former attacks the latter. $parentSCCs(\{a\}) = ancestorSCCs(\{a\}) = \emptyset$, so $\{a\}$ is an initial SCC, and $parentSCCs(\{b, c\}) = ancestorSCCs(\{b, c\}) = \{a\}$.

An *argumentation framework (AF) with input* [18] is a tuple

$$\mathcal{AF}_I = (\mathcal{AF}, I, LabArg_I, Att_I)$$

where I is a set of *input* arguments such that $I \cap Ar = \emptyset$, $LabArg_I$ is the *input labelling* of I (i.e. a labelling of I), and Att_I is an attack relation between I and Ar, i.e. $Att_I \subseteq (I \times Ar)$. We say that argument $A \in I$ *attacks* argument $B \in Ar$ if $(A, B) \in Att_I$.

The semantics of an AF with input is defined as follows. A labelling $LabArg$ of $\mathcal{AF}$ is a *complete labelling w.r.t.* $\mathcal{AF}_I$ if and only if for all $A \in Ar$ it holds that[4]:

[4]Baroni et al. [18] call this the "canonical local function" of the complete semantics.

- if $A \in \texttt{in}(LabArg)$, then $\forall B \in Ar \cup I$ attacking A it holds that $B \in \texttt{out}(LabArg) \cup \texttt{out}(LabArg_I)$;
- if $A \in \texttt{out}(LabArg)$, then $\exists B \in Ar \cup I$ attacking A such that $B \in \texttt{in}(LabArg) \cup \texttt{in}(LabArg_I)$;
- if $A \in \texttt{undec}(LabArg)$, then $\exists B \in Ar \cup I$ attacking A such that $B \in \texttt{undec}(LabArg) \cup \texttt{undec}(LabArg_I)$, and $\forall B \in Ar \cup I$ attacking A it holds that $B \notin \texttt{in}(LabArg) \cup \texttt{in}(LabArg_I)$.

A labelling $LabArg$ of $\mathcal{AF}$ is a *stable labelling w.r.t.* $\mathcal{AF}_I$ if and only if $LabArg$ is a complete labelling w.r.t. $\mathcal{AF}_I$ and $\texttt{undec}(LabArg) = \emptyset$. We sometimes say that $LabArg$ is a complete/stable labelling *of* $\mathcal{AF}$ *w.r.t. the input* I.

Example 5. An AF with input $(\mathcal{AF}_1, I, LabArg_I, Att_I)$ is depicted in Figure 6, where the set of input arguments is $I = \{a', b'\}$, the labelling of input arguments is $LabArg_I = \{(a', \texttt{in}), (b', \texttt{undec})\}$, and $Att_I = \{(a', a)\}$. There are two complete labellings w.r.t. $(\mathcal{AF}_1, I, LabArg_I, Att_I)$, namely
$\{(a, \texttt{out}), (b, \texttt{undec}), (c, \texttt{undec})\}$ and $\{(a, \texttt{out}), (b, \texttt{out}), (c, \texttt{in})\}$,
where the latter is a stable labelling w.r.t. $(\mathcal{AF}_1, I, LabArg_I, Att_I)$.

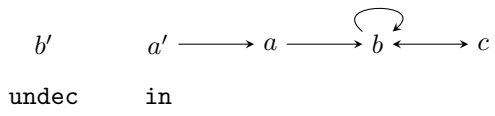

Figure 6: The AF with input from Example 5.

3. Preliminaries

The aim of this paper is to give characterisations of parts of an AF responsible that a given preferred labelling is not stable. To prove that the characterised parts are in fact responsible, we show that when re-labelling arguments labelled `undec` by the preferred labelling as `in` or `out`, with the aim to obtain a stable labelling, the labels of arguments in the characterised parts have to be *enforced* through some structural revision to ensure their legality. In contrast, labels of arguments not in the characterised parts may not have to be enforced.

Importantly, we are here not interested in the *exact* structural revision, but rather in any (sensible) structural revision of the AF that turns illegally labelled arguments into legally re-labelled ones. In other words, we see structural revisions as a way to "enforce" the legality of labels. Since we are only interested in enforcing the labels of *certain* arguments (usually those re-labelled as `in` or `out` that have been identified as responsible), we restrict structural revisions to a given set of arguments.

The rational behind this enforcement of labels through structural revision is as follows: If a preferred labelling is not stable, our characterisations of responsible sets point the user towards parts of the AF responsible that the acceptability

of some arguments cannot be decided. The user can then further investigate the nature of these arguments and decide which label they should have. The user may for example notice that she expected one of the responsible arguments to be accepted and would thus label it as `in`. To ensure that this argument is legally labelled, the structure of the AF (in particular, the structure of the part including the re-labelled argument) then has to be revised. This revision may be grounded in the user's realisation that the structure of the AF is incorrect, for example, that an important argument is missing (adding an argument to the AF) or that an existing attack should not be present in the AF (deleting an attack).

Following this intuition, we introduce *set-driven* revisions, which ensure that labels (according to some desired labelling) of arguments in a given set $Args$ are legal after structurally revising the part of the AF consisting of $Args$, while not making any structural changes affecting arguments not in $Args$.

Definition 1 (Set-Driven Revision and Revision Labelling). Let $LabArg$ be a labelling of $\mathcal{AF}$ and let $Args \subseteq Ar$. A *(set-driven) revision* of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ is $\mathcal{AF}^{\circledast} = \langle Ar^{\circledast}, Att^{\circledast} \rangle$ such that:

- $Ar \subseteq Ar^{\circledast}$;
- $\{(A, B) \in Att \mid B \in Ar \setminus Args\} = \{(A, B) \in Att^{\circledast} \mid B \in Ar \setminus Args\}$;
- there exists a labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ satisfying that:
  - $\forall C \in Ar$: $LabArg^{\circledast}(C) = LabArg(C)$;
  - $\forall D \in Ar^{\circledast} \setminus Ar$: D is legally labelled `in` or `out` by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$;
  - $\forall E \in Args$: E is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$.

Any such $LabArg^{\circledast}$ is called a *revision labelling* of $\mathcal{AF}^{\circledast}$.

Since a set-driven revision enforces desired labels for arguments in the given set $Args$, we do not allow the deletion of arguments in $Args$. Arguments may thus only be *added* in a set-driven revision (specified by the first bullet in Definition 1). Furthermore, structural changes that may affect (the legality of labels of) arguments *not* in $Args$ are prohibited. Thus, all attacks on arguments not in $Args$ have to remain the same in the revision (specified by the second bullet). It follows, that permitted structural changes are the addition of attacks between arguments in $Args$ or between new arguments and arguments in $Args$, and the deletion of attacks between arguments in $Args$ or from arguments not in $Args$ to arguments in $Args$ (indirectly specified by the second bullet). Since $LabArg$ specifies the desired labels of all arguments, a revision labelling is a simple "enlargement" of $LabArg$ to include (legal) labels of new arguments; the labels of all other arguments remain unchanged (specified by the first and second item of the third bullet). Furthermore, and most importantly, a revision labelling ensures that all arguments in $Args$ are *legally* labelled in the revision (specified by the third item of the third bullet).

From here onwards, we will refer to set-driven revisions simply as *revisions*.

Example 6. Consider the AF in Figure 1 (see page 3), call it $\mathcal{AF}_{therapy}$. From here on, we use a shorthand notation for each argument according to the letter of the respective therapy, e.g. A denotes the argument "A is effective". Let $LabArg$ be the labelling of $\mathcal{AF}_{therapy}$ illustrated in Figure 2 (see page 4). Figure 3 (see page 6) depicts a revision of $\mathcal{AF}_{therapy}$ w.r.t. $\{A\}$ by $LabArg$, which we denote $\mathcal{AF}^{\circledast}_{therapy}$, and the labelling in Figure 3 is a revision labelling of $\mathcal{AF}^{\circledast}_{therapy}$. Note that $\mathcal{AF}^{\circledast}_{therapy}$ is also a revision of $\mathcal{AF}_{therapy}$ w.r.t. any superset of $\{A\}$ by $LabArg$.

As shown by the following lemma, a revision exists for any given set of arguments and any labelling.

Lemma 1. *Let $LabArg$ be a labelling of $\mathcal{AF}$ and let $Args \subseteq Ar$. Then there exists a revision of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$.*

The proof of this lemma and of most other lemmas and propositions presented throughout this paper can be found in Appendix B.

Note that, in this paper, we are not concerned with the *exact* structural change of a revision compared to the original AF. We simply use the structural change of an AF as a tool to ensure that labels of arguments are legal. As a result, there may be various revisions of an AF w.r.t. a given set of arguments and labelling. Furthermore, a revision may have various different revision labellings. It is in general up to the preference of users and the application scenario to decide which of these revisions and revision labellings to use. For example, a user may be interested in revisions with "minimal" structural changes as in [19, 20].

Example 7. Let $\mathcal{AF}_2$ be the AF depicted on the left of Figure 7 and $LabArg$ the labelling of $\mathcal{AF}_2$ illustrated on the left of Figure 7, which is the labelling we desire. Argument a is illegally labelled by $LabArg$, so a revision can be used to enforce the desired label for argument a. A possible revision of $\mathcal{AF}_2$ w.r.t. $\{a\}$ by $LabArg$ is illustrated on the right of Figure 7, alongside a revision labelling. Another revision of $\mathcal{AF}_2$ w.r.t. $\{a\}$ by $LabArg$ is illustrated in Figure 8, alongside two different revision labellings.

Figure 7: Left – $\mathcal{AF}_2$ and a labelling $LabArg$, where the underline indicates that the argument is illegally labelled. Right – A revision of $\mathcal{AF}_2$ and its only revision labelling (see Example 7).

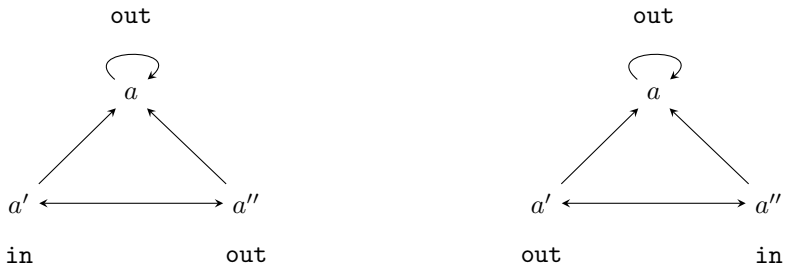

Figure 8: A revision of $\mathcal{AF}_2$, which has two different revision labellings (see Example 7).

Next, we extend the comparison notion of *commitment* of two labellings of an AF [11] to the comparison of labellings of potentially different AFs, where the arguments of one AF form a superset of the arguments of the other.

Definition 2 (Commitment of Labellings). Let $LabArg$ be a labelling of $\mathcal{AF}$ and let $LabArg'$ be a labelling of $\mathcal{AF}' = \langle Ar', Att' \rangle$, where $Ar \subseteq Ar'$.

- $LabArg'$ is *more or equally committed* than $LabArg$, denoted $LabArg \sqsubseteq LabArg'$, if and only if $\texttt{in}(LabArg) \subseteq \texttt{in}(LabArg')$, $\texttt{out}(LabArg) \subseteq \texttt{out}(LabArg')$ and $\texttt{undec}(LabArg') \subseteq \texttt{undec}(LabArg)$.
- $LabArg'$ is *more committed* than $LabArg$, denoted $LabArg \sqsubset LabArg'$, if and only if $LabArg \sqsubseteq LabArg'$ and $\texttt{undec}(LabArg') \subset \texttt{undec}(LabArg)$.

We note that a revision labelling is more or equally committed than the original labelling.

Observation 2. *Let $LabArg$ be a labelling of $\mathcal{AF}$ and $Args \subseteq Ar$. Then, for all revisions $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and all revision labellings $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$, it holds that $LabArg \sqsubseteq LabArg^{\circledast}$.*

For instance, the two revision labellings of the revision of $\mathcal{AF}_2$ illustrated in Figure 8 (see Example 7) are more committed than the original labelling of $\mathcal{AF}_2$, depicted on the left of Figure 7.

Notation 1. In the remainder of this paper, and if not stated otherwise, we assume as given a fixed but arbitrary argumentation framework $\mathcal{AF} = \langle Ar, Att \rangle$ and a preferred labelling $LabArg_{pref}$ of $\mathcal{AF}$.

4. Labelling-Based Characterisations

As previously explained, we aim to 1) characterise parts of an AF responsible that a chosen preferred labelling is not stable and 2) prove the responsibility of these parts by showing that, in order to obtain a stable labelling that is more committed than this preferred labelling, `in` or `out` labels have to be enforced for arguments in these responsible parts.

In this section, we give three declarative characterisations of sets of arguments responsible if $LabArg_{pref}$ is not a stable labelling. All characterisations identify the empty set as responsible if and only if $LabArg_{pref}$ is stable. These characterisations are *labelling-based*, which means that they identify responsible sets based on labellings that are more or equally committed than $LabArg_{pref}$. We also investigate our characterisations in the light of revisions of the AF that (do not) have a stable labelling that is more committed than $LabArg_{pref}$. In particular, we show that our two non-naïve characterisations, which we introduce in Sections 4.2 and 4.3, define *necessary and sufficient* conditions for the existence and non-existence of a more committed stable labelling of a revision.

4.1. The Basic Approach

A naïve way to characterise arguments responsible that $LabArg_{pref}$ is not a stable labelling is in terms of *all* arguments labelled `undec` by $LabArg_{pref}$, since these are the arguments violating the definition of a stable labelling.

Definition 3 (Labelling-Based Characterisation 1). $\texttt{undec}(LabArg_{pref})$ is the *labelling-based responsible set* w.r.t. $LabArg_{pref}$.

Trivially, if $LabArg_{pref}$ is a stable labelling, the *labelling-based responsible set* is the empty set.

It is straightforward that this characterisation provides a sufficient condition for turning a non-stable preferred labelling into a stable labelling (of a revision), as stated in the following proposition. That is, a labelling-based responsible set comprises all those arguments whose labels definitely need to be enforced in order to obtain a stable labelling of a revision, i.e. the responsible arguments are included in this set. However, (since this characterisation gives only a sufficient but not a necessary condition) a labelling-based responsible set may also comprise arguments whose labels do not necessarily need to be enforced, i.e. non-responsible arguments.

Proposition 3. *Let Args be the labelling-based responsible set w.r.t.* $LabArg_{pref}$ *and let LabArg be a labelling of* $\mathcal{AF}$ *such that* $LabArg_{pref} \sqsubseteq LabArg$ *and* $\texttt{undec}(LabArg) = \emptyset$*. Then, for all revisions* $\mathcal{AF}^{\circledast}$ *of* $\mathcal{AF}$ *w.r.t. Args by LabArg and all revision labellings* $LabArg^{\circledast}$ *of* $\mathcal{AF}^{\circledast}$*,* $LabArg^{\circledast}$ *is a stable labelling of* $\mathcal{AF}^{\circledast}$*.*

Example 8. Consider again $\mathcal{AF}_{therapy}$ from Example 6 (see page 11) and its only preferred labelling $LabArg_{pref}$, which labels all arguments `undec` except for argument E, which is labelled `in`. Thus, the labelling-based responsible set w.r.t. $LabArg_{pref}$ is $\{A, B, C, D\}$. Let $LabArg$ be the labelling of $\mathcal{AF}_{therapy}$ illustrated in Figure 2 (see page 4). The revision labelling of the revision $\mathcal{AF}^{\circledast}_{therapy}$ of $\mathcal{AF}_{therapy}$ w.r.t. $\{A, B, C, D\}$ by $LabArg$ (see Figure 3, page 6) is a stable labelling of $\mathcal{AF}^{\circledast}_{therapy}$.

Since by Lemma 1 a revision exists w.r.t. any set of arguments and any labelling, it follows that there exists a revision w.r.t. the labelling-based responsible set, and in particular (by Proposition 3) a revision that has a stable labelling.

Corollary 4. *Let $Args$ be the labelling-based responsible set w.r.t. $LabArg_{pref}$ and let $LabArg$ be a labelling of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$. Then there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$.*

Note that, by Observation 2, a stable labelling obtained through such a revision is more committed than $LabArg_{pref}$ (if $LabArg_{pref}$ is not stable). Thus, as desired, the labelling-based responsible set can be used to turn a non-stable preferred labelling into a stable labelling (of a revision).

4.2. Enforcement Sets

The definition of labelling-based responsible set is a rather naïve characterisation of arguments responsible if a preferred labelling is not stable, since it is often possible to legally label some of its arguments as `in` or `out`. For example, considering the arguments A, B, C, and D labelled `undec` by the preferred labelling of $\mathcal{AF}_{therapy}$ (see Figure 1, page 3), we observe that three out of these four arguments can in fact be legally labelled `in` or `out`, as illustrated in Figure 2 (see page 4).

Our next characterisation takes this observation into account, characterising specific subsets of the labelling-based responsible set as responsible. In particular, arguments that are legally labelled by an `in-out` labelling that is more or equally committed than $LabArg_{pref}$ will not be deemed responsible. More precisely, our second labelling-based characterisation defines a *minimal* set of arguments labelled `undec` by $LabArg_{pref}$ satisfying that some `in-out` labelling that is more or equally committed than $LabArg_{pref}$ legally labels all non-responsible arguments (i.e. all arguments not contained in this set).

Definition 4 (Labelling-Based Characterisation 2). $Args \subseteq Ar$ is an *enforcement set* w.r.t. $LabArg_{pref}$ if and only if it is a minimal set of arguments (w.r.t. $\subseteq$) such that

$Args \subseteq \texttt{undec}(LabArg_{pref})$

and $\exists LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$

such that $\forall A \in \texttt{undec}(LabArg_{pref}) \setminus Args$: A is legally labelled by $LabArg$.

Any such $LabArg$ is an *enforcement labelling* w.r.t. $Args$.

Example 9. Consider $\mathcal{AF}_{therapy}$ and its only preferred labelling $LabArg_{pref}$ (see Example 8 on page 13 and Figure 1 on page 3). Then $\{A\}$ is an enforcement set w.r.t. $LabArg_{pref}$, where the labelling shown in Figure 2 (page 4) is an enforcement labelling, as it is an `in-out` labelling that is more committed than $LabArg_{pref}$ and it legally labels all arguments labelled `undec` by $LabArg_{pref}$

except for argument A (i.e. arguments B, C, and D). Furthermore, $\{A\}$ is a minimal set satisfying this condition, since for its only subset $\{\}$ there exists no `in-out` labelling that is more committed than $LabArg_{pref}$ and that legally labels *all* arguments labelled `undec` by $LabArg_{pref}$. There are two more enforcement sets w.r.t. $LabArg_{pref}$, namely $\{B\}$ and $\{C\}$. Note that $\{D\}$ is not an enforcement set since there exists no `in-out` labelling that legally labels A, B, and C. Furthermore, no superset of $\{D\}$ is an enforcement set, as no such superset fulfils the minimality condition.

In Example 9, all enforcement sets are disjoint. The following example illustrates that different enforcement sets may contain the same arguments and that an enforcement set may have various different enforcement labellings.

Example 10. Let $\mathcal{AF}_3$ be the AF on the left of Figure 9, whose only preferred labelling $LabArg_{pref}$ labels all arguments as `undec`. There are three enforcement sets w.r.t. $LabArg_{pref}$: $\{a, e\}$, $\{b, e\}$, and $\{c, e\}$. Note that for all of them various enforcement labellings exist, e.g. the labelling illustrated on the left of Figure 9 is an enforcement labelling of $\{b, e\}$, and so is $\{(a, \texttt{out}), (b, \texttt{out}), (c, \texttt{in}), (d, \texttt{in}), (e, \texttt{in})\}$ (among others).

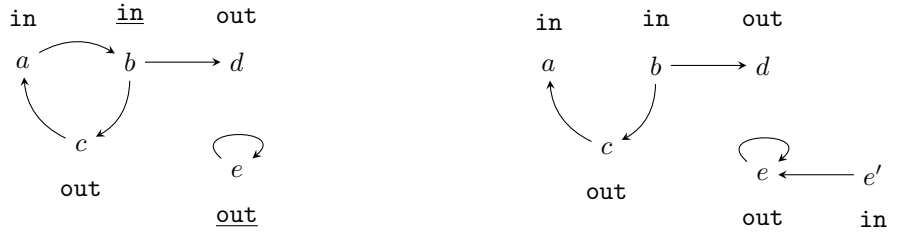

Figure 9: Left – $\mathcal{AF}_3$ and labelling $LabArg$, where underlined labels are illegal. Right – A revision $\mathcal{AF}_3^{\circledast}$ of $\mathcal{AF}_3$ by $LabArg$ and a revision labelling that is a stable labelling of $\mathcal{AF}_3$ (see Examples 10 and 11).

It follows from Definition 4 that all arguments in an enforcement set are illegally labelled by an enforcement labelling. For example, arguments b and e are illegally labelled by both enforcement labellings discussed in Example 10. Still, we note that an enforcement set is defined as a minimal set of arguments such that all arguments *not in* this set are *legally* labelled. An alternative definition of enforcement set as a minimal set of arguments such that all arguments *in* this set are *illegally* labelled is however not equivalent as it would always yield the empty set as the only enforcement set (w.r.t. any enforcement labelling). Clearly, the empty set, and thus the alternative definition, would not be helpful in characterising parts of an AF responsible that a given preferred labelling is not stable.

In the following lemma, we show that at least one enforcement set exists and that enforcement sets are always non-empty if $LabArg_{pref}$ is not a stable

labelling. Both are important properties for sets of arguments characterising parts of an AF responsible that a preferred labelling is not stable.

Lemma 5. *Enforcement sets have the following properties:*

1. *There exists an enforcement set w.r.t.* $LabArg_{pref}$.
2. $Args = \emptyset$ *is an enforcement set w.r.t.* $LabArg_{pref}$ *if and only if* $LabArg_{pref}$ *is a stable labelling.*

Note that if $Args = \emptyset$ is an enforcement set w.r.t. $LabArg_{pref}$, it is the *only* enforcement set w.r.t. $LabArg_{pref}$ as it is the minimal set satisfying Definition 4, where $LabArg_{pref}$ is the only enforcement labelling.

4.2.1. Responsibility of Enforcement Sets

The reason for naming our second labelling-based characterisation "enforcement sets" is illustrated by Theorem 6: "enforcing" the labels of an enforcement labelling for arguments in an enforcement set in terms of a revision results in a stable labelling (of a revision). An enforcement set is thus a *sufficient* condition for obtaining a stable labelling through a revision, which is more refined than the condition given by the labelling-based responsible set (since every enforcement set is a subset of the labelling-based responsible set). This proves that all arguments that are jointly responsible that a preferred labelling is not stable are contained in an enforcement set.

Theorem 6. *Let* $Args \supseteq Args_{enf}$ *where* $Args_{enf}$ *is an enforcement set w.r.t.* $LabArg_{pref}$ *and let* $LabArg$ *be an enforcement labelling w.r.t.* $Args_{enf}$. *Then, for all revisions* $\mathcal{AF}^{\circledast}$ *of* $\mathcal{AF}$ *w.r.t.* $Args$ *by* $LabArg$ *and all revision labellings* $LabArg^{\circledast}$ *of* $\mathcal{AF}^{\circledast}$, $LabArg^{\circledast}$ *is a stable labelling of* $\mathcal{AF}^{\circledast}$.

Proof. Since by Definition 4, $\texttt{undec}(LabArg) = \emptyset$, it follows from Observation 2 that $\texttt{undec}(LabArg^{\circledast}) = \emptyset$. By Definition 1, all $A \in Ar^{\circledast} \setminus Ar$ are legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Let $B \in Ar$. If $B \in Args$, then, by Definition 1, B is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. If $B \notin Args$, and thus B $\notin Args_{enf}$, then by Lemma 39 in Appendix A, B is legally labelled by $LabArg$ in $\mathcal{AF}$, so by Lemma 37 in Appendix A, B is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Since all arguments are legally labelled by $LabArg^{\circledast}$ and $\texttt{undec}(LabArg^{\circledast}) = \emptyset$, $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. □

Example 11. Consider the enforcement set $\{b, e\}$ and the enforcement labelling $LabArg$ of $\mathcal{AF}_3$, illustrated on the left of Figure 9 (see page 15). The AF on the right of Figure 9 is a revision $\mathcal{AF}_3^{\circledast}$ of $\mathcal{AF}_3$ w.r.t. $\{b, e\}$ by $LabArg$ and the revision labelling $LabArg^{\circledast}$ illustrated in the figure is a stable labelling of $\mathcal{AF}_3^{\circledast}$.

Note that Theorem 6 shows that in order to obtain a stable labelling (of a revision), it is sufficient to enforce the label `in` or `out` for *certain* arguments that are labelled `undec` by $LabArg_{pref}$. In particular, it tells us *which* subsets of

undec arguments can be chosen for the enforcement, namely enforcement sets. In general enforcing labels for some (unrestricted) subset of undec arguments may not result in a stable labelling (of a revision). For example, for $\mathcal{AF}_3$, whose only preferred labelling labels all arguments as undec, choosing $\{d\}$ (or even $\{d, e\}$) as a subset of undec arguments and enforcing the labels in or out for these arguments, we do not obtain a stable labelling. If, however, one of the enforcement sets is chosen, as illustrated in Example 11, and labels are enforced appropriately, then a stable labelling (of the revision) is obtained.

Since by Lemma 1 a revision exists w.r.t. any set of arguments and labelling, it follows that there exists a revision w.r.t. an enforcement set by an enforcement labelling and that the revision has a stable labelling that is more committed than $LabArg_{pref}$.

Corollary 7. *Let $Args \supseteq Args_{enf}$ where $Args_{enf}$ is an enforcement set w.r.t. $LabArg_{pref}$ and let $LabArg$ be an enforcement labelling w.r.t. $Args_{enf}$. Then there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$.*

4.3. Preventing Sets

Enforcement sets characterise a responsible set of arguments with respect to a *specific* more committed labelling, which illegally labels *all* arguments in this set. Instead, our second non-naïve characterisation defines a responsible set of arguments as a minimal set containing at least *one* illegally labelled argument with respect to *every* in-out labelling that is more committed than $LabArg_{pref}$.

Definition 5 (Labelling-Based Characterisation 3). $Args \subseteq Ar$ is a *preventing set* w.r.t. $LabArg_{pref}$ if and only if it is a minimal set of arguments (w.r.t. $\subseteq$) such that

$Args \subseteq \texttt{undec}(LabArg_{pref})$
and $\forall LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$
it holds that $\exists A \in Args$ such that A is illegally labelled by $LabArg$.

Example 12. Consider $\mathcal{AF}_{therapy}$ and its only preferred labelling $LabArg_{pref}$ (see Example 8 on page 13 and Figure 1 on page 3). The only preventing set w.r.t. $LabArg_{pref}$ is $\{A, B, C\}$, since no matter how the labels in and out are assigned to this set of arguments, at least one argument is illegally labelled. In contrast, for all subsets there exists some in-out labelling that legally labels all arguments. For instance, for the set $\{A, B\}$, an in-out labelling that labels A as in and B and C as out legally labels both A and B.

Note that in contrast to the definition of enforcement sets, we only consider labellings that are *more* committed than $LabArg_{pref}$ in the definition of preventing sets. This is because for enforcement sets $LabArg_{pref}$ itself can be an enforcement labelling (if it is a stable labelling), whereas for preventing sets $LabArg_{pref}$ needs to be excluded as a labelling having illegally labelled arguments if it is a stable labelling.

As for enforcement sets, at least one preventing set exists w.r.t. $LabArg_{pref}$ and preventing sets are always non-empty if $LabArg_{pref}$ is not a stable labelling.

Lemma 8. *Preventing sets have the following properties:*

1. *There exists a preventing set w.r.t.* $LabArg_{pref}$.
2. $Args = \emptyset$ *is a preventing set w.r.t.* $LabArg_{pref}$ *if and only if* $LabArg_{pref}$ *is a stable labelling.*

Note that, analogously to enforcement sets, if $Args = \emptyset$ is a preventing set w.r.t. $LabArg_{pref}$, it is the only preventing set w.r.t. $LabArg_{pref}$.

4.3.1. Responsibility of Preventing Sets

Theorem 9 illustrates the reason for naming our third labelling-based characterisation "preventing sets": all revisions w.r.t. a set of arguments not comprising any argument from some preventing set have no stable labelling that is more committed than $LabArg_{pref}$. Thus, preventing sets define a *sufficient* condition for "preventing" the existence of a stable labelling that is more committed than $LabArg_{pref}$.

Theorem 9. *Let* $Args \subseteq Ar \setminus Args_{prev}$ *where* $Args_{prev}$ *is a preventing set w.r.t.* $LabArg_{pref}$. *Then, for all labellings* $LabArg$ *of* $\mathcal{AF}$ *such that* $LabArg_{pref} \sqsubset LabArg$ *and* $\texttt{undec}(LabArg) = \emptyset$, *there exists no revision* $\mathcal{AF}^{\circledast}$ *of* $\mathcal{AF}$ *w.r.t.* $Args$ *by* $LabArg$ *such that some revision labelling* $LabArg^{\circledast}$ *of* $LabArg^{\circledast}$ *is a stable labelling of* $\mathcal{AF}^{\circledast}$.

Proof. Assume there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. By Definition 5, $\exists A \in Args_{prev}$ such that A is illegally labelled by $LabArg$ in $\mathcal{AF}$. Since $A \in Ar \setminus Args$, it follows from Lemma 37 in Appendix A that A is illegally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Contradiction. □

Example 13. Recall $\mathcal{AF}_3$, depicted on the left of Figure 9 (see page 15), and its only preferred labelling $LabArg_{pref}$, which labels all arguments as `undec`. There are two preventing sets w.r.t. $LabArg_{pref}$, namely $\{a, b, c\}$ and $\{e\}$. Consider the preventing set $\{e\}$ and some `in-out` labelling of $\mathcal{AF}_3$, e.g. $LabArg$ illustrated on the left of Figure 9. In order to ensure that e is legally labelled by $LabArg$, an attack on e from some argument labelled `in` has to be added (e.g. as illustrated on the right of Figure 9). Conversely, if e was labelled `in` by an `in-out` labelling, the self-attack of e would have to be deleted in order to ensure that e was legally labelled. Thus, no revision w.r.t. a set of arguments not containing e can result in e being legally labelled.

4.4. Enforcement versus Preventing Sets

Theorems 6 and 9 hint at a connection between enforcement and preventing sets: one provides a sufficient condition for the *existence* of a stable labelling after revision, the other a sufficient condition for the *non-existence*. In this section, we investigate the relationship between enforcement and preventing sets in more detail.

We first show that a preventing set is a minimal set containing at least one argument from each enforcement set (if non-empty enforcement sets exist).

Theorem 10. *Let S_{enf} be the set of all enforcement sets w.r.t. $LabArg_{pref}$. Then $S = \{Args \subseteq Ar \mid Args$ is a minimal set satisfying that $\forall Args_{enf} \neq \emptyset \in S_{enf} : Args \cap Args_{enf} \neq \emptyset\}$ is the set of all preventing sets w.r.t. $LabArg_{pref}$.*

PROOF. We prove that all $Args \in S$ are preventing sets and that all preventing sets are contained in S. We note that, by Lemma 5, $S_{enf} \neq \emptyset$. If $S_{enf} = \{\emptyset\}$ then $S = \{\emptyset\}$. By Lemma 5 $LabArg_{pref}$ is a stable labelling and by Lemma 8 the empty set is the only preventing set. If $S_{enf} \neq \{\emptyset\}$ then $\forall Args_{enf} \in S_{enf}$: $Args_{enf} \neq \emptyset$ and $LabArg_{pref}$ is not a stable labelling.

- Let $Args \in S$ and assume that $Args$ is not a preventing set. Then either $Args$ is not a minimal set satisfying the conditions in Definition 5 or $Args$ does not satisfy the conditions at all.
  - In the first case, $\exists Args_{prev} \subset Args$ such that $Args_{prev}$ is a preventing set. Since $Args$ is a minimal set satisfying that $\forall Args_{enf} \in S_{enf} : Args \cap Args_{enf} \neq \emptyset$, it follows that $\exists Args'_{enf} \in S_{enf}$ such that $Args_{prev} \cap Args'_{enf} = \emptyset$. Since $Args'_{enf}$ is an enforcement set there exists an enforcement labelling $LabArg$. By Lemma 39 in Appendix A it holds that $\forall B \in Ar \setminus Args'_{enf}$, B is legally labelled by $LabArg$. Since $Args_{prev}$ is a preventing set it holds that $\exists C \in Args_{prev}$ such that C is illegally labelled by $LabArg$. Contradiction since $C \in Ar \setminus Args'_{enf}$.
  - In the second case, we note that $Args \subseteq \texttt{undec}(LabArg_{pref})$ since $\forall A \in Args : \exists Args_{enf}$ such that $A \in Args_{enf}$ and $Args_{enf} \subseteq \texttt{undec}(LabArg_{pref})$ by Definition 4. Thus, $Args$ violates Definition 5 because $\exists LabArg$ such that $LabArg_{pref} \sqsubset LabArg$, $\texttt{undec}(LabArg) = \emptyset$, and $\forall A \in Args$ it holds that A is legally labelled by $LabArg$. Let $Args' = Ar \setminus Args$. Then, $\forall A \in Ar \setminus Args' = Args$, A is legally labelled by $LabArg$, in particular all $A \in \texttt{undec}(LabArg_{pref}) \setminus Args'$ are legally labelled by $LabArg$. Thus, $Args'$ satisfies the conditions of an enforcement set (disregarding minimality). Since by definition of $Args'$ it holds that $Args \cap Args' = \emptyset$, $Args'$ is not an enforcement set (by definition of $Args$). Thus, $Args'$ is not a minimal set satisfying the conditions of an enforcement set, i.e. $\exists Args_{enf} \in S_{enf}$ such that $Args_{enf} \subset Args'$. Then, by definition of $Args$, it holds $Args \cap Args_{enf} \neq \emptyset$ and thus $Args \cap Args' \neq \emptyset$. Contradiction.

  Thus, $Args$ is a preventing set.

- Let $Args_{prev}$ be a preventing set and assume that $Args_{prev} \notin S$. Then either $\exists Args_{enf} \in S_{enf}$ such that $Args_{prev} \cap Args_{enf} = \emptyset$ or there exists a minimal set $Args \subset Args_{prev}$ satisfying that $Args \cap Args_{enf} \neq \emptyset$ for all $Args_{enf} \in S_{enf}$.
  - In the first case, since $Args_{enf}$ is an enforcement set there exists an enforcement labelling $LabArg$. By Lemma 39 in Appendix A it holds that $\forall A \in Ar \setminus Args_{enf}$, A is legally labelled by $LabArg$. Since

$Args_{prev}$ is a preventing set it holds that $\exists B \in Args_{prev}$ such that B is illegally labelled by $LabArg$. Contradiction since $B \in Ar \setminus Args_{enf}$.

– In the second case, $Args \in S$, so it follows from the first item of this proof that $Args$ is a preventing set. Contradiction since $Args_{prev}$ is a preventing set (and thus minimal).

Thus, $Args_{prev} \in S$. □

Example 14. From Example 10, we know that for $\mathcal{AF}_3$ the set of all enforcement sets is $S_{enf} = \{\{a, e\}, \{b, e\}, \{c, e\}\}$. Then both $\{a, b, c\}$ and $\{e\}$ are minimal sets containing an argument from each enforcement set. Indeed, $\{a, b, c\}$ and $\{e\}$ are the two preventing sets w.r.t. $LabArg_{pref}$ of $\mathcal{AF}_3$ (see Example 13).

Conversely, an enforcement set is a minimal set containing at least one argument from each preventing set (if non-empty preventing sets exist).

Theorem 11. *Let S_{prev} the set of all preventing sets w.r.t. $LabArg_{pref}$. Then $S = \{Args \subseteq Ar \mid Args$ is a minimal set satisfying that $\forall Args_{prev} \neq \emptyset \in S_{prev} : Args \cap Args_{prev} \neq \emptyset\}$ is the set of all enforcement sets w.r.t. $LabArg_{pref}$.*

Proof. Analogous to the proof of Theorem 10, see Appendix B. □

Example 15. From Example 13, we know that $S_{prev} = \{\{a, b, c\}, \{e\}\}$ for $\mathcal{AF}_3$. Then $\{a, e\}$, $\{b, e\}$, and $\{c, e\}$ are all the minimal sets containing an argument from each preventing set. Indeed, these three sets are the enforcement sets of $\mathcal{AF}_3$ w.r.t. $LabArg_{pref}$ (see Example 10 on page 15).

4.5. Necessary Conditions for the (Non-)Existence of Stable Labellings

Based on the correspondence results between enforcement and preventing sets, we now further investigate their responsibility regarding a non-stable preferred labelling $LabArg_{pref}$. We prove that both enforcement and preventing sets define not only sufficient but also *necessary* conditions for the existence and non-existence, respectively, of a stable labelling (of a revision) that is more committed than $LabArg_{pref}$.

Concerning enforcement sets, Theorem 12 proves that in order to obtain a stable labelling (through a revision) that is more committed than $LabArg_{pref}$, the labels of all arguments in some enforcement set have to be enforced for sure. Enforcement sets thus provide a *necessary* condition for obtaining a stable labelling (of a revision) that is more committed than $LabArg_{pref}$. It follows that all arguments in an enforcement set are responsible that the given preferred labelling is not stable, in other words, the enforcement set does not comprise non-responsible arguments.

Theorem 12. *Let $Args \subseteq Ar$ and let $LabArg$ be a labelling of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubseteq LabArg$, $\texttt{undec}(LabArg) = \emptyset$, and there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. Then there exists an enforcement set $Args_{enf}$ w.r.t. $LabArg_{pref}$ such that $Args_{enf} \subseteq Args$.*

PROOF. If $LabArg_{pref}$ is a stable labelling then the only more or equally committed labelling is $LabArg_{pref}$ itself. Thus, for any set $Args \subseteq Ar$ it holds that there exists a revision $\mathcal{AF}^{\circledast} = \mathcal{AF}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg = LabArg_{pref}$ and a revision labelling $LabArg^{\circledast} = LabArg_{pref}$ of $\mathcal{AF}^{\circledast} = \mathcal{AF}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. By Lemma 5, the only enforcement set is the empty set, so it holds that $Args_{enf} \subseteq Args$. Let $LabArg_{pref}$ be a non-stable preferred labelling and let $Args \subseteq Ar$. By (the contrapositive of) Theorem 9 it holds that: if there exists a labelling $LabArg$ of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubset LabArg$, $\texttt{undec}(LabArg) = \emptyset$, and there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$, then $Args \nsubseteq Ar \setminus Args_{prev}$ where $Args_{prev}$ is a preventing set w.r.t. $LabArg_{pref}$. Thus, $\exists A \in Args$ such that $A \notin Ar \setminus Args_{prev}$, and consequently $A \in Args_{prev}$. Since this holds for all preventing sets $Args_{prev}$, let $Args'$ be the set of all such $A \in Args$ that are part of a preventing set, so $Args'$ consists of at least one argument from each preventing set. By Theorem 11, $Args_{enf} \subseteq Args'$, where $Args_{enf}$ is an enforcement set, and since $Args' \subseteq Args$, it follows that $Args_{enf} \subseteq Args$. □

Example 16. Consider $\mathcal{AF}_4$ and its only preferred labelling $LabArg_{pref}$, illustrated at the top of Figure 10. Let $LabArg$ be the labelling illustrated at the bottom of Figure 10 and let $Args = \{d, g\}$. Then $\mathcal{AF}_4^{\circledast}$ at the top of Figure 11 is a revision of $\mathcal{AF}_4$ w.r.t. $Args$ by $LabArg$, where the labelling $LabArg^{\circledast}$ at the top of Figure 11 is a revision labelling of $\mathcal{AF}_4^{\circledast}$. We note that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. As stated by Theorem 12, $Args$ is a superset of some enforcement set, in fact, it is a superset of both enforcement set $\{d\}$ and enforcement set $\{g\}$.

Note that even if a set of arguments used to revise an AF is a superset of an enforcement set, the labelling used for the revision may be different from all enforcement labellings of this enforcement set. For example, $LabArg$ from Example 16 (see bottom of Figure 10) is used for the revision of $\mathcal{AF}_4$ w.r.t. $Args$, but it is not an enforcement labelling of either of the two enforcement sets that are subsets of $Args$. For instance, the only enforcement labelling of the enforcement set $\{d\}$ is illustrated at the bottom of Figure 11.

The next Corollary follows directly from Theorem 12 and states that the converse of Theorem 6 holds.

Corollary 13. *Let $Args \subseteq Ar$ and let $LabArg$ be a labelling of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubseteq LabArg$, $\texttt{undec}(LabArg) = \emptyset$, and for all revisions $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and all revision labellings $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ it holds that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. Then there exists an enforcement set $Args_{enf}$ w.r.t. $LabArg_{pref}$ such that $Args_{enf} \subseteq Args$.*

Theorem 14 proves that the converse of Theorem 9 holds. That is, if no revision w.r.t. a set of arguments $Args$ is such that some revision labelling is a stable labelling of the revision, then there exists a preventing set that is disjoint from $Args$. In other words, preventing sets define a *necessary* condition for the non-existence of a stable labelling that is more committed than $LabArg_{pref}$.

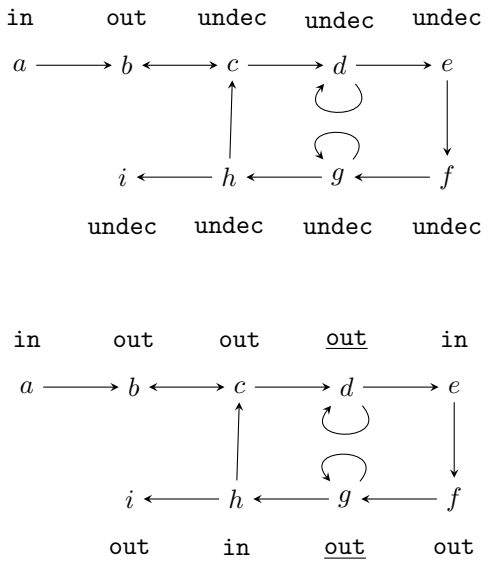

Figure 10: $\mathcal{AF}_4$ with its only preferred labelling $LabArg_{pref}$ (top) and with a labelling $LabArg$ that is more committed than $LabArg_{pref}$, where arguments d and g are illegally labelled (bottom).

Theorem 14. *Let $Args \subseteq Ar$ be such that, for all labellings $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, there exists no revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ such that some revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. Then there exists a preventing set $Args_{prev}$ w.r.t. $LabArg_{pref}$ such that $Args \subseteq Ar \setminus Args_{prev}$.*

Proof. Since $LabArg_{pref} \sqsubset LabArg$ it follows that $LabArg_{pref}$ is not a stable labelling. Let $Args \subseteq Ar$. By (the contrapositive of) Corollary 7 it holds that: if for all labellings $LabArg$ of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, there exists no revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ such that some revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$, then $Args \not\supseteq Args_{enf}$ where $Args_{enf}$ is an enforcement set. Thus, $\exists A \in Args_{enf}$ such that $A \notin Args$. Since this holds for all enforcement sets $Args_{enf}$, let $Args'$ be the set of all such A occurring in some enforcement set such that $A \notin Args$, so $Args'$ consists of at least one argument from each enforcement set. By Theorem 10, $Args' \supseteq Args_{prev}$ where $Args_{prev}$ is a preventing set. Clearly, $Args \subseteq Ar \setminus Args'$, so $Args \subseteq Ar \setminus Args_{prev}$ where $Args_{prev}$ is a preventing set. □

Example 17. Consider again $\mathcal{AF}_3 = \langle Ar_3, Att_3 \rangle$ illustrated on the left of Figure 9 (see page 15) and the set of arguments $Args = \{c, d\}$. Then, for any $\texttt{in-out}$ labelling $LabArg$ of $\mathcal{AF}_3$ that is more committed than $LabArg_{pref}$, there exists no revision $\mathcal{AF}_3^{\circledast}$ of $\mathcal{AF}_3$ w.r.t. $Args$ by $LabArg$ such that a revision labelling of $\mathcal{AF}_3^{\circledast}$ is a stable labelling of $\mathcal{AF}_3^{\circledast}$, since any revision labelling will illegally label

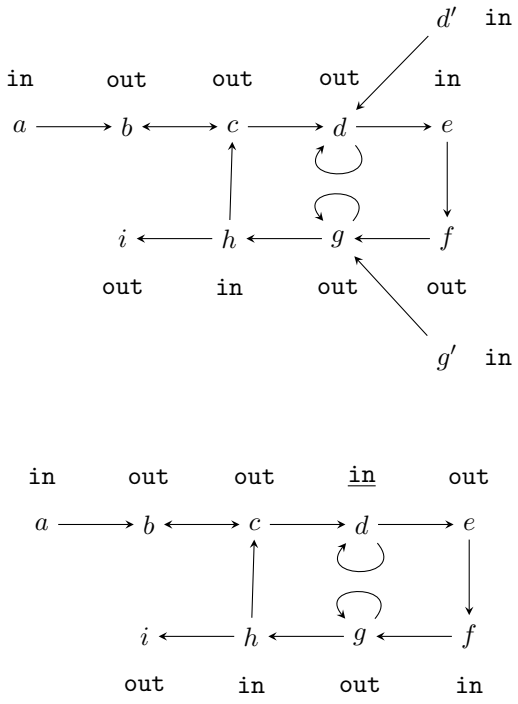

Figure 11: Top – A revision $\mathcal{AF}_4^{\circledast}$ of $\mathcal{AF}_4$ and a revision labelling $LabArg^{\circledast}$ (see Example 16). Bottom – The only enforcement labelling of the enforcement set $\{d\}$ of $\mathcal{AF}_4$ w.r.t. $LabArg_{pref}$.

e (as no attacks can be added to or deleted from e). As stated by Theorem 14, it holds that for the preventing set $\{e\}$, $Args \subset Ar_3 \setminus \{e\}$.

Note that the correspondence between preventing and enforcement sets implies that, for a revision w.r.t. a set of arguments comprising an argument from each preventing set, there exists a revision whose revision labelling is a stable labelling (of the revision). Thus, a revision w.r.t. the union of all preventing sets by an appropriately chosen labelling is a *sufficient* condition for obtaining a stable labelling (of the revision) that is more committed than $LabArg_{pref}$.

Theorems 6 and 12 as well as Theorems 9 and 14 show that enforcement and preventing sets indeed characterise *exactly* those sets of arguments *responsible* if a preferred labelling is not a stable labelling. Enforcement sets are responsible since they consist of exactly the arguments whose labels need to be enforced in order to obtain a stable labelling (of a revision), whereas preventing sets are responsible because they consist of exactly those arguments that prevent the existence of a stable labelling (of a revision) if the label of no argument in the set is enforced.

5. Structural Characterisations

Determining responsible sets of arguments according to the declarative labelling-based characterisations from Section 4 involves guessing sets of arguments and checking if they satisfy the respective definition. In this section, we instead characterise sets of arguments as responsible if a preferred labelling is not a stable labelling based on the *structure* of the AF. We thereby aim at characterisations that allow for a *constructive* determination of responsible sets of arguments.

5.1. Odd-Length Cycles

Our first structural characterisation is inspired by the seminal work of Dung [10], who proved that if an AF has no odd-length cycles, then a stable extension – and thus a stable labelling[5] – exists. Consequently, the non-existence of stable labellings implies the existence of an odd-length cycle.

We show that, furthermore, an odd-length cycle exists if some preferred labelling is not stable, even if the AF has a stable labelling. In particular, there exists an odd-length cycle of arguments labelled `undec` by this (non-stable) preferred labelling. Thus, we define such odd-length cycles of arguments labelled `undec` as responsible if the preferred labelling is not stable. The reason to exclude odd-length cycles of arguments labelled `in` or `out` is that such cycles do not violate the definition of a stable labelling.

Definition 6 (Structural Characterisation 1). $\mathscr{C} \subseteq Ar$ is a *responsible cycle* w.r.t. $LabArg_{pref}$ if and only if $\mathscr{C}$ is an odd-length cycle of $\mathcal{AF}$ and for all $A \in \mathscr{C}$ it holds that $A \in \texttt{undec}(LabArg_{pref})$.

Example 18. Let $\mathcal{AF}_5$ be the AF illustrated in Figure 12 and let $LabArg_{pref}$ be its only preferred labelling, also depicted in the figure. $\mathcal{AF}_5$ has five odd-length cycles, including nested ones: the odd-length cycle $\{d, e, f\}$ contains the two odd-length cycles $\{d\}$ and $\{e\}$. However, only the two odd-length cycles $\mathscr{C}_1 = \{c\}$ and $\mathscr{C}_2 = \{e\}$ are responsible cycles w.r.t. $LabArg_{pref}$.

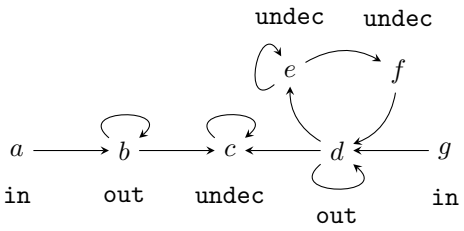

Figure 12: $\mathcal{AF}_5$ and its only preferred labelling (see Example 18).

[5] by the correspondence between extensions and labellings [11]

In contrast to our labelling-based characterisations, which always exist but coincide with the empty set in case $LabArg_{pref}$ is a stable labelling, responsible cycles exist if and only if $LabArg_{pref}$ is not stable. Thus, responsible cycles are well-defined characterisations of parts of an AF responsible if $LabArg_{pref}$ is not a stable labelling.

Proposition 15. *There exists a responsible cycle w.r.t. $LabArg_{pref}$ if and only if $LabArg_{pref}$ is not a stable labelling.*

We are again interested in characterising our responsible parts as indeed being responsible that $LabArg_{pref}$ is not stable. The following proposition states that it is *sufficient* to consider the set of all responsible cycles to obtain a stable labelling by enforcing labels for arguments in the responsible cycles. Importantly, to obtain a stable labelling (of a revision) that is more or equally committed than $LabArg_{pref}$, the labelling used for the revision has to be chosen carefully.

Proposition 16. *Let $S = \{A \in Ar \mid \mathscr{C}$ is a responsible cycle w.r.t. $LabArg_{pref}$, $A \in \mathscr{C}\}$. Then there exists a labelling $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$ such that, for all revisions $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. S by $LabArg$ and all revision labellings $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$, $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$.*

Note that this also holds if $LabArg_{pref}$ is a stable labelling, in which case $S = \emptyset$, $LabArg = LabArg^{\circledast} = LabArg_{pref}$, and $\mathcal{AF}^{\circledast} = \mathcal{AF}$.

Example 19. Consider again $\mathcal{AF}_4$ illustrated at the top of Figure 10 (see page 22). The set of arguments occurring in responsible cycles w.r.t. $LabArg_{pref}$ is $S = \{d, g\}$. Consider the labelling $LabArg$ depicted at the bottom of Figure 10, which is more committed than $LabArg_{pref}$ and labels no arguments as `undec`. A revision $\mathcal{AF}_4^{\circledast}$ of $\mathcal{AF}_4$ w.r.t. S by $LabArg$ is shown at the top of Figure 11, along with a revision labelling that is a stable labelling of $\mathcal{AF}_4^{\circledast}$.

Since by Lemma 1 a revision exists w.r.t. any set of arguments and any labelling, it follows that there indeed exists a revision w.r.t. responsible cycles that has a stable labelling that is more committed than the given non-stable preferred labelling.

Corollary 17. *Let $S = \{A \in Ar \mid \mathscr{C}$ is a responsible cycle w.r.t. $LabArg_{pref}$, $A \in \mathscr{C}\}$. Then there exists a labelling $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, and there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. S by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$.*

5.2. Strongly Connected Components

Our second structural characterisation is based upon a result on the composition of stable labellings, namely that stable labellings can be computed

along the SCCs [16] of the AF. That is, the stable labellings of initial SCCs are computed and, subsequently, the stable labellings of the following SCCs are iteratively determined, while taking the labels of arguments in their parent SCCs into account. It follows that if the AF has no stable labelling, some SCC in this iterative computation has no stable labelling (when taking the labels in parent SCCs into account).

The following structural characterisation of sets of arguments responsible if $LabArg_{pref}$ is not a stable labelling refines this observation. It defines as responsible the "first" SCCs that have no stable labelling in the iterative computation of a stable labelling, given the labels of $LabArg_{pref}$. More precisely, responsible sets are SCCs satisfying that: 1) the SCC has no stable labelling w.r.t. the input from its parent SCCs, i.e. w.r.t. the labels of attackers in parent SCCs according to $LabArg_{pref}$; and 2) all parent SCCs have a stable labelling w.r.t. the input from their parent SCCs that coincides with the labels assigned by $LabArg_{pref}$.

Definition 7 (Structural Characterisation 2). $Args \subseteq Ar$ is a *responsible SCC* w.r.t. $LabArg_{pref}$ if and only if $Args$ is an SCC of $\mathcal{AF}$ such that

1. there exists no stable labelling w.r.t. $(\mathcal{AF}{\downarrow_{Args}}, parents(Args), LabArg_{pref}{\downarrow_{parents(Args)}}, Att \cap (parents(Args) \times Args))$ that is more or equally committed than $LabArg_{pref}{\downarrow_{Args}}$, and
2. for all $Args' \in parentSCCs(Args)$, $LabArg_{pref}{\downarrow_{Args'}}$ is a stable labelling w.r.t. $(\mathcal{AF}{\downarrow_{Args'}}, parents(Args'), LabArg_{pref}{\downarrow_{parents(Args')}}, Att \cap (parents(Args') \times Args'))$.

Example 20. The only responsible SCC of $\mathcal{AF}_{therapy}$ w.r.t. its only preferred labelling $LabArg_{pref}$ (see Figure 1 on page 3) is $\{A, B, C\}$. Since this is an initial SCC, it is trivially satisfied that its parent SCCs have a stable labelling.

The following example illustrates an AF where a responsible SCC is not an initial SCC of the AF.

Example 21. Consider again $\mathcal{AF}_4$ and its only preferred labelling $LabArg_{pref}$, illustrated at the top of Figure 10 (see page 22). The only responsible SCC w.r.t. $LabArg_{pref}$ is the SCC $\{b, c, d, e, f, g, h\}$ since: 1) there exists no stable labelling w.r.t. the AF with input $(\mathcal{AF}{\downarrow_{\{b,c,d,e,f,g,h\}}}, \{a\}, \{(a, \texttt{in})\}, \{(a, b)\})$, which is depicted in Figure 13; and 2) $\{b, c, d, e, f, g, h\}$ only has one parent SCC, namely $\{a\}$, and $LabArg_{pref}$ restricted to $\{a\}$, i.e $\{(a, \texttt{in})\}$, is a stable labelling w.r.t. $(\mathcal{AF}{\downarrow_{\{a\}}}, \emptyset, \emptyset, \emptyset)$.

Note that Definition 7 does not require a responsible SCC to not have a stable labelling at all (w.r.t. its parent SCCs), but rather that it has no stable labelling that is more committed than the labels assigned to the SCC by $LabArg_{pref}$.

Example 22. Let $\mathcal{AF}_6$ be the AF in Figure 14, which has no stable labelling, and consider the depicted preferred labelling $LabArg_{pref}$. $\mathcal{AF}_6$ has three SCCs, namely $\{a, b, c\}$, $\{d\}$, and $\{e\}$. The SCC $\{a, b, c\}$ *has* a stable labelling w.r.t.

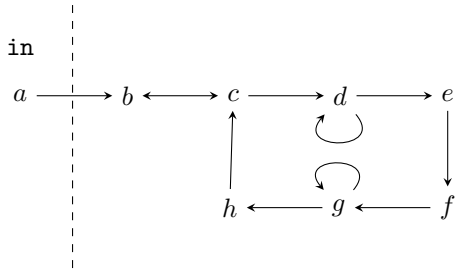

Figure 13: The AF with input made of the SCC $\{b, c, d, e, f, g, h\}$ of $\mathcal{AF}_4$ (right of dashed line) and the input arguments from its parent SCCs (left of dashed line) with the input labelling (given by the preferred labelling of $\mathcal{AF}_4$).

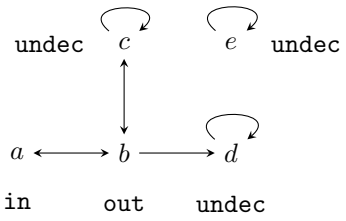

Figure 14: $\mathcal{AF}_6$ and a preferred labelling $LabArg_{pref}$.

$(\mathcal{AF}{\downarrow_{\{a,b,c\}}}, \emptyset, \emptyset, \emptyset)$, namely $\{(a, \texttt{out}), (b, \texttt{in}), (c, \texttt{out})\}$, but this stable labelling is not more or equally committed than $LabArg_{pref}{\downarrow_{\{a,b,c\}}}$. This illustrates the importance of the comparison with $LabArg_{pref}$ in the first condition of Definition 7: due to the comparison, $\{a, b, c\}$ satisfies the condition; without the comparison, $\{a, b, c\}$ would not satisfy the condition. Thus, without the comparison $\{a, b, c\}$ would not be identified as a responsible SCC. However, $\{a, b, c\}$ should be identified as responsible since it is the "first" SCC that provides a reason why $LabArg_{pref}$ is not a stable labelling.

Of similar importance is the comparison with $LabArg_{pref}$ in the second condition of Definition 7. Consider the SCC $\{d\}$ and its parent SCC $\{a, b, c\}$. $\{a, b, c\}$ *has* a stable labelling w.r.t. $(\mathcal{AF}{\downarrow_{\{a,b,c\}}}, \emptyset, \emptyset, \emptyset)$, namely $\{(a, \texttt{out}), (b, \texttt{in}), (c, \texttt{out})\}$, so without the comparison with $LabArg_{pref}$, $\{d\}$ would be identified as a responsible SCC. However, since the stable labelling w.r.t. $(\mathcal{AF}{\downarrow_{\{a,b,c\}}}, \emptyset, \emptyset, \emptyset)$ does not coincide with $LabArg_{pref}{\downarrow_{\{a,b,c\}}}$, $\{d\}$ is not a responsible SCC.

We prove that at least one responsible SCC exists if and only if the given preferred labelling is not stable.

Proposition 18. *There exists a responsible SCC w.r.t. $LabArg_{pref}$ if and only if $LabArg_{pref}$ is not a stable labelling.*

Differently from our previous characterisations, we do not investigate the role of responsible SCCs w.r.t. the existence of a stable labelling (of a revision), since

we do so for our next structural characterisation, which refines our definition of responsible SCCs.

5.3. Strongly Connected `undec` *Parts (SCUPs)*

Our characterisation of responsible SCCs relies on the decomposability of stable labellings with regards to the SCCs of an AF. In this section, we refine this notion by using another decomposability result. Baroni et al. [18] show that the complete labellings of an AF can be obtained by splitting the AF into *any* partition and then determining complete labellings of the different parts in such a way that they are compatible. We can thus think of $LabArg_{pref}$ as a combination of two compatible labellings: a labelling of the part of the AF whose arguments are labelled `in` or `out` by $LabArg_{pref}$, and a labelling of the part of the AF whose arguments are labelled `undec` by $LabArg_{pref}$. We call these two parts the `in`/`out`-part and the `undec`-part, respectively.

The fact that all arguments in the `undec`-part are labelled `undec` by $LabArg_{pref}$ implies that this is the only labelling compatible with the `in` and `out` labels in the `in`/`out`-part (if there was another labelling, $LabArg_{pref}$ would not be maximal). Proposition 19 proves that, furthermore, labelling all arguments in the `undec`-part as `undec` is the *only complete labelling* of this part on its own (disregarding the `in`/`out`-part). In other words, the labels of arguments in the `in`/`out`-part are not responsible that all arguments in the `undec`-part are labelled `undec`. Rather, the structure of the `undec`-part itself is responsible that the arguments cannot be legally labelled `in` or `out`.

Proposition 19. *Let* $\texttt{undec}(LabArg_{pref}) \neq \emptyset$*. The only complete labelling of* $\mathcal{AF}{\downarrow_{\texttt{undec}(LabArg_{pref})}}$ *labels all arguments as* `undec`*.*

Since the `undec`-part has only one complete labelling, which labels all arguments as `undec`, this labelling is also its only preferred labelling. Thus, the question as to why $LabArg_{pref}$ is not a stable labelling can be reduced to the question as to why the only preferred labelling of the `undec`-part is not a stable labelling.

Applying our notion of responsible SCCs, we obtain that the preferred labelling of the `undec`-part is not a stable labelling because of its "first" SCCs that have no stable labelling. These "first" SCCs are the initial SCCs of the `undec`-part since *no* SCC in the `undec`-part has a stable labelling. This observation results in the following new characterisation of sets of arguments responsible if $LabArg_{pref}$ is not a stable labelling: a set of arguments is responsible if it is an initial SCC of the `undec`-part.

Definition 8 (Structural Characterisation 3). $Args \subseteq Ar$ is a *strongly connected* `undec` *part (SCUP)* w.r.t. $LabArg_{pref}$ if and only if $Args$ is an initial SCC of $\mathcal{AF}{\downarrow_{\texttt{undec}(LabArg_{pref})}}$.

Example 23. $\mathcal{AF}_{therapy}$ from Section 1 has only one SCUP w.r.t. its only preferred labelling (see Figure 1 on page 3), namely $\{A, B, C\}$.

Importantly, at least one SCUP exists w.r.t. a preferred labelling if and only if this preferred labelling is not stable, which shows that SCUPs provide a well-defined characterisation of responsible sets of arguments.

Proposition 20. *There exists a SCUP w.r.t. $LabArg_{pref}$ if and only if $LabArg_{pref}$ is not stable.*

The following example illustrates that an AF may have various SCUPs w.r.t. a preferred labelling.

Example 24. Let $\mathcal{AF}_7$ and its only preferred labelling $LabArg_{pref}$ be as illustrated in Figure 15. There are two SCUPs w.r.t. $LabArg_{pref}$, namely $\{c\}$ and $\{d\}$.

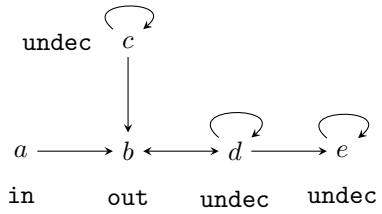

Figure 15: $\mathcal{AF}_7$ and its only preferred labelling $LabArg_{pref}$ (see Example 24).

We now prove that SCUPs are refinements of responsible SCCs in the sense that every responsible SCC comprises a SCUP.

Proposition 21. *Let $Args$ be a responsible SCC w.r.t. $LabArg_{pref}$. Then $\exists Args' \subseteq Args$ such that $Args'$ is a SCUP w.r.t. $LabArg_{pref}$.*

Example 25. Consider again $\mathcal{AF}_4$ and its only preferred labelling $LabArg_{pref}$ illustrated at the top of Figure 10 (see page 22). As discussed in Example 21, the only responsible SCC w.r.t. $LabArg_{pref}$ is $\{b, c, d, e, f, g, h\}$. As expected, there exists a SCUP that is a subset of this responsible SCC, namely $\{c, d, e, f, g, h\}$, which is the only SCUP of $\mathcal{AF}_4$ w.r.t. $LabArg_{pref}$.

Note that the converse of Proposition 21 does not hold in general, i.e. it is not the case that every SCUP is a subset of some responsible SCC. For example, $\{d\}$ is a SCUP of $\mathcal{AF}_7$ w.r.t. $LabArg_{pref}$ (see Figure 15), but the SCC containing d, i.e. $\{b, d\}$, is not a responsible SCC, since the parent SCC $\{c\}$ has no stable labelling.

Even though SCUPs are defined based on the *structure* of the AF rather than based on labellings that are more committed than $LabArg_{pref}$, as our labelling-based characterisations, we prove that SCUPs constitute sets of arguments that cannot all be legally labelled `in` or `out`. More precisely, with respect to any `in-out` labelling that is more committed than $LabArg_{pref}$, at least one argument in every SCUP is illegally labelled. This indicates that SCUPs indeed characterise sets of arguments responsible if $LabArg_{pref}$ is not a stable labelling.

Lemma 22. *Let $Args$ be a SCUP w.r.t. $LabArg_{pref}$. Then, for all labellings $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, it holds that there exists $A \in Args$ such that A is illegally labelled by $LabArg$.*

Since by Proposition 21 every responsible SCC comprises a SCUP, an analogous result to Lemma 22 also holds for responsible SCCs. That is, with respect to all `in-out` labellings that are more committed than $LabArg_{pref}$, at least one argument in every responsible SCC is illegally labelled.

5.4. Revising SCUPs

In this section, we investigate the responsibility of SCUPs in more detail, by examining revisions that turn $LabArg_{pref}$ into a stable labelling (of the revision). We first prove that, as for preventing sets, SCUPs provide a *sufficient* condition for "preventing" the existence of a stable labelling that is more committed than $LabArg_{pref}$. That is, a revision w.r.t. a set of arguments not containing an argument from some SCUP will not yield a stable labelling that is more committed than $LabArg_{pref}$.

Theorem 23. *Let $Args \subseteq Ar \setminus Args_{SCUP}$ where $Args_{SCUP}$ is a SCUP w.r.t. $LabArg_{pref}$ and let $LabArg$ be a labelling of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$. Then there exists no revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ such that some revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$.*

Proof. Assume there exists a revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and a revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. By Lemma 22, $\exists A \in Args_{SCUP}$ such that A is illegally labelled by $LabArg$ in $\mathcal{AF}$. Since $A \in Ar \setminus Args$, by Lemma 37 in Appendix A, A is illegally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Contradiction. □

Example 26. Consider again the SCUPs of $\mathcal{AF}_7$ w.r.t. its only preferred labelling $LabArg_{pref}$ (see Example 24 on page 29). Let $LabArg$ be the `in-out` labelling illustrated in Figure 16, which is more committed than $LabArg_{pref}$. The set $\{a, b, d, e\}$ does not contain any argument from the SCUP $\{c\}$. It is easy to see that there exists no revision $\mathcal{AF}_7^{\circledast}$ of $\mathcal{AF}_7$ w.r.t. $\{a, b, d, e\}$ by $LabArg$ such that a revision labelling is a stable labelling of $\mathcal{AF}_7^{\circledast}$, since c will always be illegally labelled `out`.

Therefore, if we are to obtain a stable labelling, a revision has to involve arguments from every SCUP. In what follows, we thus investigate whether enforcing labels for arguments in all SCUPs yields a stable labelling (of a revision). For this purpose, we define a *SCUP revision* as a revision w.r.t. the set of all arguments in all SCUPs by a labelling that is more committed than $LabArg_{pref}$ and labels all arguments in all SCUPs as `in` or `out`. That is, a SCUP revision enforces the labels `in` and `out` for all arguments in all SCUPs.

Notation 2. Let $Args_1, \ldots, Args_n$ be all SCUPs w.r.t. $LabArg_{pref}$. $\mathcal{SCUPS} = Args_1 \cup \ldots \cup Args_n$ denotes the set of all arguments in SCUPs.

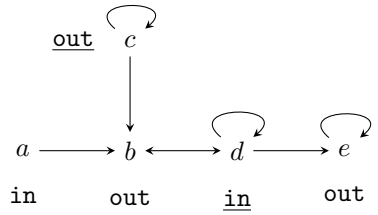

Figure 16: $\mathcal{AF}_7$ and a labelling $LabArg$ (see Example 26), where illegal labels are underlined.

Definition 9 (SCUP Revision and SCUP Revision Labelling). Let $LabArg_{\mathcal{SCUPS}}$ be a labelling of $\mathcal{AF}{\downarrow_{\mathcal{SCUPS}}}$ with $\texttt{undec}(LabArg_{\mathcal{SCUPS}}) = \emptyset$ and let $LabArg = LabArg_{\mathcal{SCUPS}} \cup LabArg_{pref}{\downarrow_{Ar \setminus \mathcal{SCUPS}}}$. $\mathcal{AF}^{\circledast}$ is a *SCUP revision* of $\mathcal{AF}$ if and only if $\mathcal{AF}^{\circledast}$ is a revision of $\mathcal{AF}$ w.r.t. $\mathcal{SCUPS}$ by $LabArg$. A revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ is called a *SCUP revision labelling* of $\mathcal{AF}^{\circledast}$.

Example 27. Consider again $\mathcal{AF}_7$ from Example 24 (see Figure 15 on page 29). A SCUP revision of $\mathcal{AF}_7$ along with a SCUP revision labelling is depicted on the left of Figure 17. The labelling of arguments in $\mathcal{SCUPS}$ used for the SCUP revision is $LabArg_{\mathcal{SCUPS}} = \{(c, \texttt{out}), (d, \texttt{in})\}$.

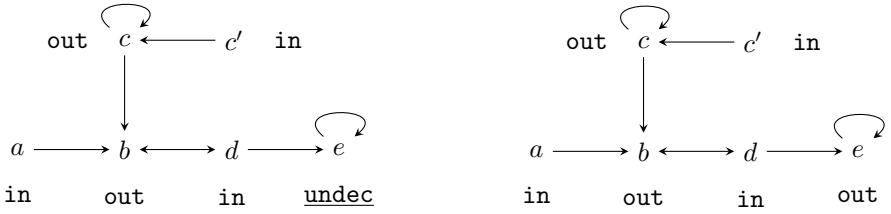

Figure 17: Left – $\mathcal{AF}_7^{\circledast}$ and a SCUP revision labelling $LabArg^{\circledast}$ (see Example 27), where illegal labels are underlined. Right – $\mathcal{AF}_7^{\circledast}$ and a preferred labelling that is more committed than $LabArg^{\circledast}$ (see Example 28).

Since, by Lemma 1, a revision exists w.r.t. any set of arguments and labelling and since, by Proposition 20, there exists a SCUP w.r.t. a preferred labelling if and only if it is not stable, the following holds.

Corollary 24. *There exists a SCUP revision $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ if and only if $LabArg_{pref}$ is not a stable labelling.*

The SCUP revision from Example 27 illustrates that a SCUP revision labelling may not be a complete labelling of the SCUP revision since arguments labelled `undec` by $LabArg_{pref}$ that are not contained in $\mathcal{SCUPS}$ may be illegally labelled (see the left of Figure 17). However, we prove that there exists a *preferred labelling* of the SCUP revision that is more or equally committed than

the SCUP revision labelling. In other words, illegally labelled `undec` arguments can be appropriately changed to `in` or `out` labels, yielding a preferred labelling of the SCUP revision.

Theorem 25. *Let $\mathcal{AF}^{\circledast}$ be a SCUP revision of $\mathcal{AF}$ and $LabArg^{\circledast}$ a SCUP revision labelling of $\mathcal{AF}^{\circledast}$. Then there exists a preferred labelling $LabArg^{\circledast}_{pref}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg^{\circledast} \sqsubseteq LabArg^{\circledast}_{pref}$.*

Proof. Let $\mathcal{SCUPS}^{\circledast} = \{A \in Ar^{\circledast} \mid A \in \mathcal{SCUPS} \vee A \notin Ar\}$. Let $Args_1 = \mathtt{in}(LabArg_{pref}) \cup \mathtt{out}(LabArg_{pref}) \cup \mathcal{SCUPS}^{\circledast}$, $Args_2 = Ar^{\circledast} \setminus Args_1$, and $LabArg_1 = LabArg^{\circledast}{\downarrow}_{Args_1}$.
By Definitions 9 and 1 it holds that $\forall A \in \mathcal{SCUPS}^{\circledast}$, A is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Since $\mathcal{SCUPS}$ consists of arguments in SCUPs, it holds that $\forall A \in \mathcal{SCUPS}^{\circledast}$ and $\forall B$ attacking A in $\mathcal{AF}^{\circledast}$, $B \in Args_1$. Thus, A being legally labelled by $LabArg^{\circledast}$ only depends on $LabArg_1$. Let $LabArg_2$ be some labelling of $Args_2$. Then, $\forall A \in \mathcal{SCUPS}^{\circledast}$, A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}^{\circledast}$. Note that for any $LabArg_2$ of $Args_2$ it holds that $LabArg^{\circledast}{\downarrow}_{Args_2} \sqsubseteq LabArg_2$ since $\mathtt{undec}(LabArg^{\circledast}{\downarrow}_{Args_2}) = Args_2$, because $Args_2 \subseteq \mathtt{undec}(LabArg_{pref}) \setminus \mathcal{SCUPS}$. Then $LabArg^{\circledast} \sqsubseteq LabArg_1 \cup LabArg_2$.
Let $LabArg = LabArg_{\mathcal{SCUPS}} \cup LabArg_{pref}{\downarrow}_{Ar \setminus \mathcal{SCUPS}}$ be the labelling used for the SCUP revision. By Lemma 38 in Appendix A, $\forall A \in \mathtt{in}(LabArg_{pref}) \cup \mathtt{out}(LabArg_{pref})$ it holds that A is legally labelled by $LabArg$ in $\mathcal{AF}$ since $LabArg_{pref} \sqsubset LabArg$. Then, by Lemma 37 in Appendix A, $\forall A \in \mathtt{in}(LabArg_{pref}) \cup \mathtt{out}(LabArg_{pref})$ it holds that A is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Since $LabArg^{\circledast} \sqsubseteq LabArg_1 \cup LabArg_2$, by Lemma 38 in Appendix A it holds that $\forall A \in \mathtt{in}(LabArg_{pref}) \cup \mathtt{out}(LabArg_{pref})$, A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}^{\circledast}$.
Thus, $\forall A \in Args_1$, A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}^{\circledast}$. Then, by Lemma 43 in Appendix A, $LabArg_1$ is compatible with $LabArg_2$ (for any labelling $LabArg_2$ of $Args_2$). Furthermore by Lemma 45 in Appendix A, there exists a labelling $LabArg_2'$ that is compatible with $LabArg_1$. Then, by Lemma 40 in Appendix A, $LabArg_1 \cup LabArg_2'$ is a complete labelling of $\mathcal{AF}^{\circledast}$. Then either $LabArg_1 \cup LabArg_2'$ is a preferred labelling of $\mathcal{AF}^{\circledast}$ or there exists a preferred labelling $LabArg^{\circledast\prime}$ such that $LabArg_1 \cup LabArg_2' \sqsubset LabArg^{\circledast\prime}$ and thus $LabArg^{\circledast} \sqsubset LabArg^{\circledast\prime}$. □

Example 28. Given the SCUP revision $\mathcal{AF}_7^{\circledast}$ and the SCUP revision labelling $LabArg^{\circledast}$ from Example 27 (see left of Figure 17), there exists a preferred labelling of $\mathcal{AF}_7^{\circledast}$ that is more committed than $LabArg^{\circledast}$, as illustrated on the right of Figure 17.

Since a SCUP revision labelling is more committed than $LabArg_{pref}$ (because all arguments in SCUPs are labelled `in` or `out` by the SCUP revision labelling, but are labelled `undec` by $LabArg_{pref}$), it follows that there exists a preferred labelling of the SCUP revision that is more committed than $LabArg_{pref}$.

Corollary 26. *Let $\mathcal{AF}^{\circledast}$ be a SCUP revision of $\mathcal{AF}$. Then there exists a preferred labelling $LabArg^{\circledast}_{pref}$ of $\mathcal{AF}^{\circledast}$ such that $LabArg_{pref} \sqsubset LabArg^{\circledast}_{pref}$.*

In Example 28, there exists a preferred labelling of the SCUP revision that is more committed than the SCUP revision labelling and that is also a *stable* labelling of the SCUP revision. However, in general a SCUP revision may not have a stable labelling that is more committed than the SCUP revision labelling.

Example 29. Let $\mathcal{AF}_8$ and its only preferred labelling $LabArg_{pref}$ be as illustrated at the top of Figure 18. There are two SCUPs w.r.t. $LabArg_{pref}$, namely $\{a\}$ and $\{e\}$. A SCUP revision $\mathcal{AF}_8^{\circledast}$ of $\mathcal{AF}_8$ is depicted at the bottom of Figure 18, along with the SCUP revision labelling. A preferred labelling $LabArg^{\circledast}_{pref}$ of $\mathcal{AF}_8^{\circledast}$ that is more committed than the revision labelling is illustrated in Figure 19. However, $LabArg^{\circledast}_{pref}$ is not a stable labelling of $\mathcal{AF}_8^{\circledast}$. Furthermore, in this example there exist no SCUP revision and SCUP revision labelling that result in a stable labelling that is more committed than $LabArg_{pref}$.

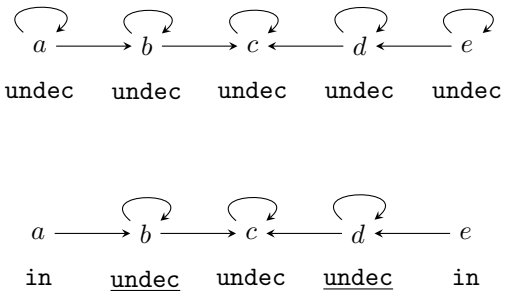

Figure 18: Top – $\mathcal{AF}_8$ and its only preferred labelling $LabArg_{pref}$. Bottom – A SCUP revision $\mathcal{AF}_8^{\circledast}$ of $\mathcal{AF}_8$ (see Example 29) and the SCUP revision labelling, where illegal labels are underlined.

To summarise, differently from preventing sets, revisions w.r.t. the union of all SCUPs are *not* guaranteed to have a stable labelling that is more committed than $LabArg_{pref}$. Nevertheless, they yield a more committed *preferred* labelling.

If a SCUP revision has a preferred labelling that is not a stable labelling, then by Proposition 20 there exists a SCUP w.r.t. this preferred labelling. In order to obtain a stable labelling of the whole AF, these "new" SCUPs thus have to be revised. We therefore define an iterative procedure of SCUP revisions w.r.t. preferred labellings, which can be used to obtain a stable labelling (of a revision). As for preventing sets, this shows the responsibility of SCUPs, since it provides a sufficient condition for obtaining a stable labelling.

Definition 10 (Iterative SCUP Revision). A sequence $\langle \mathcal{AF}^1, LabArg^1 \rangle, \ldots, \langle \mathcal{AF}^n, LabArg^n \rangle$ $(n > 1)$ is an *iterative SCUP revision* of $\mathcal{AF}$ if and only if

- $\mathcal{AF}^1 = \mathcal{AF}$ and $LabArg^1 = LabArg_{pref}$, and

- $\forall i$ $(1 \leq i < n)$ it holds that $\mathcal{AF}^{i+1}$ is a SCUP revision of $\mathcal{AF}^i$, with $LabArg^{\circledast^{i+1}}$ a SCUP revision labelling of $\mathcal{AF}^{i+1}$, and $LabArg^{i+1}$ is a preferred labelling of $\mathcal{AF}^{i+1}$ such that $LabArg^{\circledast^{i+1}} \sqsubseteq LabArg^{i+1}$.

We are, of course, most interested in iterative SCUP revisions that result in a stable labelling.

Definition 11 (Stable Iterative SCUP Revision). An iterative SCUP revision $\langle \mathcal{AF}^1, LabArg^1 \rangle, \ldots, \langle \mathcal{AF}^n, LabArg^n \rangle$ of $\mathcal{AF}$ is a *stable iterative SCUP revision* of $\mathcal{AF}$ if and only if $LabArg^n$ is a stable labelling of $\mathcal{AF}^n$.

Example 30. Consider again $\mathcal{AF}_8$ and its preferred labelling, illustrated at the top of Figure 18. An example of a stable iterative SCUP revision of $\mathcal{AF}_8$ is $\langle \mathcal{AF}_8^1, LabArg^1 \rangle, \langle \mathcal{AF}_8^2, LabArg^2 \rangle, \langle \mathcal{AF}_8^3, LabArg^3 \rangle$, where $\mathcal{AF}_8^2$ and $LabArg^2$ are depicted in Figure 12, and $\mathcal{AF}_8^3$ and $LabArg^3$ are as illustrated in Figure 19.

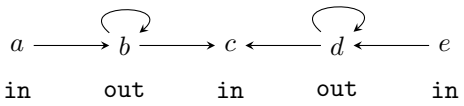

Figure 19: The AF obtained from a stable iterative SCUP revision of $\mathcal{AF}_8$ (see Example 30).

Since a SCUP revision has a preferred labelling that is more committed than $LabArg_{pref}$, each iteration in the iterative SCUP revision reduces the set of arguments labelled `undec`. Since there are only finitely many arguments, there exists an iterative SCUP revision that results in a stable labelling (of the revision).

Theorem 27. *There exists a stable iterative SCUP revision of $\mathcal{AF}$ if and only if $LabArg_{pref}$ is not a stable labelling.*

PROOF. If $LabArg_{pref}$ is a stable labelling then there exists no iterative SCUP revision since by Proposition 20, there exists no SCUP w.r.t. $LabArg_{pref}$. If $LabArg_{pref}$ is not a stable labelling, by Proposition 20, there exists a SCUP w.r.t. $LabArg_{pref}$ and thus by Corollary 26 there exists a preferred labelling $LabArg^2$ of $\mathcal{AF}^2$ such that $LabArg_{pref} \sqsubset LabArg^2$. If $LabArg^2$ is not a stable labelling, then, by Proposition 20, there exists a SCUP w.r.t. $LabArg^2$ and thus a SCUP revision $\mathcal{AF}^3$ of $\mathcal{AF}^2$, and by Corollary 26 a preferred labelling $LabArg^3$ of $\mathcal{AF}^3$ such that $LabArg^2 \sqsubset LabArg^3$. The same then applies to $\mathcal{AF}^3$, and so on. Thus, the set of `undec` arguments in $LabArg^i$ monotonically decreases, and since there are only finitely many arguments, the sequence terminates with some $\mathcal{AF}^n$ such that $\texttt{undec}(LabArg^n) = \emptyset$. □

Our results show that iteratively revising SCUPs provides a *sufficient* condition for turning a non-stable preferred labelling into a a stable labelling (of a revision), and SCUPs are thus parts of the AF that can be deemed *responsible*

that $LabArg_{pref}$ is not a stable labelling. Furthermore, SCUPs can be used for a well-directed revision of the AF that leads to a stable labelling that is more committed than $LabArg_{pref}$.

Since by Proposition 21 every responsible SCC comprises a SCUP, responsible SCCs also define a sufficient condition for for turning $LabArg_{pref}$ into a a stable labelling (of a revision), and consequently need to be revised in order to obtain a stable labelling. However, the condition provided by responsible SCCs is less refined than the notion of SCUPs. Therefore, we do not investigate the revision w.r.t. responsible SCCs in more detail.

5.5. Responsible Cycles versus Responsible SCCs and SCUPs

The characterisation of responsible arguments in terms of responsible cycles differs considerably from our second and third structural characterisations, which are based on SCCs. Nevertheless, we prove that the three characterisations are related. In particular, every SCUP comprises a responsible cycle.

Proposition 28. *Let $Args$ be a SCUP w.r.t. $LabArg_{pref}$. Then there exists a responsible cycle $\mathscr{C}$ w.r.t. $LabArg_{pref}$ such that $\mathscr{C} \subseteq Args$.*

Example 31. The only SCUP of $\mathcal{AF}_4$ (see top of Figure 10 on page 22) is $\{c, d, e, f, g, h\}$. There are furthermore two responsible cycles that form subsets of the SCUP, namely $\{d\}$ and $\{g\}$ (see Example 19 on page 25).

Note that the converse of Proposition 28 does not hold in general, i.e. it is not the case that every responsible cycle is a subset of some SCUP. For instance, in $\mathcal{AF}_8$ (see top of Figure 18 on page 33) each of the five self-attacking arguments is a responsible cycle. However, there are only two SCUPs, namely $\{a\}$ and $\{e\}$, so for instance the responsible cycle $\{b\}$ is not a subset of any SCUP.

Since by Proposition 21 every responsible SCC comprises a SCUP, it follows that every responsible SCC contains a responsible cycle.

Corollary 29. *Let $Args$ be a responsible SCC w.r.t. $LabArg_{pref}$. Then there exists a responsible cycle $\mathscr{C}$ w.r.t. $LabArg_{pref}$ such that $\mathscr{C} \subseteq Args$.*

Note that Propositions 16 and 28 imply that rather than defining a SCUP revision w.r.t. all arguments in SCUPs, we could only revise the responsible cycles in the SCUPs. This is illustrated by Example 19 (see page 25), where a revision w.r.t. the responsible cycles contained in the only SCUP is illustrated.

On the other hand, the responsible cycles in a SCUP do not *have to* be revised in order to legally label all arguments in the SCUP. Instead, the SCUP may be revised w.r.t. a subset of the SCUP not containing arguments from responsible cycles. For instance, a SCUP revision of $\mathcal{AF}_4$ w.r.t. $LabArg_{pref}$ (see top of Figure 10 on page 22) where no responsible cycles are revised is illustrated in Figure 20 (see page 36), along with a preferred labelling that is more committed than the SCUP revision labelling.

It is therefore up to the user to decide what type of SCUP revision is most suitable for the application at hand.

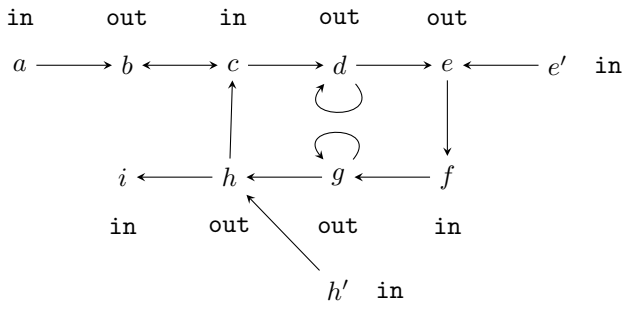

Figure 20: A SCUP revision of $\mathcal{AF}_4$ and a preferred labelling that is more committed than the SCUP revision labelling.

6. Labelling-Based versus Structural Characterisations

We presented two different approaches to characterising sets of arguments responsible if $LabArg_{pref}$ is not a stable labelling: a labelling-based and a structural approach. We proved that the labelling-based characterisations in terms of enforcement and preventing sets define necessary and sufficient conditions for the (non-) existence of a stable labelling that is more committed than $LabArg_{pref}$. However, these characterisations are not constructive, whereas our structural characterisations are. The structural characterisation in terms of SCUPs can also be used to guide the revision of an AF in such a way that a stable labelling is obtained, defining a sufficient but not a necessary condition for obtaining a stable labelling.

In this section, we examine the connection between our labelling-based and structural characterisations in more detail. Note that we omit the naïve characterisation of labelling-based responsible sets, since both enforcement and preventing sets are refinements of this characterisation. Similarly, we do not include responsible SCCs in our comparison since SCUPs provide a refinement of responsible SCCs.

6.1. SCUPs versus Preventing Sets

SCUPs and preventing sets share the property that if none of their arguments is involved in a revision, then the revision has no stable labelling that is more committed than $LabArg_{pref}$ (see Theorems 9 and 23). These results hint at a close connection between SCUPs and preventing sets. Indeed, Theorem 30 proves that each SCUP comprises a preventing set.

Theorem 30. *Let $Args_{SCUP}$ be a SCUP w.r.t. $LabArg_{pref}$. Then there exists a preventing set $Args_{prev}$ w.r.t. $LabArg_{pref}$ such that $Args_{prev} \subseteq Args_{SCUP}$.*

Proof. By Lemma 22, for all labellings $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, it holds that there exists $A \in Args_{SCUP}$ such that A is illegally labelled by $LabArg$. Then either $Args_{SCUP}$ is a minimal set

satisfying this property, and thus $Args_{SCUP}$ is a preventing set, or there exists a minimal set $Args_{prev} \subset Args_{SCUP}$ satisfying this property, so $Args_{prev}$ is a preventing set. □

Example 32. Consider again $\mathcal{AF}_4$ illustrated on the top of Figure 10 (see page 22). As discussed in Example 25 (see page 29), the only SCUP w.r.t. $LabArg_{pref}$ is $\{c, d, e, f, g, h\}$. Here, two different preventing sets w.r.t. $LabArg_{pref}$ are subsets of the SCUP, namely $\{c, d, g, h\}$ and $\{d, e, f, g\}$.

Note that, conversely, it is not the case that every preventing set is a subset of some SCUP.

Example 33. Consider again $\mathcal{AF}_8$, illustrated at the top of Figure 18 (see page 33). There are three preventing sets w.r.t. $LabArg_{pref}$: $\{a\}$, $\{e\}$, and $\{b, c, d\}$. The first two coincide with the two SCUPs w.r.t. $LabArg_{pref}$, but the latter is not a subset of any SCUP.

Since SCUPs only characterise the "first" problematic sets of arguments, whereas preventing sets define "all" problematic sets, it is not surprising that some preventing sets are disjoint from SCUPs. However, when considering all SCUPs in a stable iterative SCUP revision, every preventing set shares an argument with some SCUP.

Notation 3. Let $\langle \mathcal{AF}^1, LabArg^1 \rangle, \ldots, \langle \mathcal{AF}^n, LabArg^n \rangle$ be an iterative SCUP revision. $\biguplus \mathcal{SCUPS} = \{\mathcal{SCUPS}^i \mid \mathcal{SCUPS}^i$ *is the set of all arguments in SCUPs w.r.t.* $LabArg^i, 1 \leq i \leq n\}$ consists of the sets of arguments in SCUPs at every step in the iterative SCUP revision.

Theorem 31. *Let* $\langle \mathcal{AF}^1, LabArg^1 \rangle, \ldots, \langle \mathcal{AF}^n, LabArg^n \rangle$ *be a stable iterative SCUP revision. Then, for all preventing sets* $Args_{prev}$ *w.r.t.* $LabArg_{pref}$*, it holds that* $\exists \mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}$ *such that* $\mathcal{SCUPS} \cap Args_{prev} \neq \emptyset$.

Proof. Let $Args_{prev}$ be a preventing set w.r.t. $LabArg_{pref}$. By (the contrapositive of) Theorem 9, it holds that if $\mathcal{AF}^{\circledast}$ is a revision of $\mathcal{AF}$ w.r.t. some $Args \subseteq Ar$ by some $LabArg$ such that some revision labelling $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$, then $Args \cap Args_{prev} \neq \emptyset$. Since $\mathcal{AF}^n$ has a stable labelling $LabArg^n$ and since $\mathcal{AF}^n$ is a revision of $\mathcal{AF}$ w.r.t. $\bigcup_{\mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}} \mathcal{SCUPS}$ by $LabArg^n \cap (Ar \times \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\})$ it holds that $\exists \mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}$ such that $\mathcal{SCUPS} \cap Args_{prev} \neq \emptyset$. □

Example 34. Consider again $\mathcal{AF}_8$, illustrated at the top of Figure 18 (see page 33), and the stable iterative SCUP revision of $\mathcal{AF}_8$ discussed in Example 30 (see page 34). The set of arguments in SCUPs in every step of the stable iterative SCUP revision is $\biguplus \mathcal{SCUPS} = \{\{a, e\}, \{c\}\}$. For the preventing set $\{b, c, d\}$ w.r.t. $LabArg_{pref}$, which is not a subset of any SCUP w.r.t. $LabArg_{pref}$ (see Example 33), there exists the set $\{c\}$ in $\biguplus \mathcal{SCUPS}$, which shares an argument with $\{b, c, d\}$. Clearly, the preventing sets $\{a\}$ and $\{e\}$, which are subsets of SCUPs w.r.t. $LabArg_{pref}$, also have a non-empty intersection with a set in $\biguplus \mathcal{SCUPS}$, namely with $\{a, e\}$.

6.2. SCUPs versus Enforcement Sets

Next, we investigate the relationship between SCUPs and enforcement sets. We first show that a SCUP contains an argument from each enforcement set.

Theorem 32. *Let $Args_{SCUP}$ be a SCUP w.r.t. $LabArg_{pref}$. Then, for all enforcement sets $Args_{enf}$ w.r.t. $LabArg_{pref}$, it holds that $Args_{SCUP} \cap Args_{enf} \neq \emptyset$.*

Proof. By Theorem 30, there exists a preventing set $Args_{prev}$ w.r.t. $LabArg_{pref}$ such that $Args_{prev} \subseteq Args_{SCUP}$. Since by Theorem 10 it holds that for all enforcement sets $Args_{enf}$ w.r.t. $LabArg_{pref}$, $Args_{prev} \cap Args_{enf} \neq \emptyset$, it follows that $Args_{SCUP} \cap Args_{enf} \neq \emptyset$. □

Example 35. $\mathcal{AF}_8$, illustrated at the top of Figure 18 (see page 33), has two SCUPs w.r.t. $LabArg_{pref}$, namely $\{a\}$ and $\{e\}$ (see Example 29 on page 33). Both SCUPs contain an argument from each of the three enforcement sets w.r.t. $LabArg_{pref}$, i.e. $\{a, b, e\}$, $\{a, c, e\}$, $\{a, d, e\}$. In fact, both SCUPs are subsets of each enforcement set.

In contrast, $\mathcal{AF}_4$, illustrated at the top of Figure 10 (see page 22), has only one SCUP w.r.t. $LabArg_{pref}$, namely $\{c, d, e, f, g, h\}$. Again the SCUP contains an argument from each enforcement set w.r.t. $LabArg_{pref}$, i.e. from $\{d\}$, $\{g\}$, $\{c, e\}$, $\{c, f\}$, $\{e, h\}$, and $\{f, h\}$. In fact, here each enforcement set is a subset of the SCUP.

Note that, in general, SCUPs are not subsets of enforcement sets or vice versa. For instance, the SCUP $\{a, b, c\}$ of $\mathcal{AF}_3$ (see left of Figure 9 on page 15) is not a subset of any of the enforcement sets $\{a, e\}$, $\{b, e\}$, or $\{c, e\}$ or vice versa.

By Theorem 12, we know that if a revision has a stable labelling that is more committed than $LabArg_{pref}$, the set of arguments used for the revision must be a superset of some enforcement set. Since a stable iterative SCUP revision results in such a stable labelling, it follows that there exists an enforcement set that is a subset of the set of all arguments occurring in SCUPs of the iterative SCUP revision.

Theorem 33. *Let $\langle \mathcal{AF}^1, LabArg^1 \rangle, \ldots, \langle \mathcal{AF}^n, LabArg^n \rangle$ be a stable iterative SCUP revision. Then there exists an enforcement set $Args_{enf}$ w.r.t. $LabArg_{pref}$ such that $\forall A \in Args_{enf} : \exists \mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}$ with $A \in \mathcal{SCUPS}$.*

Proof. By Theorem 31, for each preventing set $Args_{prev}$ it holds that $\exists A \in Args_{prev}$ such that $\exists \mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}$ with $A \in \mathcal{SCUPS}$. It then follows from Theorem 11 that there exists an enforcement set $Args_{enf}$ such that such that $\forall A \in Args_{enf} : \exists \mathcal{SCUPS} \in \biguplus \mathcal{SCUPS}$ with $A \in \mathcal{SCUPS}$. □

Example 36. Consider again $\mathcal{AF}_8$ illustrated in Figure 18 (see page 33) and the stable iterative SCUP revision of $\mathcal{AF}_8$ discussed in Example 30 (see page 34). $\biguplus \mathcal{SCUPS} = \{\{a, e\}, \{c\}\}$, so there exists an enforcement set such that each argument is contained in a set in $\biguplus \mathcal{SCUPS}$, namely the enforcement set $\{a, c, e\}$.

The relation between enforcement sets and SCUPs implies that even though a stable iterative SCUP revision is not a *minimal* way of revising the AF to obtain a stable labelling, it includes the arguments that definitely have to be revised.

6.3. Responsible Cycles versus Enforcement and Preventing Sets

We now turn to the comparison between responsible cycles and enforcement and preventing sets. We first prove that there exists an enforcement set that consists only of arguments from responsible cycles.

Theorem 34. *Let $S = \{A \in Ar \mid \mathscr{C}$ is a responsible cycle w.r.t. $LabArg_{pref}$, $A \in \mathscr{C}\}$. Then there exists an enforcement set $Args$ w.r.t. $LabArg_{pref}$ such that $Args \subseteq S$.*

PROOF. By Proposition 16, there exists a labelling $LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$ such that, for all revisions $\mathcal{AF}^{\circledast}$ of $\mathcal{AF}$ w.r.t. S by $LabArg$ and all revision labellings $LabArg^{\circledast}$ of $\mathcal{AF}^{\circledast}$, $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. It then follows from Theorem 12 that there exists an enforcement set $Args$ w.r.t. $LabArg_{pref}$ such that $Args \subseteq S$. □

Example 37. Consider again $\mathcal{AF}_4$, illustrated at the top of Figure 10 (see page 22). The set of arguments in responsible cycles w.r.t. $LabArg_{pref}$ is $S = \{d, g\}$. There are two different enforcement sets that are subsets of S, namely $\{d\}$ and $\{g\}$. This example also illustrates that not all enforcement sets contain arguments that are part of a responsible cycle, e.g. the enforcement set $\{c, e\}$ is disjoint from S.

Note that not every responsible cycle shares arguments with some enforcement set. For instance, the responsible cycle $\{e\}$ w.r.t. the preferred labelling $LabArg_{pref}$ of $\mathcal{AF}_7$, illustrated in Figure 15 (see page 29), and the only enforcement set w.r.t. $LabArg_{pref}$, namely $\{c, d\}$, do not have any arguments in common.

Next, we show the connection between responsible cycles and preventing set. In particular, every preventing set comprises a responsible cycle.

Theorem 35. *Let $Args \neq \emptyset$ be a preventing set w.r.t. $LabArg_{pref}$. Then there exists a responsible cycle $\mathscr{C}$ w.r.t. $LabArg_{pref}$ such that $\mathscr{C} \subseteq Args$.*

PROOF. By Lemma 8, $LabArg_{pref}$ is not a stable labelling. Let $ArgsIO = \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$. Assume there exists no responsible cycle $\mathscr{C}$ w.r.t. $LabArg_{pref}$ such that $\mathscr{C} \subseteq Args$. Thus, $\mathcal{AF}{\downarrow_{Args}}$ comprises no odd-length cycles, so by Corollary 36 in [10] $\mathcal{AF}{\downarrow_{Args}}$ has a stable labelling $LabArg_{Args}$. By Lemma 46 in Appendix A, $LabArg_{pref}{\downarrow_{ArgsIO}}$ is compatible with $LabArg_{Args}$. Furthermore, by the same reasoning as in the proof of Proposition 19, $LabArg_{Args}$ is compatible with $LabArg_{pref}{\downarrow_{ArgsIO}}$. It follows from Lemma 40 in Appendix A that $LabArg_{Args} \cup LabArg_{pref}{\downarrow_{ArgsIO}}$ is a complete labelling of $\mathcal{AF}{\downarrow_{Args \cup ArgsIO}}$. Let $LabArg'$ be a labelling of

$Args' = Ar \setminus (Args \cup ArgsIO)$ such that $\texttt{out}(LabArg') = Args'$. Let $LabArg = LabArg_{Args} \cup LabArg_{pref}{\downarrow}_{ArgsIO} \cup LabArg'$. Clearly $LabArg_{pref} \sqsubset LabArg$. Furthermore, $\forall A \in Args$ it holds that A is legally labelled by $LabArg$. Contradiction, since by Definition 5, $\forall LabArg$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$ it holds that $\exists A \in Args$ such that A is illegally labelled by $LabArg$. □

Example 38. Consider again $\mathcal{AF}_4$, illustrated at the top of Figure 10 (see page 22). The two preventing sets w.r.t. the preferred labelling $LabArg_{pref}$ of $\mathcal{AF}_4$ are $\{c, d, g, h\}$ and $\{d, e, f, g\}$. Both contain a responsible cycle w.r.t. $LabArg_{pref}$, in this case even two responsible cycles, namely $\{d\}$ and $\{g\}$.

These results imply that odd-length cycles of arguments labelled `undec` by $LabArg_{pref}$ are an important characteristic of sets of arguments that prevent $LabArg_{pref}$ from being a stable labelling (Theorem 35). Furthermore, it is sufficient to revise (specific) arguments in odd-length cycles to obtain a stable labelling (of the revision) that is more committed than $LabArg_{pref}$ (Theorem 34).

7. Discussion and Related Work

Having introduced and analysed various characterisations of parts of an AF responsible if a given preferred labelling is not stable, we now discuss the implications for the non-existence of stable labellings, the choice of preferred semantics, and connections with related work.

7.1. Non-existence of Stable Labellings

Throughout this paper, we gave different characterisations of parts of an AF responsible that a given preferred labelling is not stable, irrespective of the existence of a stable labelling. That is, in general, the AF may have various preferred labellings, some that are stable and some that are not. These preferred labellings differ in their assignment of the labels `in` and `out` to certain arguments, in other words, an argument may be labelled `in` by one but `out` by another preferred labelling. This gives users the freedom to choose an assignment according to their own preferences.

In applications where decisiveness is required, users can thus decide whether they only care about finding *some* labelling without `undec` labels, in which case they can simply choose a stable labelling (if one exists), or they can choose one of the preferred labellings according to their preference concerning the assignment of `in` and `out` labels, and, if this preferred labelling is not stable, identify a suitable revision of the AF. If the AF has no stable labelling at all, the second situation is the only possible one. Our characterisations are thus versatile, as they can be applied both in scenarios where an AF has no stable labelling and in scenarios where a stable labelling exists, but the desired preferred labelling is not stable.

Since every stable labelling is a preferred labelling [11], it follows that if no stable labelling exists, then no preferred labelling is stable. Thus, in the case

of non-existence of stable labellings, our characterisations can explain the non-existence in terms of the preferred labellings not being stable in the following sense.

Proposition 36. *$\mathcal{AF}$ has no stable labelling if and only if for all preferred labellings $LabArg_{pref}$ of $\mathcal{AF}$ there exist a non-empty enforcement and a non-empty preventing set w.r.t. $LabArg_{pref}$, and there exist a responsible cycle and a SCUP w.r.t. $LabArg_{pref}$.*

Proof. Follows from Lemmas 5 and 8, and Propositions 15 and 20. □

Note that any one of the "responsibility conditions", e.g. the existence of a non-empty enforcement set, implies all the other "responsibility conditions", e.g. the existence of a SCUP. Which of our characterisations is most suitable for an application in question is left to the user to decide. As we have shown, each characterisation defines parts of an AF that are indeed responsible that a preferred labelling is not stable, and consequently that no stable labelling exists.

7.2. Preferred versus Other Approximation Labellings

The preferred semantics is not the only one to "approximate" the stable semantics. In particular, semi-stable labellings [13] are specific preferred labellings. Furthermore, stage and CF2 semantics [21] capture special types of maximal conflict-free sets. Even though stable labellings are always stage and CF2 labellings, stage and CF2 semantics do not generally adhere to the same basic properties as stable semantics in that, in particular, they do not satisfy admissibility [12]. Therefore, stage and CF2 semantics do not lend themselves to investigating the non-existence of stable labellings. However, it will be interesting future work to investigate whether similar methods as presented in this paper could help characterise why stage or CF2 labellings are not stable.

In contrast to CF2 and stage semantics, *semi-stable labellings* fulfil the admissibility property. They are defined as preferred labellings where the union of `in` and `out` labelled arguments is maximal (w.r.t. $\subseteq$) among all complete labellings [13]. We therefore also considered to use semi-stable labellings for our characterisations, instead of preferred labellings. In fact, most of our definitions apply to semi-stable labellings, too. For example, we could define enforcement and preventing sets w.r.t. a semi-stable rather than a preferred labelling since every semi-stable labelling is preferred.

However, with regards to SCUPs, semi-stable labellings lead to a problem: even though SCUPs can be defined with respect to a semi-stable instead of a preferred labelling, stable iterative SCUP revisions may *not* exist when defining SCUPs with respect to a semi-stable labelling. The reason is that Theorem 25 and Corollary 26 are not guaranteed to hold for semi-stable labellings, i.e. a SCUP revision may not have a semi-stable labelling that is more committed than the semi-stable labelling of the original AF, as illustrated by the following example.

Example 39. Let $\mathcal{AF}_9$ be the AF on the left of Figure 21, which also illustrates the only preferred and only semi-stable labelling $LabArg_{pref}$ of $\mathcal{AF}_9$. The only SCUP w.r.t. $LabArg_{pref}$ is $\{a\}$. A SCUP revision $\mathcal{AF}_9^{\circledast}$ of $\mathcal{AF}_9$ and its SCUP revision labelling $LabArg^{\circledast}$ are shown on the right of Figure 21. The left of Figure 22 illustrates the only preferred labelling of $\mathcal{AF}_9^{\circledast}$ that is more committed than $LabArg^{\circledast}$ and $LabArg_{pref}$. Note that this preferred labelling is not a semi-stable labelling of $\mathcal{AF}_9^{\circledast}$. The only semi-stable labelling of $\mathcal{AF}_9^{\circledast}$ is illustrated on the right of Figure 22. It is not more (or equally) committed than $LabArg^{\circledast}$. The same problem arises if the SCUP is revised in such a way that a is labelled `out` in the SCUP revision labelling, as illustrated at the top of Figure 23. The only semi-stable labelling of the SCUP revision is shown at the bottom of Figure 23, which is not more or equally committed than the SCUP revision labelling or $LabArg_{pref}$.

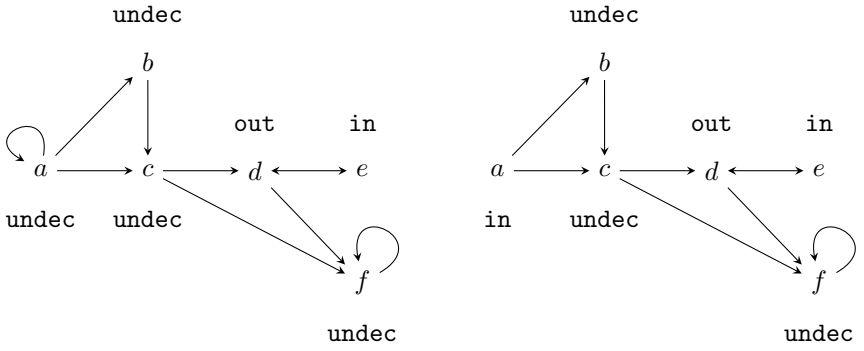

Figure 21: Left – The only preferred and semi-stable labelling of $\mathcal{AF}_9$. Right – A SCUP revision $\mathcal{AF}_9^{\circledast}$ of $\mathcal{AF}_9$ and a SCUP revision labelling.

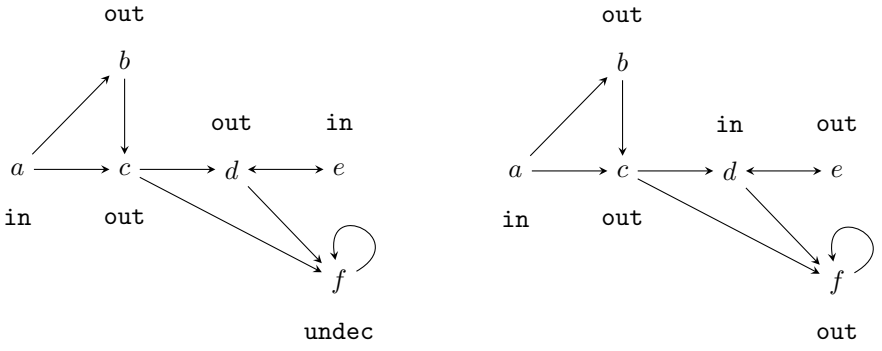

Figure 22: A preferred labelling of $\mathcal{AF}_9^{\circledast}$ that is more committed than $LabArg_{pref}$ (left) and the only semi-stable labelling of $\mathcal{AF}_9^{\circledast}$ (right).

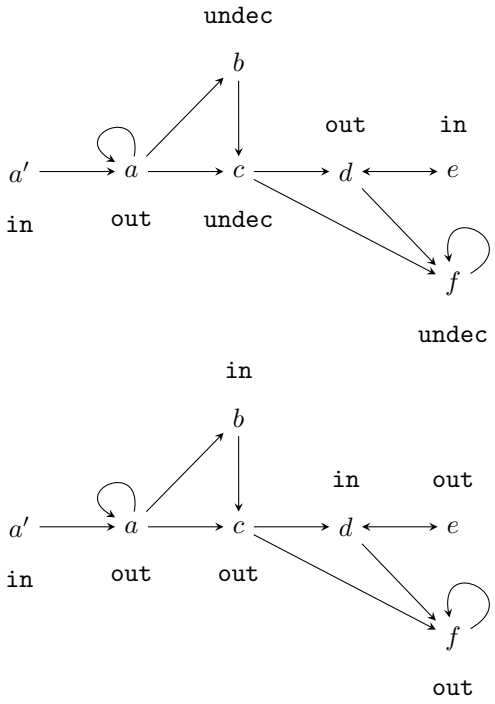

Figure 23: Another SCUP revision of $\mathcal{AF}_9$ and a SCUP revision labelling (top), and the only semi-stable labelling of this SCUP revision (bottom).

The problem with defining SCUPs with respect to semi-stable rather than preferred labellings is thus that iterative SCUP revisions cannot be applied, unless we are prepared to change the labels of arguments already labelled `in` and `out` by the semi-stable labelling of the original AF. However, this would defeat the spirit of our work, as we are interested in why a *particular* labelling is not a stable labelling. Of course, one could start with a semi-stable labelling and then use preferred labellings in the iterative SCUP revision.

7.3. Related Work

Related to our work on stable semantics, Baumann and Strass [22] focus on the question *how many* stable extensions an AF has on average and what the maximal number of stable extensions is. Furthermore, Dunne and Bench-Capon [23] investigate AFs whose stable and preferred extensions coincide, so-called coherent AFs, and thus deal with AFs that *always* have a stable extension.

To the best of our knowledge, the only work investigating the non-existence of stable extensions or labellings is by Nouioua and Würbel [24], who propose a revision operator that transforms an AF without stable extensions into one with a stable extension. Their setting is different from ours as they assume that the AF in question, which has no stable extension, was obtained from

an addition of arguments and attacks to some original AF. Assuming that the added arguments and attacks are "correct", they restrict the structural change performed by the revision operator to the original AF. In contrast, our work aims at characterising parts of an AF *responsible* for the non-existence of stable labellings and only uses the notion of structural change to prove that our characterised parts are indeed responsible. Furthermore, Nouioua and Würbel's approach differs from ours in various ways: Firstly, they revise an AF through a particular structural change, namely the deletion of attacks, whereas in our approach the addition of arguments and attacks is allowed, too. Importantly, in some cases, an enforcement labelling may be such that a mere deletion of attacks does not yield the desired enforcement, in other words, adding arguments may be necessary to obtain the desired enforcement.[6] Furthermore, their approach is not concerned with preserving a particular preferred, or even the grounded, labelling when performing the structural change. Another difference is that they are concerned with minimal (w.r.t. cardinality) changes that guarantee the existence of a stable extension. In our approach minimality also plays a role, as enforcement sets are minimal (w.r.t. set inclusion) sets of arguments used to obtain a stable labelling. However, as far as structural changes are concerned we do not make any minimality assumptions.

Like us, Baroni, Giacomin and Liao [25] are interested in arguments labelled `undec`. However, rather than investigating how `undec` labels can be turned into definite `in` or `out` labels, they argue that undecidedness is desirable in some situations and review various semantics that include different notions of "undecidedness".

Caminada and Pigozzi [26] also investigate labellings that are "as close as possible" to a given labelling and fulfil certain conditions, e.g. down-admissible labellings are the closest admissible labellings less committed than the given labelling and up-complete labellings are the closest complete labellings more committed than the given labelling. Instead, our enforcement labellings define more committed labellings that are "closest" to a stable labelling in terms of a minimal set of *illegally* labelled arguments, so they are not complete labellings. Another idea shared with their work is that of an iterative re-labelling (contraction and expansion functions) to obtain a labelling that legally labels all arguments. This is related to our iterative SCUP revision, where arguments in SCUPs are iteratively re-labelled (and enforced through structural revision) until a stable labelling (of a revision) is obtained.

The ideas of enforcement and preventing sets introduced here are similar in spirit as Reiter's [27] diagnoses and conflict sets. They describe components of a system causing abnormal behaviour of the system. Similarly to enforcement and preventing sets, they form duals (in terms of Reiter, one is a "hitting set" of the other). Inspired by that, Ignatiev et al. [28] define similar duals for propositional logic in terms of minimal corrections subsets and minimal unsatisfiable subsets.

[6] An example is a self-attacking argument whose label should be `out`, achievable by adding a new argument attacking the self-attacking one.

In the following sections, we review some further strands of research sharing particular aspects with our work.

7.3.1. Cycles in Argumentation Frameworks

Recently, cycles (of attacking arguments) in AFs have received considerable attention, including a special issue of the Journal of Logic and Computation [29]. Many authors regard the behaviour of preferred semantics with respect to cycles as "problematic", as it treats odd-length and even-length cycles differently. In particular, arguments in odd-length cycles can often only be labelled `undec`, as is the case for our responsible cycles, whereas arguments in even-length cycles can be alternately labelled `in` and `out`.

Baroni, Giacomin and Guida [16] discuss this "problematic" behaviour of preferred semantics and introduce the *CF2* semantics for AFs, which "correctly" handles odd- and even-length cycles. Dvořák and Gaggl [30] extend the CF2 semantics to the so-called *stage2* semantics, which fulfils some additional properties. Arieli [31] introduces a new family of *conflict-tolerant* semantics, where the conflict-freeness requirement for extensions is dropped. Therefore, odd- and even-length cycles are treated the same by the new semantics. Gabbay [32] defines another family of new semantics, able to handle the "problematic" behaviour of the preferred semantics with regards to cycles. In the *loop busting* semantics no argument is labelled `undec`. The procedure for computing the semantics has similarities with ideas used in our approach, since it iteratively applies a specific type of revision of initial SCCs. More precisely, an argument in an initial SCC of the `undec`-part with respect to the grounded extension is chosen and a new attacker is added. Then the grounded labelling of the new AF is computed and the same procedure is performed iteratively for the new AF restricted to arguments labelled `undec`. The iterative SCUP revision introduced here applies a similar approach since an initial SCC of the `undec`-part (i.e. a SCUP) is revised and the revision is then repeated on the AF restricted to arguments still labelled `undec`. However, we allow for any revision and use the *preferred* rather than grounded semantics. Bodanza and Tohmé [33] propose two new semantics for handling odd-length cycles: the first one allows to accept arguments attacked by an odd-length cycle, and the second one additionally allows to accept single arguments in an odd-length cycle. Both types of semantics yield labellings that are more committed than preferred labellings.

In contrast to the aforementioned works, Bench-Capon [34] argues that the way the preferred semantics handles cycles is not "problematic", by providing an interpretation of even-length cycles as dilemmas and odd-length cycles as paradoxes. He argues that using this point of view, it is reasonable that arguments in odd-length cycles are neither true nor false, and that consequently their justification status cannot be decided.

Note that the motivation of our approach is completely different from the motivations of the works reviewed above. We do not make any claims about whether or not the preferred semantics handles cycles "correctly", and are therefore not concerned with new semantics. Instead, we characterise specific parts of

an AF, which turn out to comprise odd-length cycles, as *responsible* that a preferred labelling is not stable, which also characterise parts of the AF responsible for the non-existence of stable labellings.

Like us, Baumann and Woltran [35] are not concerned with the "correct" or "incorrect" behaviour of semantics regarding odd-length cycles. Instead, they study the role of self-attacking arguments, i.e. cycles of length one, with regards to the equivalence of AFs.

7.3.2. Splitting Argumentation Frameworks

Two of our structural characterisations build upon the idea of SCCs introduced in [16]. We investigate a particular type of SCCs, namely specific *initial* SCCs, and use Baroni, Giacomin and Guida's results [16] that the preferred and stable semantics are SCC-recursive, i.e. that the preferred and stable extensions (or equivalently labellings) of an AF can be obtained by computing the respective extensions for initial SCCs and using them recursively for computing the extensions of the following SCCs. Liao [36] shows how the semantics of an AF can be computed through the step-wise computation of semantics of SCCs, and Baroni et al. [18] generalise the decomposability results about SCCs, showing how complete labellings of an AF can be computed by combining complete labellings of *arbitrary* parts of the AF. We apply and extend Baroni et al.'s results for a particular partitions of an AF into the set of arguments labelled `in` or `out` by a preferred labelling, and (a subset of the) arguments labelled `undec`.

Our results about combining a labelling of a SCUP with the `in` and `out` labels in a preferred labelling are also related to the *splitting* results of Baumann et al. [37, 38]. They show that, for the stable semantics, extensions of an AF can be obtained by splitting the AF into two parts and computing the extensions of the two parts using a method that takes the extensions of the respective other part into account. Another related approach was introduced by Rienstra et al. [39], who propose *multi-sorted extensions* as a new semantics of an AF with respect to a partition of the AF. A multi-sorted extension is such that its restriction to a part coincides with a given semantics for this part. This approach is conceptually related to our work, which combines the stable labellings of parts of the AF, namely SCUPs, with `in` and `out` labels from a preferred labelling.

7.3.3. Dynamics in Argumentation Frameworks

The study of dynamics in AFs has received considerable attention in recent years. Our work investigates the dynamics of AFs from a special angle, since we are not concerned with the exact structural change of an AF and its effect (as e.g. in [40]), but rather with the existence of *some* structural change resulting in the enforcement of a desired label for an argument. Importantly, *which* structural change is chosen is not of importance for our work.

Liao, Jin, and Koons [41] introduce a general approach for computing extensions of an AF that has been structurally changed, allowing for any number of additions and deletions of arguments and attacks. The idea is that, in order to compute the semantics of the new AF, only the semantics of the part of the AF that is *affected* by the structural change has to be re-computed. The

semantics of the *unaffected* part stays the same as before the structural change and only "conditions" the extensions of the affected part. This idea is related to our iterative SCUP revisions, where we do not change the labels of arguments labelled `in` or `out` in the SCUP revision labelling (they are "unaffected"), but only of those labelled `undec` that are "conditioned" by the `in` and `out` labels of the SCUP revision labelling.

The work of Booth et al. [42] is of similar spirit to our work, but concerned with the complete rather than the stable semantics: they investigate how to turn a non-complete labelling into a complete one through a structural change. In contrast to our work, Booth et al. assume an *intended* complete labelling, whereas for our approach no intended stable labelling is required.

Baumann and Brewka [17] were the first to investigate whether certain sets of arguments can be *enforced* as an extension according to a chosen semantics. In contrast to our general revisions, they only allow structural changes called "expansions", where arguments and attacks can be added, and new attacks must involve a new argument. Baumann and Brewka prove that for certain kinds of expansions, all arguments that are part of extensions before the structural change are also part of extensions after the structural change. In line with their work, we show that for any revision w.r.t. an enforcement set by an enforcement labelling, a stable labelling is obtained in which all previously `in`- and `out`-labelled arguments keep their labels. Baumann [19] as well as Coste-Marquis et al. [20] then studied how to enforce a set of arguments through a *minimal* structural change of adding or deleting attacks. Similarly, we prove that enforcement sets are minimal sets of arguments that, when used for a revision, yield a stable labelling. Coste-Marquis et al. [43] introduce a whole family of revision operators that can be used for enforcement, generalising revision operators defined by others, e.g. [44, 45, 46]. Other authors [47, 48, 49] study enforcements as logical formulae to be satisfied through structural change. It is important to note that even though enforcement is a related problem, we do not assume a set of arguments to be "enforced" as an extension of the SCUP. In contrast, we only require that some stable extension exists after the revision. However, the previously mentioned approaches could be used for enforcing a certain set of arguments as a stable extension of a SCUP.

8. Conclusion

In this paper, we investigated why a preferred labelling may not be stable. Our contributions can be summarised as follows:

1. We gave three labelling-based and three structural characterisations of sets of `undec` arguments deemed responsible that a given preferred labelling is not a stable labelling.
2. We proved that our characterised sets of arguments are indeed *responsible* by examining the effect of enforcing the label `in` or `out` for responsible arguments in these sets through some structural revision. In particular:

- each enforcement set provides a necessary and sufficient condition for turning a non-stable preferred labelling (of the original AF) into a stable labelling (of the revised AF);
- the union of all preventing sets as well as the union of all responsible cycles provide a sufficient condition for turning a non-stable preferred labelling (of the original AF) into a stable labelling (of the revised AF);
- the union of all SCUPs in a stable iterative SCUP revision provides a sufficient condition for turning a non-stable preferred labelling (of the original AF) into a stable labelling (of the revised AF);
- each preventing set provides a necessary and sufficient condition for the failure of turning a non-stable preferred labelling (of the original AF) into a stable labelling (of the revised AF);
- each SCUP provides a sufficient condition for the failure of turning a non-stable preferred labelling (of the original AF) into a stable labelling (of the revised AF).

3. The stable iterative SCUP revision can be used to constructively obtain a stable labelling (of a revision) from a given preferred labelling (of the original AF).
4. Our characterisations also explain the non-existence of stable labellings, in the sense that they characterise for each preferred labelling why it is not a stable labelling.

We compared our labelling-based and structural characterisations, proving that SCUPs provide a constructive approximation of our precise labelling-based characterisations. Furthermore, our comparison shows that odd-length cycles are an important feature of all our characterisations.

One of our results states that a SCUP revision has a preferred labelling that is more (or equally) committed than the chosen preferred labelling of the original AF. We did, however, not discuss how to obtain such a preferred labelling. The naïve approach is to compute all preferred labellings of the SCUP revision, compare them to the SCUP revision labelling, and choose a preferred labelling that is more or equally committed than the SCUP revision labelling. There exists a large variety of computational tools to solve this task, see [50] for an overview. Another possibility is to use a slight modification of the algorithm by Cerutti et al. [51] for computing preferred labellings through an iterative procedure that splits the AF into different parts.

Future work includes studying the complexity and developing a tool for finding our different notions of responsible sets of arguments and revising the AF based on these sets combined with the user's choice of labels (e.g. of arguments in a SCUP). Furthermore, we plan to investigate how different revision operators from the literature (e.g. [44, 45, 46, 43]) can be combined with our characterisations.

Various authors have shown that there is a semantic correspondence between logic programs and AFs that encode the same information as the logic program

[10, 52, 53]. It follows from these results that if a logic program has no stable models, the encoding AF has no stable labellings. A future direction of research is thus to study whether our notions of responsible sets and revisions based on these sets can be carried over to logic programs, and how they relate to existing work on the inconsistency and debugging of logic programs, e.g. [54, 55, 56, 57, 58, 59, 60, 61].

We here only considered semantics that assign one of three labels to arguments. Other types of semantics for AFs rank arguments or assign a numerical value to each argument, e.g. [62, 63, 64, 65, 66, 67, 68, 69]. It will be interesting to investigate if there is any connection between our responsible sets of arguments and their numerical value or rank according to these different semantics.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable feedback.

Appendix A. Auxiliary Results

Lemma 37. *Let $LabArg$ be a labelling of $\mathcal{AF}$, $Args \subseteq Ar$ and $A \in Ar \setminus Args$. Let $\mathcal{AF}^{\circledast}$ be a revision of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$ and $LabArg^{\circledast}$ a revision labelling of $\mathcal{AF}^{\circledast}$. Then A is legally labelled by $LabArg$ in $\mathcal{AF}$ if and only if A is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$.*

Proof. From left to right: Let A be legally labelled by $LabArg$. By Definition 1, $(B, A) \in Att$ if and only if $(B, A) \in Att^{\circledast}$. Furthermore, $LabArg^{\circledast}(A) = LabArg(A)$ and for all $B \in Ar$, $LabArg^{\circledast}(B) = LabArg(B)$. Since it only depends on the labels of attackers of A whether or not A is legally labelled, it follows that A is legally labelled by $LabArg^{\circledast}$. The proof of the opposite direction is analogous. □

Lemma 38. *Let $LabArg$ and $LabArg'$ be two labellings of $\mathcal{AF}$ such that $LabArg \sqsubseteq LabArg'$. Then, $\forall A \in \mathtt{in}(LabArg) \cup \mathtt{out}(LabArg)$ it holds that if A is legally labelled by $LabArg$, then A is legally labelled by $LabArg'$.*

Proof. Let $A \in \mathtt{in}(LabArg)$. Then, for all attackers B of A, $B \in \mathtt{out}(LabArg)$. By definition of $LabArg'$, $A \in \mathtt{in}(LabArg')$ and for all attackers B of A, $B \in \mathtt{out}(LabArg')$. Thus, A is legally labelled $\mathtt{in}$ by $LabArg'$. Let $A \in \mathtt{out}(LabArg)$. Then there exists an attacker B of A such that $B \in \mathtt{in}(LabArg)$. By definition of $LabArg'$, $A \in \mathtt{out}(LabArg')$ and $B \in \mathtt{in}(LabArg')$. Thus, A is legally labelled $\mathtt{out}$ by $LabArg'$. □

Lemma 39. *Let $Args$ be an enforcement set w.r.t. $LabArg_{pref}$ and $LabArg$ an enforcement labelling w.r.t. $Args$. Then, $\forall A \in Ar \setminus Args$, A is legally labelled by $LabArg$.*

Proof. Let $A \in Ar \setminus Args$. By Definition 4, if $A \in \mathtt{undec}(LabArg_{pref})$, then A is legally labelled by $LabArg$. If $A \in \mathtt{in}(LabArg_{pref}) \cup \mathtt{out}(LabArg_{pref})$, then by Lemma 38 A is legally labelled by $LabArg$, since $LabArg_{pref} \sqsubset LabArg$. □

Definition 12 (Compatible Labelling). Let $Args_1, Args_2 \subseteq Ar$ such that $Args_1 \cap Args_2 = \emptyset$ and $Args_1 \cup Args_2 = Ar$. Let $LabArg_1$ be a labelling of $\mathcal{AF}{\downarrow_{Args_1}}$ and $LabArg_2$ a labelling of $\mathcal{AF}{\downarrow_{Args_2}}$. $LabArg_1$ is *compatible* with $LabArg_2$ if and only if $LabArg_1$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, Att \cap (Args_2 \times Args_1))$.

Lemma 40. *Let $Args_1, Args_2 \subseteq Ar$ such that $Args_1 \cap Args_2 = \emptyset$ and $Args_1 \cup Args_2 = Ar$. Let $LabArg_1$ be a labelling of $\mathcal{AF}{\downarrow_{Args_1}}$ and $LabArg_2$ a labelling of $\mathcal{AF}{\downarrow_{Args_2}}$. $LabArg = LabArg_1 \cup LabArg_2$ is a complete labelling of $\mathcal{AF}$ if and only if $LabArg_1$ is compatible with $LabArg_2$ and $LabArg_2$ is compatible with $LabArg_1$.*

Proof. Follows from Definition 12 and Theorem 3 in [18]. □

Lemma 41. *Let $Args_1, Args_2 \subseteq Ar$ such that $Args_1 \cap Args_2 = \emptyset$, $Args_1 \cup Args_2 = Ar$, and $Args_2$ does not attack $Args_1$. Let $LabArg_1$ be a complete labelling of $\mathcal{AF}{\downarrow_{Args_1}}$ and $LabArg_2$ a labelling of $\mathcal{AF}{\downarrow_{Args_2}}$. $LabArg = LabArg_1 \cup LabArg_2$ is a complete labelling of $\mathcal{AF}$ if and only if $LabArg_2$ is compatible with $LabArg_1$.*

Proof. From left to right: Let $LabArg = LabArg_1 \cup LabArg_2$ be a complete labelling of $\mathcal{AF}$. Then, by Lemma 40, $LabArg_2$ is compatible with $LabArg_1$. From right to left: Let $LabArg_2$ be compatible with $LabArg_1$. Since $LabArg_1$ is a complete labelling of $\mathcal{AF}{\downarrow_{Args_1}}$, by Proposition 1 in [18] $LabArg_1$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow_{Args_1}}, \emptyset, \emptyset, \emptyset)$. Since $Args_2$ does not attack $Args_1$, it follows that $LabArg_1$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, \emptyset)$, so $LabArg_1$ is compatible with $LabArg_2$. Thus by Lemma 40, $LabArg_1 \cup LabArg_2$ is a complete labelling of $\mathcal{AF}$. □

We can generalise Lemma 41 to SCCs.

Corollary 42. *Let $Args_1, \ldots, Args_n$ ($n \geq 1$) be a sequence of all SCCs of $\mathcal{AF}$ and for all $i \neq j$, $Args_i \neq Args_j$, and if $Args_i$ is attacked by $Args_k$ ($i \neq k$), then $k < i$. Let $LabArg_i$ be a labelling of $\mathcal{AF}{\downarrow_{Args_i}}$. Then $LabArg = LabArg_1 \cup \ldots \cup LabArg_n$ is a complete labelling of $\mathcal{AF}$ if and only if $LabArg_1$ is a complete labelling of $Args_1$ and $LabArg_i$ is compatible with $LabArg_1 \cup \ldots \cup LabArg_{i-1}$ for all $i \in \{2 \ldots n\}$.*

Lemma 43. *Let $Args_1, Args_2 \subseteq Ar$ such that $Args_1 \cap Args_2 = \emptyset$ and $Args_1 \cup Args_2 = Ar$. Let $LabArg_1$ be a labelling of $\mathcal{AF}{\downarrow_{Args_1}}$ and $LabArg_2$ a labelling of $\mathcal{AF}{\downarrow_{Args_2}}$. If $\forall A \in Args_1$ it holds that A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}$, then $LabArg_1$ is compatible with $LabArg_2$.*

Proof. Let $LabArg = LabArg_1 \cup LabArg_2$ and let $A \in Args_1$.

- If $A \in \texttt{in}(LabArg_1)$, then clearly $A \in \texttt{in}(LabArg)$. Thus, $\forall B$ attacking A, $B \in \texttt{out}(LabArg)$. It follows that if $B \in Args_1$, $B \in \texttt{out}(LabArg_1)$, and if $B \in Args_2$, then $B \in \texttt{out}(LabArg_2)$.

- If $A \in \texttt{out}(LabArg_1)$, then clearly $A \in \texttt{out}(LabArg)$. Thus, $\exists B$ attacking A such that $B \in \texttt{in}(LabArg)$. It follows that $B \in Args_1$ and $B \in \texttt{in}(LabArg_1)$, or $B \in Args_2$ and $B \in \texttt{in}(LabArg_2)$.

- If $A \in \texttt{undec}(LabArg_1)$, then clearly $A \in \texttt{undec}(LabArg)$. Thus, $\forall B$ attacking A, $B \notin \texttt{in}(LabArg)$, and $\exists C$ attacking A such that $C \in \texttt{undec}(LabArg)$. It follows that if $B \in Args_1$, $B \notin \texttt{in}(LabArg_1)$, and if $B \in Args_2$, then $B \notin \texttt{in}(LabArg_2)$. Furthermore, it follows that $C \in Args_1$ and $C \in \texttt{undec}(LabArg_1)$ or $C \in Args_2$ and $C \in \texttt{undec}(LabArg_2)$.

Thus, all $A \in Args_1$ satisfy the conditions in Definition 12, so $Args_1$ is compatible with $Args_2$. □

Lemma 44. *Let* $ArgsIO = \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ *and* $ArgsU = \texttt{undec}(LabArg_{pref})$. *Then* $LabArg_{pref}{\downarrow_{ArgsU}}$ *is the only complete labelling w.r.t.* $(\mathcal{AF}{\downarrow_{ArgsU}}, ArgsIO, LabArg_{pref}{\downarrow_{ArgsIO}}, Att \cap (ArgsIO \times ArgsU))$.

Proof. Since $LabArg_{pref}$ is a complete labelling of $\mathcal{AF}$, it holds by Lemma 40 that $LabArg_{pref}{\downarrow_{ArgsIO}}$ is compatible with $LabArg_{pref}{\downarrow_{ArgsU}}$ and vice versa. By Definition 12, it follows that $LabArg_{pref}{\downarrow_{ArgsU}}$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow_{ArgsU}}, ArgsIO, LabArg_{pref}{\downarrow_{ArgsIO}}, Att \cap (ArgsIO \times ArgsU))$.
To prove that $LabArg_{pref}{\downarrow_{ArgsU}}$ is the only such labelling, assume there exists a labelling $LabArgU \neq LabArg_{pref}{\downarrow_{ArgsU}}$ of $\mathcal{AF}{\downarrow_{ArgsU}}$ such that $LabArgU$ is a complete labelling w.r.t.
$(\mathcal{AF}{\downarrow_{ArgsU}}, ArgsIO, LabArg_{pref}{\downarrow_{ArgsIO}}, Att \cap (ArgsIO \times ArgsU))$.
Thus by Definition 12, $LabArgU$ is compatible with $LabArg_{pref}{\downarrow_{ArgsIO}}$.
Clearly, $LabArg_{pref} \sqsubset LabArg_{pref}{\downarrow_{ArgsIO}} \cup LabArgU$, so by Lemma 38 all $A \in ArgsIO$ are legally labelled by $LabArg_{pref}{\downarrow_{ArgsIO}} \cup LabArgU$. Then, by Lemma 43, $LabArg_{pref}{\downarrow_{ArgsIO}}$ is compatible with $LabArgU$. It follows by Lemma 40, that $LabArg_{pref}{\downarrow_{ArgsIO}} \cup LabArgU$ is a complete labelling of $\mathcal{AF}$. Contradiction, since $LabArg_{pref} \sqsubset LabArg_{pref}{\downarrow_{ArgsIO}} \cup LabArgU$ and $LabArg_{pref}$ is a preferred labelling. Thus, $LabArg_{pref}{\downarrow_{ArgsU}}$ is the only complete labelling w.r.t.
$(\mathcal{AF}{\downarrow_{ArgsU}}, ArgsIO, LabArg_{pref}{\downarrow_{ArgsIO}}, Att \cap (ArgsIO \times ArgsU))$. □

Lemma 45. *Let* $Args_1, Args_2 \subseteq Ar$ *such that* $Args_1 \cap Args_2 = \emptyset$ *and* $Args_1 \cup Args_2 = Ar$. *Let* $LabArg_2$ *be a labelling of* $\mathcal{AF}{\downarrow_{Args_2}}$. *Then there exists a labelling* $LabArg_1$ *of* $\mathcal{AF}{\downarrow_{Args_1}}$ *such that* $LabArg_1$ *is compatible with* $LabArg_2$.

Proof. Since Definition 12 mirrors the definition of canonical local function of the complete semantics (Definition 24 in [18]), a labelling $LabArg_1$ of $Args_1$ is compatible with a labelling $LabArg_2$ of $Args_2$ if and only if $LabArg_1$ is an element of the canonical local function of the complete semantics of the argumentation framework with input $(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, Att \cap (Args_2 \times Args_1))$. By Definition 13 in [18], the canonical local function of the complete semantics of $(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, Att \cap (Args_2 \times Args_1))$ can be computed via the complete labellings of the standard argumentation framework of

$(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, Att \cap (Args_2 \times Args_1))$. Since a standard argumentation framework always exists, it has a complete labelling, so the canonical local function of the complete semantics for $(\mathcal{AF}{\downarrow_{Args_1}}, Args_2, LabArg_2, Att \cap (Args_2 \times Args_1))$ is non-empty. Thus $LabArg_1$ exists. □

Lemma 46. *Let* $ArgsIO = \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ *and* $ArgsU \subseteq \texttt{undec}(LabArg_{pref})$. *Let* $LabArgIO = LabArg_{pref}{\downarrow_{ArgsIO}}$ *and let* $LabArgU$ *be some labelling of* $\mathcal{AF}{\downarrow_{ArgsU}}$. *Then* $LabArgIO$ *is compatible with* $LabArgU$.

Proof. We note that $\forall B \in ArgsU$ attacking some $A \in ArgsIO$ it holds that $A \in \texttt{out}(LabArgIO)$ since arguments labelled `in` are not attacked by arguments labelled `undec` in $LabArg_{pref}$. Let $A \in ArgsIO$.

- If $A \in \texttt{in}(LabArgIO)$, then, for all $B \in ArgsIO$ attacking A, it holds that $B \in \texttt{out}(LabArgIO)$ since $LabArg_{pref}$ is a complete labelling. Furthermore, no $B \in ArgsU$ attacks A.
- If $A \in \texttt{out}(LabArgIO)$, then there exists $B \in ArgsIO$ attacking A such that $B \in \texttt{in}(LabArgIO)$ since $LabArg_{pref}$ is a complete labelling.

Thus, $LabArgIO$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow_{ArgsIO}}, ArgsU, LabArgU, Att \cap (ArgsU \times ArgsIO))$, and therefore $LabArgIO$ is compatible with $LabArgU$. □

Appendix B. Proofs

Proof (of Lemma 1). Let $\mathcal{AF}^{\circledast} = \langle Ar^{\circledast}, Att^{\circledast} \rangle$ be such that

- $Ar^{\circledast} = Ar \cup \{X\}$ where $X \notin Ar$ and
- $Att^{\circledast} = (Att \setminus \{(B, A) \in Att \mid A \in Args, A \in \texttt{in}(LabArg) \cup \texttt{undec}(LabArg)\})$
  $\cup\ (\{(X, A) \mid A \in Args, A \in \texttt{out}(LabArg)\}$
  $\cup \{(A, A) \mid A \in Args, A \in \texttt{undec}(LabArg)\})$.

Let $LabArg^{\circledast} = LabArg \cup \{(X, \texttt{in})\}$. Then clearly $Ar \subseteq Ar^{\circledast}$ and $\{(A, B) \in Att \mid B \in Ar \setminus Args\} = \{(A, B) \in Att^{\circledast} \mid B \in Ar \setminus Args\}$, and $\forall C \in Ar$: $LabArg^{\circledast}(C) = LabArg(C)$.
Let $A \in Args$. If $A \in \texttt{in}(LabArg^{\circledast})$, then A is not attacked by any argument B in $\mathcal{AF}^{\circledast}$, so trivially, for all attackers B of A in $\mathcal{AF}^{\circledast}$, $B \in \texttt{out}(LabArg^{\circledast})$. Thus, A is legally labelled `in` by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. If $A \in \texttt{out}(LabArg^{\circledast})$, then A is attacked by X in $\mathcal{AF}^{\circledast}$ and $X \in \texttt{in}(LabArg^{\circledast})$, so A is legally labelled `out` by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. If $A \in \texttt{undec}(LabArg^{\circledast})$, then A is only attacked by itself in $\mathcal{AF}^{\circledast}$. Thus, there exists an attacker of A in $\mathcal{AF}^{\circledast}$ labelled `undec` by $LabArg^{\circledast}$ and there exists no attacker of A in $\mathcal{AF}^{\circledast}$ labelled `in` by $LabArg^{\circledast}$, so A is legally labelled `undec` by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$.
Since furthermore $X \in Ar^{\circledast} \setminus Ar$ is legally labelled `in` by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$, $\mathcal{AF}^{\circledast}$ and $LabArg^{\circledast}$ satisfy the conditions in Definition 1, so $\mathcal{AF}^{\circledast}$ is a revision of $\mathcal{AF}$ w.r.t. $Args$ by $LabArg$. □

PROOF (OF PROPOSITION 3). Since $\texttt{undec}(LabArg) = \emptyset$, it follows from Observation 2 that $\texttt{undec}(LabArg^{\circledast}) = \emptyset$. Furthermore, by Definition 1 all $A \in Ar^{\circledast} \setminus Ar$ are legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Let $B \in Ar$. If $B \in Args$, then, by Definition 1, B is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. If $B \notin Args$, then $B \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$, so B is legally labelled by $LabArg_{pref}$ in $\mathcal{AF}$. By Lemma 38 in Appendix A, B is legally labelled by $LabArg$ in $\mathcal{AF}$, and by Lemma 37 in Appendix A, B is legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Since all arguments in $\mathcal{AF}^{\circledast}$ are legally labelled by $LabArg^{\circledast}$ and $\texttt{undec}(LabArg^{\circledast}) = \emptyset$, $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. □

PROOF (OF LEMMA 5).

1. Let $Args = \texttt{undec}(LabArg_{pref})$. Clearly there exists some $LabArg$ with $LabArg_{pref} \sqsubseteq LabArg$ and $\texttt{undec}(LabArg) = \emptyset$. Then trivially, $\forall A \in \texttt{undec}(LabArg_{pref}) \setminus Args = \emptyset$ it holds that A is legally labelled by $LabArg$. Thus, $Args$ and $LabArg$ satisfy the conditions in Definition 4, but $Args$ may not be a minimal set satisfying the conditions. If for all $Args_1 \subset Args$ and for all $LabArg'$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubseteq LabArg'$ and $\texttt{undec}(LabArg') = \emptyset$ there exists some $A \in \texttt{undec}(LabArg_{pref}) \setminus Args_1$ that is illegally labelled by $LabArg'$, then $Args$ is a minimal set satisfying the conditions in Definition 4, so it is an enforcement set (and $LabArg$ an enforcement labelling w.r.t. $Args$). Else, there is a smallest $Args_1 \subset Args$ satisfying that $\exists LabArg_1$ with $LabArg_{pref} \sqsubseteq LabArg_1$ and $\texttt{undec}(LabArg_1) = \emptyset$ such that $\forall A \in \texttt{undec}(LabArg_{pref}) \setminus Args_1$: A is legally labelled by $LabArg_1$. Thus, $Args_1$ is an enforcement set (and $LabArg_1$ an enforcement labelling).
2. Let $LabArg$ be an enforcement labelling w.r.t. $Args$. If $Args = \emptyset$, then, by Lemma 39 in Appendix A, all arguments in Ar are legally labelled by $LabArg$, so since $\texttt{undec}(LabArg) = \emptyset$, $LabArg$ is a stable labelling. Thus, $LabArg_{pref} = LabArg$ is a stable labelling.
   If $LabArg_{pref}$ is a stable labelling then $\texttt{undec}(LabArg_{pref}) = \emptyset$. Then $Args = \emptyset$ is the minimal set satisfying Definition 4 with $LabArg = LabArg_{pref}$ as the only enforcement labelling. □

PROOF (OF LEMMA 8).

1. Let $Args = \texttt{undec}(LabArg_{pref})$ and let $LabArg$ be such that $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$. Since $LabArg_{pref}$ is a maximal complete labelling, $\exists A \in Args$ such that A is illegally labelled by $LabArg$. Since this holds for all such labellings $LabArg$, $Args$ satisfies the conditions in Definition 5. However, $Args$ may not be a minimal set satisfying these conditions. If for all $Args_1 \subset Args$ there exists $LabArg_1$ with $LabArg_{pref} \sqsubset LabArg_1$ and $\texttt{undec}(LabArg_1) = \emptyset$ such that all $A \in Args_1$ are legally labelled by $LabArg_1$, then $Args$ is a minimal set satisfying the conditions in Definition 5, so it is a preventing set w.r.t. $LabArg_{pref}$. Else, there is a smallest $Args_1 \subset Args$ satisfying that $\forall LabArg'$ with $LabArg_{pref} \sqsubset LabArg'$ and $\texttt{undec}(LabArg') = \emptyset$ it holds that $\exists A \in Args_1$

such that A is illegally labelled by $LabArg'$. Then $Args_1$ is a preventing set w.r.t. $LabArg_{pref}$.

2. Assume $Args = \emptyset$ is a preventing set w.r.t. $LabArg_{pref}$ and $LabArg_{pref}$ is not a stable labelling. Then $\exists LabArg \sqsubset LabArg_{pref}$ and, by Definition 5, $\exists A \in Args$ such that A is illegally labelled. Contradiction since $\nexists A \in Args$, so $LabArg_{pref}$ is a stable labelling.
   Assume $LabArg_{pref}$ is a stable labelling. Then $\texttt{undec}(LabArg_{pref}) = \emptyset$ and $Args = \emptyset$ satisfies Definition 5. □

Proof (of Theorem 11). We prove that all $Args \in S$ are enforcement sets and that all enforcement sets are contained in S. We note that, by Lemma 8 $S_{prev} \neq \emptyset$. If $S_{prev} = \{\emptyset\}$ then $S = \{\emptyset\}$. Then, by Lemma 8, $LabArg_{pref}$ is a stable labelling and by Lemma 5 the empty set is the only enforcement set. If $S_{prev} \neq \{\emptyset\}$ then by Lemma 8 $\forall Args_{prev} \in S_{prev}$: $Args_{prev} \neq \emptyset$.

- Let $Args \in S$ and assume that $Args$ is not an enforcement set. Then either $Args$ is not a minimal set satisfying the conditions in Definition 4 or it does not satisfy the conditions at all.
  - In the first case, $\exists Args_{enf} \subset Args$ such that $Args_{enf}$ is an enforcement set. Since $Args$ is a minimal set satisfying that $\forall Args_{prev} \in S_{prev} : Args \cap Args_{prev} \neq \emptyset$, it follows that $\exists Args'_{prev} \in S_{prev}$ such that $Args_{enf} \cap Args'_{prev} = \emptyset$. Since $Args'_{prev}$ is a preventing set it holds that $\forall LabArg$ such that $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, $\exists A \in Args'_{prev}$ such that A is illegally labelled by $LabArg$. However, since $Args_{enf}$ is an enforcement set it holds that $\exists LabArg'$ such that, by Lemma 39 in Appendix A, $\forall B \in Ar \setminus Args_{enf}$: B is legally labelled by $LabArg'$. Contradiction since $B \in Args'_{prev}$.
  - In the second case, we note that $Args \subseteq \texttt{undec}(LabArg_{pref})$ since $\forall A \in Args : \exists Args_{prev}$ such that $A \in Args_{prev}$ and $Args_{prev} \subseteq \texttt{undec}(LabArg_{pref})$ by Definition 5. Thus, $Args$ violates Definition 4 because $\forall LabArg$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$ it holds that $\exists A \in \texttt{undec}(LabArg_{pref}) \setminus Args$ such that A is illegally labelled by $LabArg$. Let $Args' = Ar \setminus Args$. Then $\exists A \in Args'$ such that A is illegally labelled by $LabArg$, so $Args'$ satisfies the conditions of a preventing set (disregarding minimality). Since by definition of $Args'$ it holds that $Args \cap Args' = \emptyset$, $Args'$ is not a preventing set (by definition of $Args$). Thus $Args'$ is not a minimal set satisfying the conditions of a preventing set, i.e. $\exists Args_{prev} \in S_{prev}$ such that $Args_{prev} \subset Args'$. Then, by definition of $Args$, $Args \cap Args_{prev} \neq \emptyset$ and thus $Args \cap Args' \neq \emptyset$. Contradiction.

  Thus $Args$ is an enforcement set.

- Let $Args_{enf}$ be an enforcement set and assume that $Args_{enf} \notin S$. Then either $\exists Args_{prev} \in S_{prev}$ such that $Args_{enf} \cap Args_{prev} = \emptyset$ or there exists

a minimal set $Args \subset Args_{enf}$ satisfying that $\forall Args_{prev} \in S_{prev} : Args \cap Args_{prev} \neq \emptyset$.

- In the first case, since $Args_{prev}$ is a preventing set it holds that $\forall LabArg$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$, $\exists A \in Args_{prev}$ such that A is illegally labelled by $LabArg$. However, since $Args_{enf}$ is an enforcement set, by Lemma 39 in Appendix A there exists $LabArg'$ such that $\forall B \in Ar \setminus Args_{enf}$ it holds that B is legally labelled by $LabArg'$. Contradiction since $B \in Args_{prev}$.
- In the second case, $Args \in S$, so it follows from the first item of this proof that $Args$ is an enforcement set. Contradiction since $Args_{enf}$ is an enforcement set (and thus minimal).

Thus, $Args_{enf} \in S$. □

Proof (of Proposition 15). Assume there exists no odd-length cycle of arguments all labelled `undec` by $LabArg_{pref}$. Then $\mathcal{AF}_u = \mathcal{AF}{\downarrow_{\texttt{undec}(LabArg_{pref})}}$ comprises no odd-length cycle. By Corollary 36 in [10], $\mathcal{AF}_u$ has a stable labelling $LabArg_u$. We observe that, for all arguments $A \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ that are attacking some argument in $\texttt{undec}(LabArg_{pref})$, it holds that $A \in \texttt{out}(LabArg_{pref})$ and that, for all arguments $B \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ that are attacked by some argument in $\texttt{undec}(LabArg_{pref})$, it holds that $B \in \texttt{out}(LabArg_{pref})$.
Let $LabArg = LabArg_{pref}{\downarrow_{\texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})}} \cup LabArg_u$, so $\texttt{undec}(LabArg) = \emptyset$ and $LabArg_{pref} \sqsubseteq LabArg$. We show that $LabArg$ is a complete labelling of $\mathcal{AF}$:

- Let $A \in \texttt{in}(LabArg)$. If $A \in \texttt{in}(LabArg_{pref})$, then, by Lemma 38 in Appendix A, A is legally labelled by $LabArg$. If $A \in \texttt{in}(LabArg_u)$, then, for all attackers B of A such that $B \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$, $B \in \texttt{out}(LabArg_{pref})$ (by the above observation), and thus $B \in \texttt{out}(LabArg)$. Furthermore, for all attackers C of A such that $C \in \texttt{undec}(LabArg_{pref})$, $C \in \texttt{out}(LabArg_u)$ since $LabArg_u$ is a stable labelling of $\mathcal{AF}_u$, and thus $C \in \texttt{out}(LabArg)$. Thus, A is legally labelled `in` by $LabArg$.
- Let $A \in \texttt{out}(LabArg)$. If $A \in \texttt{out}(LabArg_{pref})$, then, by Lemma 38 in Appendix A, A is legally labelled by $LabArg$. If $A \in \texttt{out}(LabArg_u)$, then there exists an attacker B of A such that $B \in \texttt{undec}(LabArg_{pref})$ and $B \in \texttt{in}(LabArg_u)$ since $LabArg_u$ is a stable labelling of $\mathcal{AF}_u$, and thus $B \in \texttt{in}(LabArg)$. Thus, A is legally labelled `out` by $LabArg$.

Thus, $LabArg$ is a stable labelling of $\mathcal{AF}$, so $LabArg_{pref} = LabArg$ is a stable labelling. It follows that there exists an odd-length cycle of arguments all labelled `undec` by $LabArg_{pref}$.
If $LabArg_{pref}$ is a stable labelling, then $\texttt{undec}(LabArg_{pref}) = \emptyset$. Thus there exists no odd-length cycle of arguments all labelled `undec`. □

PROOF (OF PROPOSITION 16). Since $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})\setminus S}$ comprises no odd-length cycles, by Corollary 36 in [10] it has a stable labelling $LabArg_{stable}$. Let $LabArg' = LabArg_{stable} \cup LabArg_o$ be a labelling of $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})}$ where $LabArg_o$ is a labelling of arguments in $\mathcal{AF}{\downarrow}_S$ such that $\texttt{out}(LabArg_o) = S$, and let $LabArg = LabArg' \cup LabArg_{pref}{\downarrow}_{\texttt{in}(LabArg_{pref})\cup\texttt{out}(LabArg_{pref})}$. Clearly $LabArg$ is a labelling of $\mathcal{AF}$ such that $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$.

- Let $A \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$. By Lemma 38 in Appendix A, A is legally labelled by $LabArg$.
- Let $A \in \texttt{undec}(LabArg_{pref})\setminus S$ and $LabArg(A) = \texttt{in}$. Then, for all attackers B of A such that $B \in \texttt{undec}(LabArg_{pref}) \setminus S$, $LabArg_{stable}(B) = \texttt{out}$ and thus $LabArg(B) = \texttt{out}$. Furthermore, for all attackers C of A such that $C \in S$, $LabArg_o(C) = \texttt{out}$ and thus $LabArg(C) = \texttt{out}$. Additionally, for all attackers D of A such that $D \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$, $LabArg_{pref}(D) = \texttt{out}$ and thus $LabArg(D) = \texttt{out}$. Hence, A is legally labelled $\texttt{in}$ by $LabArg$.
- Let $A \in \texttt{undec}(LabArg_{pref}) \setminus S$ and $LabArg(A) = \texttt{out}$. Then there exists an attacker B of A such that $B \in \texttt{undec}(LabArg_{pref}) \setminus S$ and $LabArg_{stable}(B) = \texttt{in}$ and thus $LabArg(B) = \texttt{in}$. Hence, A is legally labelled $\texttt{out}$ by $LabArg$.

Thus, all $A \in Ar \setminus S$ are legally labelled by $LabArg$. Let $\mathcal{AF}^{\circledast}$ be a revision of $\mathcal{AF}$ w.r.t. S by $LabArg$ and $LabArg^{\circledast}$ a revision labelling of $\mathcal{AF}^{\circledast}$. By Definition 1, all $A \in S$ and all $B \in Ar^{\circledast} \setminus Ar$ are legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Furthermore, by Lemma 37 in Appendix A, all $A \in Ar \setminus S$ are legally labelled by $LabArg^{\circledast}$ in $\mathcal{AF}^{\circledast}$. Therefore, $LabArg^{\circledast}$ is a stable labelling of $\mathcal{AF}^{\circledast}$. □

PROOF (OF PROPOSITION 18). If $LabArg_{pref}$ is a stable labelling, then for all SCCs $Args$ it holds that $LabArg_{pref}{\downarrow}_{Args}$ is a stable labelling w.r.t. $(\mathcal{AF}{\downarrow}_{Args}, parents(Args), LabArg_{pref}{\downarrow}_{parents(Args)}, Att \cap (parents(Args) \times Args))$. This violates the condition in Definition 7, so there exists no responsible SCC.

If $LabArg_{pref}$ is not a stable labelling then since the attacks between SCCs are by definition unidirectional, there exists a sequence of SCCs $Args_1, \ldots, Args_n$ $(\forall i \neq k : Args_i \neq Args_k)$ such that if $Args_i$ is attacked by $Args_k$ $(i \neq k)$, then $k < i$. By Corollary 42 in Appendix A, $LabArg_{pref} = LabArg_1 \cup \ldots \cup LabArg_n$ where $LabArg_i$ is a labelling of $Args_i$, $LabArg_1$ is a complete labelling of $Args_1$, and for all $j \in \{2 \ldots n\}$ it holds that $LabArg_j$ is compatible with $LabArg_1 \cup \ldots \cup LabArg_{j-1}$. If $LabArg_1$ is not a stable labelling of $Args_1$, then $Args_1$ satisfies Definition 7, so there exists a responsible SCC w.r.t. $LabArg_{pref}$. Else, there exists $LabArg_i$ such that, for all $LabArg_j$ with $j < i$, it holds that $\texttt{undec}(LabArg_j) = \emptyset$ and $\texttt{undec}(LabArg_i) \neq \emptyset$. Since by the construction of our sequence of SCCs, for all $Args' \in parentSCCs(Args_i)$ it holds that $Args' = Args_j$ for some $j < i$, it follows that, for all these $Args'$, $LabArg_{pref}{\downarrow}_{Args'}$ is a stable labelling w.r.t. $(\mathcal{AF}{\downarrow}_{Args'}, parents(Args'), LabArg_{pref}{\downarrow}_{parents(Args')}, Att\cap$

$(parents(Args') \times Args'))$. Furthermore, since $\texttt{undec}(LabArg_i) \neq \emptyset$, it follows that there exists no stable labelling w.r.t. $(\mathcal{AF}{\downarrow}_{Args_i}, parents(Args_i), LabArg_{pref}{\downarrow}_{parents(Args_i)}, Att \cap (parents(Args_i) \times Args_i))$ that is more committed than $LabArg_i$. (If there was such a labelling, then $LabArg_{pref}$ would be a stable labelling.) □

PROOF (OF PROPOSITION 19). Let $ArgsIO = \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ and $ArgsU = \texttt{undec}(LabArg_{pref})$. We observe that since arguments labelled `undec` are not attacked by arguments labelled `in` by a complete labelling, $\forall B \in ArgsIO$ attacking some $A \in ArgsU$, it holds that $B \in \texttt{out}(LabArg_{pref}{\downarrow}_{ArgsIO})$. We first prove that $LabArg_{pref}{\downarrow}_{ArgsU}$ is a complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$. Since by Lemma 44 in Appendix A, $LabArg_{pref}{\downarrow}_{ArgsU}$ is a complete labelling w.r.t. $(\mathcal{AF}{\downarrow}_{ArgsU}, ArgsIO, LabArg_{pref}{\downarrow}_{ArgsIO}, Att \cap (ArgsIO \times ArgsU))$, it follows that $\forall A \in ArgsU$ and $\forall B \in ArgsU$ attacking A, $B \notin \texttt{in}(LabArg_{pref}{\downarrow}_{ArgsU})$, and $\exists C \in ArgsU$ attacking A such that $C \in \texttt{undec}(LabArg_{pref}{\downarrow}_{ArgsU})$ since by our above observation $\nexists D \in ArgsIO$ attacking A such that $D \in \texttt{undec}(LabArg_{pref}{\downarrow}_{ArgsIO})$. Thus, all $A \in ArgsU$ are legally labelled by $LabArg_{pref}{\downarrow}_{ArgsU}$, so $LabArg_{pref}{\downarrow}_{ArgsU}$ is a complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$. We now prove that there exists no other complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$. Assume there exists a complete labelling $LabArgU$ of $\mathcal{AF}{\downarrow}_{ArgsU}$ such that $\texttt{undec}(LabArgU) \neq ArgsU$. Clearly, $LabArg_{pref} \sqsubset LabArg_{pref}{\downarrow}_{ArgsIO} \cup LabArgU$.

By Lemma 46 in Appendix A, $LabArg_{pref}{\downarrow}_{ArgsIO}$ is compatible with $LabArgU$. Furthermore, $LabArgU$ is compatible with $LabArg_{pref}{\downarrow}_{ArgsIO}$:

- If $A \in \texttt{in}(LabArgU)$, then $\forall B \in ArgsU$ attacking A, $B \in \texttt{out}(LabArgU)$ since $LabArgU$ is a complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$. Furthermore $\forall B \in ArgsIO$ attacking A, $B \in \texttt{out}(LabArg_{pref}{\downarrow}_{ArgsIO})$ as previously noted.
- If $A \in \texttt{out}(LabArgU)$, then $\exists B \in ArgsU$ attacking A such that $B \in \texttt{in}(LabArgU)$ since $LabArgU$ is a complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$.
- If $A \in \texttt{undec}(LabArgU)$, then $\forall B \in ArgsU$ attacking A, $B \notin \texttt{in}(LabArgU)$, and $\exists B \in ArgsU$ attacking A such that $B \in \texttt{undec}(LabArgU)$ since $LabArgU$ is a complete labelling of $\mathcal{AF}{\downarrow}_{ArgsU}$. Furthermore, $\forall B \in ArgsIO$ attacking A, $B \notin \texttt{in}(LabArg_{pref}{\downarrow}_{ArgsIO})$ as previously noted.

It follows by Lemma 40 in Appendix A, that $LabArg_{pref}{\downarrow}_{ArgsIO} \cup LabArgU$ is a complete labelling of $\mathcal{AF}$. Contradiction, since $LabArg_{pref} \sqsubset LabArg_{pref}{\downarrow}_{ArgsIO} \cup LabArgU$ and $LabArg_{pref}$ is a preferred labelling. □

PROOF (OF PROPOSITION 20). Since every non-empty AF has an initial SCC, $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})}$ has an initial SCC if and only if $\texttt{undec}(LabArg_{pref}) \neq \emptyset$, i.e. if and only if $LabArg_{pref}$ is not a stable labelling. By Definition 8 this initial SCC is a SCUP w.r.t. $LabArg_{pref}$. □

PROOF (OF PROPOSITION 21). By Definition 7, there exists no stable labelling w.r.t. $(\mathcal{AF}{\downarrow}_{Args}, parents(Args), LabArg_{pref}{\downarrow}_{parents(Args)}, Att \cap (parents(Args) \times$

$Args$)) that is more or equally committed than $LabArg_{pref}{\downarrow}_{Args}$. Thus, $\texttt{undec}(LabArg_{pref}{\downarrow}_{Args}) \neq \emptyset$. Let $Args' = \texttt{undec}(LabArg_{pref}{\downarrow}_{Args})$. Since $Args$ is an SCC of $\mathcal{AF}$, $Args'$ is an SCC of $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})}$.
By Definition 7, for all $Args_p \in parentSCCs(Args)$ it holds that $LabArg_{pref}{\downarrow}_{Args_p}$ is a stable labelling w.r.t. $(\mathcal{AF}{\downarrow}_{Args_p}, parents(Args_p), LabArg_{pref}{\downarrow}_{parents(Args_p)}, Att \cap (parents(Args_p) \times Args_p))$. Thus, $\nexists A \in parents(Args), B \in Args$ such that A attacks B and $A \in \texttt{undec}(LabArg_{pref})$. Since $Args' \subseteq Args$ and since $Args \setminus Args' \subseteq \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$, it follows that $\nexists A \in parents(Args'), B \in Args'$ such that A attacks B and $A \in \texttt{undec}(LabArg_{pref})$. Thus, in $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})}$ it holds that $Args'$ is an SCC and $Args'$ is not attacked by any arguments not contained in $Args'$. Thus, $Args'$ is an initial SCC of $\mathcal{AF}{\downarrow}_{\texttt{undec}(LabArg_{pref})}$, so it is a SCUP w.r.t. $LabArg_{pref}$. □

Proof (of Lemma 22). Assume $\exists LabArg$ of $\mathcal{AF}$ with $LabArg_{pref} \sqsubset LabArg$ and $\texttt{undec}(LabArg) = \emptyset$ such that $\forall A \in Args$, A is legally labelled by $LabArg$ in $\mathcal{AF}$. Let $Args_1 = \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref}) \cup Args$, $Args_2 = Ar \setminus Args_1$, and $LabArg_1 = LabArg{\downarrow}_{Args_1}$. Since $Args$ is a SCUP, it holds that $\forall A \in Args$ and $\forall B$ attacking A, $B \in Args_1$. Thus, A being legally labelled by $LabArg$ only depends on $LabArg_1$. Let $LabArg_2$ be some labelling of $Args_2$. Then, $\forall A \in Args$, A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}$. Furthermore, clearly $LabArg_{pref} \sqsubset LabArg_1 \cup LabArg_2$. Then, by Lemma 38 in Appendix A, $\forall A \in \texttt{in}(LabArg_{pref}) \cup \texttt{out}(LabArg_{pref})$ it holds that A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}$. Thus, $\forall A \in Args_1$, A is legally labelled by $LabArg_1 \cup LabArg_2$ in $\mathcal{AF}$. Then, by Lemma 43 in Appendix A, $LabArg_1$ is compatible with $LabArg_2$ (for any labelling $LabArg_2$ of $Args_2$). Furthermore, by Lemma 45 in Appendix A, there exists a labelling $LabArg_2'$ that is compatible with $LabArg_1$. Then, by Lemma 40 in Appendix A, $LabArg_1 \cup LabArg_2'$ is a complete labelling of $\mathcal{AF}$. Contradiction since $LabArg_{pref} \sqsubset LabArg_1 \cup LabArg_2'$.
□

Proof (of Proposition 28). By Proposition 19 and the SCC recursiveness of complete labellings [16], $\mathcal{AF}{\downarrow}_{Args}$ has no stable labelling. Then, by Corollary 36 in [10], there exists an odd-length cycle in $Args$. □

References

[1] A. C. Kakas, P. Moraitis, Argumentation Based Decision Making for Autonomous Agents, in: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), 2003, pp. 883–890. doi:10.1145/860575.860717.

[2] L. Amgoud, H. Prade, Using Arguments for Making and Explaining Decisions, Artificial Intelligence 173 (3-4) (2009) 413–436. doi:10.1016/j.artint.2008.11.006.

[3] W. Ouerdane, Y. Dimopoulos, K. Liapis, P. Moraitis, Towards Automating Decision Aiding Through Argumentation, Journal of Multi-Criteria Decision Analysis 18 (5-6) (2011) 289–309. doi:10.1002/mcda.486.

[4] T. J. M. Bench-Capon, K. Atkinson, P. McBurney, Using Argumentation to Model Agent Decision Making in Economic Experiments, Autonomous Agents and Multi-Agent Systems 25 (1) (2012) 183–208. doi:10.1007/s10458-011-9173-6.

[5] J. Müller, A. Hunter, An Argumentation-Based Approach for Decision Making, in: Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'12), 2012, pp. 564–571. doi:10.1109/ICTAI.2012.82.

[6] J.-R. Bourguet, R. Thomopoulos, M.-L. Mugnier, J. Abécassis, An Artificial Intelligence-Based Approach to Deal with Argumentation Applied to Food Quality in a Public Health Policy, Expert Systems with Applications 40 (11) (2013) 4539–4546. doi:10.1016/j.eswa.2013.01.059.

[7] X. Fan, F. Toni, A. Mocanu, M. Williams, Dialogical Two-Agent Decision Making with Assumption-Based Argumentation, in: Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'14), 2014, pp. 533–540.

[8] L. Carstens, X. Fan, Y. Gao, F. Toni, An Overview of Argumentation Frameworks for Decision Support, in: Revised Selected Papers of the 4th International Workshop on Graph Structures for Knowledge Representation and Reasoning (GKR'15), 2015, pp. 32–49. doi:10.1007/978-3-319-28702-7_3.

[9] E. Ferretti, L. H. Tamargo, A. J. García, M. L. Errecalde, G. R. Simari, An Approach to Decision Making based on Dynamic Argumentation Systems, Artificial Intelligence 242 (2017) 107–131. doi:10.1016/j.artint.2016.10.004.

[10] 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) (1995) 321–357. doi:10.1016/0004-3702(94)00041-X.

[11] M. Caminada, D. M. Gabbay, A Logical Account of Formal Argumentation, Studia Logica 93 (2-3) (2009) 109–145. doi:10.1007/s11225-009-9218-x.

[12] P. Baroni, M. Caminada, M. Giacomin, An Introduction to Argumentation Semantics, The Knowledge Engineering Review 26 (04) (2011) 365–410. doi:10.1017/S0269888911000166.

[13] M. Caminada, Semi-Stable Semantics, in: Proceedings of the 1st International Conference on Computational Models of Argument (COMMA'06), 2006, pp. 121–130.

[14] A. Hunter, M. Thimm, Probabilistic Reasoning with Abstract Argumentation Frameworks, Journal of Artificial Intelligence Research 59 (2017) 565–611. doi:10.1613/jair.5393.

[15] A. Hunter, M. Williams, Aggregation of Clinical Evidence Using Argumentation: A Tutorial Introduction, in: A. Hommersom, P. J. F. Lucas (Eds.), Foundations of Biomedical Knowledge Representation - Methods and Applications, Vol. 9521 of Lecture Notes in Computer Science, Springer, 2015, pp. 317–337. doi:10.1007/978-3-319-28007-3_20.

[16] P. Baroni, M. Giacomin, G. Guida, SCC-Recursiveness: A General Schema for Argumentation Semantics, Artificial Intelligence 168 (1-2) (2005) 162–210. doi:10.1016/j.artint.2005.05.006.

[17] R. Baumann, G. Brewka, Expanding Argumentation Frameworks: Enforcing and Monotonicity Results, in: Proceedings of the 3rd International Conference on Computational Models of Argument (COMMA'10), 2010, pp. 75–86. doi:10.3233/978-1-60750-619-5-75.

[18] P. Baroni, G. Boella, F. Cerutti, M. Giacomin, L. W. N. van der Torre, S. Villata, On the Input/Output Behavior of Argumentation Frameworks, Artificial Intelligence 217 (2014) 144–197. doi:10.1016/j.artint.2014.08.004.

[19] R. Baumann, What Does it Take to Enforce an Argument? Minimal Change in Abstract Argumentation, in: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12), 2012, pp. 127–132. doi:10.3233/978-1-61499-098-7-127.

[20] S. Coste-Marquis, S. Konieczny, J.-G. Mailly, P. Marquis, On the Revision of Argumentation Systems: Minimal Change of Arguments Statuses, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), 2014, pp. 52–61.

[21] P. Baroni, M. Giacomin, 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'03), 2003, pp. 440–451. doi:10.1007/978-3-540-45062-7_36.

[22] R. Baumann, H. Strass, On the Maximal and Average Numbers of Stable Extensions, in: 2nd International Workshop on Theory and Applications of Formal Argumentation (TAFA'13), 2013, pp. 111–126. doi:10.1007/978-3-642-54373-9_8.

[23] P. E. Dunne, T. J. M. Bench-Capon, Coherence in Finite Argument Systems, Artificial Intelligence 141 (1-2) (2002) 187–203. doi:10.1016/S0004-3702(02)00261-8.

[24] F. Nouioua, E. Würbel, Removed Set-Based Revision of Abstract Argumentation Frameworks, in: Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'14), 2014, pp. 784–791. doi:10.1109/ICTAI.2014.121.

[25] P. Baroni, M. Giacomin, B. Liao, I don't care, I don't know ... I know too much! On Incompleteness and Undecidedness in Abstract Argumentation, in: T. Eiter, H. Strass, M. Truszczyński, S. Woltran (Eds.), Advances in Knowledge Representation, Logic Programming, and Abstract Argumentation - Essays Dedicated to Gerhard Brewka on the Occasion of His 60th Birthday, Springer International Publishing, 2015, pp. 265–280. doi:10.1007/978-3-319-14726-0_18.

[26] M. Caminada, G. Pigozzi, On Judgment Aggregation in Abstract Argumentation, Autonomous Agents and Multi-Agent Systems 22 (1) (2011) 64–102.

[27] R. Reiter, A Theory of Diagnosis from First Principles, Artificial Intelligence 32 (1) (1987) 57 – 95. doi:10.1016/0004-3702(87)90062-2.

[28] A. Ignatiev, A. Previti, M. Liffiton, J. Marques-Silva, Smallest MUS Extraction with Minimal Hitting Set Dualization, in: Proceedings of the 21 International Conference on Principles and Practice of Constraint Programming (CP'15), 2015, pp. 173–182. doi:10.1007/978-3-319-23219-5_13.

[29] P. Baroni, D. M. Gabbay, M. Giacomin, Introduction to the Special Issue on Loops in Argumentation, Journal of Logic and Computation 26 (4) (2016) 1051–1053. doi:10.1093/logcom/exu012.

[30] W. Dvořák, S. A. Gaggl, Stage Semantics and the SCC-recursive Schema for Argumentation Semantics, Journal of Logic and Computation 26 (4) (2016) 1149–1202. doi:10.1093/logcom/exu006.

[31] O. Arieli, On the Acceptance of Loops in Argumentation Frameworks, Journal of Logic and Computation 26 (4) (2016) 1203–1234. doi:10.1093/logcom/exu009.

[32] D. M. Gabbay, The Handling of Loops in Argumentation Networks, Journal of Logic and Computation 26 (4) (2016) 1065–1147. doi:10.1093/logcom/exu007.

[33] G. A. Bodanza, F. A. Tohmé, Two Approaches to the Problems of Self-Attacking Arguments and General Odd-Length Cycles of Attack, Journal of Applied Logic 7 (4) (2009) 403–420. doi:10.1016/j.jal.2007.06.012.

[34] T. J. M. Bench-Capon, Dilemmas and Paradoxes: Cycles in Argumentation Frameworks, Journal of Logic and Computation 26 (4) (2016) 1055–1064. doi:10.1093/logcom/exu011.

[35] R. Baumann, S. Woltran, The Role of Self-Attacking Arguments in Characterizations of Equivalence Notions, Journal of Logic and Computation 26 (4) (2016) 1293–1313. doi:10.1093/logcom/exu010.

[36] B. Liao, Toward Incremental Computation of Argumentation Semantics: A Decomposition-Based Approach, Annals of Mathematics and Artificial Intelligence 67 (3-4) (2013) 319–358. doi:10.1007/s10472-013-9364-8.

[37] R. Baumann, Splitting an Argumentation Framework, in: Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11), 2011, pp. 40–53. doi:10.1007/978-3-642-20895-9_6.

[38] R. Baumann, G. Brewka, W. Dvořák, S. Woltran, Parameterized Splitting: A Simple Modification-Based Approach, in: E. Erdem, J. Lee, Y. Lierler, D. Pearce (Eds.), Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz, Springer Berlin Heidelberg, 2012, pp. 57–71. doi:10.1007/978-3-642-30743-0_5.

[39] T. Rienstra, A. Perotti, S. Villata, D. M. Gabbay, L. W. N. van der Torre, Multi-Sorted Argumentation, in: Revised Selected Papers of the 1st International Workshop on Theorie and Applications of Formal Argumentation (TAFA'11), 2011, pp. 215–231. doi:10.1007/978-3-642-29184-5_14.

[40] C. Cayrol, F. D. de Saint-Cyr, M.-C. Lagasquie-Schiex, Change in Abstract Argumentation Frameworks: Adding an Argument, Journal of Artificial Intelligence Research 38 (2010) 49–84. doi:10.1613/jair.2965.

[41] B. Liao, L. Jin, R. C. Koons, Dynamics of Argumentation Systems: A Division-Based Method, Artificial Intelligence 175 (11) (2011) 1790–1814. doi:10.1016/j.artint.2011.03.006.

[42] R. Booth, S. Kaci, T. Rienstra, L. W. N. van der Torre, A Logical Theory about Dynamics in Abstract Argumentation, in: Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM'13), 2013, pp. 148–161. doi:10.1007/978-3-642-40381-1_12.

[43] S. Coste-Marquis, S. Konieczny, J.-G. Mailly, P. Marquis, Extension Enforcement in Abstract Argumentation as an Optimization Problem, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15), 2015, pp. 2876–2882.

[44] D. Kontarinis, E. Bonzon, N. Maudet, A. Perotti, L. W. N. van der Torre, S. Villata, Rewriting Rules for the Computation of Goal-Oriented Changes in an Argumentation System, in: Proceedings of the 14th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA'13), 2013, pp. 51–68. doi:10.1007/978-3-642-40624-9_4.

[45] R. Booth, D. M. Gabbay, S. Kaci, T. Rienstra, L. W. N. van der Torre, Abduction and Dialogical Proof in Argumentation and Logic Programming, in: Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14), 2014, pp. 117–122. doi:10.3233/978-1-61499-419-0-117.

[46] S. Coste-Marquis, S. Konieczny, J.-G. Mailly, P. Marquis, A Translation-Based Approach for Revision of Argumentation Frameworks, in: Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA'14), 2014, pp. 397–411. doi:10.1007/978-3-319-11558-0_28.

[47] P. Bisquert, C. Cayrol, F. D. de Saint-Cyr, M.-C. Lagasquie-Schiex, Enforcement in Argumentation Is a Kind of Update, in: Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM'13), 2013, pp. 30–43. doi:10.1007/978-3-642-40381-1_3.

[48] S. Doutre, A. Herzig, L. Perrussel, A Dynamic Logic Framework for Abstract Argumentation, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), 2014, pp. 62–71.

[49] G. Boella, D. M. Gabbay, A. Perotti, L. W. N. van der Torre, S. Villata, Conditional Labelling for Abstract Argumentation, in: Revised Selected Papers of the 1st International Workshop on Theorie and Applications of Formal Argumentation (TAFA'11), 2011, pp. 232–248. doi:10.1007/978-3-642-29184-5_15.

[50] G. Charwat, W. Dvořák, S. A. Gaggl, J. P. Wallner, S. Woltran, Methods for Solving Reasoning Problems in Abstract Argumentation - A Survey, Artificial intelligence 220 (2015) 28–63. doi:10.1016/j.artint.2014.11.008.

[51] F. Cerutti, M. Giacomin, M. Vallati, M. Zanella, An SCC Recursive Meta-Algorithm for Computing Preferred Labellings in Abstract Argumentation, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), 2014, pp. 42–51.

[52] Y. Wu, M. Caminada, D. M. Gabbay, Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming, Studia Logica 93 (2-3) (2009) 383–403. doi:10.1007/s11225-009-9210-5.

[53] M. Caminada, S. Sá, J. Alcântara, W. Dvořák, On the Equivalence between Logic Programming Semantics and Argumentation Semantics, International Journal of Approximate Reasoning 58 (2015) 87–111. doi:10.1016/j.ijar.2014.12.004.

[54] M. Brain, M. De Vos, Debugging Logic Programs under the Answer Set Semantics, in: Proceedings of the 3rd Workshop on Answer Set Programming, Advances in Theory and Implementation (ASP'05), 2005.

[55] T. Syrjänen, Debugging Inconsistent Answer Set Programs, in: Proceedings of the 11th International Workshop on Non-Monotonic Reasoning (NMR'06), 2006, pp. 77–84.

[56] M. Caminada, C. Sakama, On the Existence of Answer Sets in Normal Extended Logic Programs, in: Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06), 2006, pp. 743–744.

[57] M. Gebser, J. Pührer, T. Schaub, H. Tompits, A Meta-Programming Technique for Debugging Answer-Set Programs, in: Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08), 2008, pp. 448–453.

[58] J. Oetsch, J. Pührer, H. Tompits, Catching the Ouroboros: On Debugging Non-Ground Answer-Set Programs, Theory and Practice of Logic Programming 10 (4-6) (2010) 513–529. doi:10.1017/S1471068410000256.

[59] J. Oetsch, J. Pührer, H. Tompits, Stepping through an Answer-Set Program, in: Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11), 2011, pp. 134–147. doi:10.1007/978-3-642-20895-9_13.

[60] C. Schulz, K. Satoh, F. Toni, Characterising and Explaining Inconsistency in Logic Programs, in: Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), 2015, pp. 467–479. doi:10.1007/978-3-319-23264-5_39.

[61] M. Ulbricht, M. Thimm, G. Brewka, Measuring Inconsistency in Answer Set Programs, in: Proceedings of the 15th European Conference on Logics in Artificial Intelligence (JELIA'16), 2016, pp. 577–583. doi:10.1007/978-3-319-48758-8_42.

[62] C. Cayrol, M.-C. Lagasquie-Schiex, Graduality in Argumentation, Journal of Artificial Intelligence Research 23 (2005) 245–297. doi:10.1613/jair.1411.

[63] P.-A. Matt, F. Toni, A Game-Theoretic Measure of Argument Strength for Abstract Argumentation, in: Proceedings of the 11th European Conference on Logics in Artificial Intelligence (JELIA'08), 2008, pp. 285–297. doi:10.1007/978-3-540-87803-2_24.

[64] M. Thimm, A Probabilistic Semantics for Abstract Argumentation, in: Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12), 2012, pp. 750–755. doi:10.3233/978-1-61499-098-7-750.

[65] L. Amgoud, J. Ben-Naim, Ranking-Based Semantics for Argumentation Frameworks, in: Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM'13), 2013, pp. 134–147. doi:10.1007/978-3-642-40381-1_11.

[66] M. Thimm, G. Kern-Isberner, On Controversiality of Arguments and Stratified Labelings, in: Proceedings of the 5th International Conference on Computational Models of Argument (COMMA'14), 2014, pp. 413–420. doi:10.3233/978-1-61499-436-7-413.

[67] D. Grossi, S. Modgil, On the Graded Acceptability of Arguments, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15), 2015, pp. 868–874.

[68] L. Amgoud, J. Ben-Naim, D. Doder, S. Vesic, Ranking Arguments With Compensation-Based Semantics, in: Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR'16), 2016, pp. 12–21.

[69] D. M. Gabbay, O. Rodrigues, Degrees of "in", "out" and "undecided" in Argumentation Networks, in: Proceedings of the 6th International Conference on Computational Models of Argument (COMMA'16), 2016, pp. 319–326. doi:10.3233/978-1-61499-686-6-319.