Complete Assumption Labellings

Claudia SCHULZ, Francesca TONI
Department of Computing, Imperial College London, UK

Abstract. Recently, argument labellings have been proposed as a new (equivalent) way to express the extension semantics of Abstract Argumentation (AA) frameworks. Here, we introduce a labelling approach for the complete semantics in Assumption-Based Argumentation (ABA), where labels are assigned to assumptions rather than whole arguments. We prove that the complete assumption labelling corresponds to the complete extension semantics in ABA, as well as to the complete extension semantics and the complete argument labelling in AA.

Keywords. Assumption-Based Argumentation, Complete Semantics, Labelling Semantics

1. Introduction

Abstract Argumentation (AA) [1] studies conflicts between abstract entities called arguments, and provides semantics for deciding which sets of arguments may be accepted. Different semantics have been defined [1,2], yielding different sets of accepted arguments, referred to as *extensions*. Another (equivalent) way of defining argumentation semantics is by assigning *labels* to arguments [3], identifying not only accepted arguments, but also rejected and neutral ones.

In contrast to AA, where a set of arguments and a set of attacks between them is given, *Assumption-Based Argumentation* (ABA) [4,5] provides a mechanism for constructing arguments from given rules and assumptions. Moreover, attacks between arguments are not predefined as in AA, but arise based on the structure of arguments and a notion of contrary of assumptions. Another difference is that the semantics of an ABA framework can be defined in terms of sets of accepted assumptions as well as sets of accepted arguments [6]. Since ABA is an instance of AA [6], an ABA framework can be mapped onto a corresponding AA framework, such that the extensions of the ABA framework correspond to the extensions of the AA framework (with the exception of the semi-stable extension semantics [7]).

Inspired by the complete argument labelling for AA, which coincides with the complete AA extension semantics, we introduce a way to express the complete ABA extension semantics in terms of a labelling, where labels are assigned to single assumptions as opposed to the AA approach to label whole arguments. We show that this complete assumption labelling corresponds to the complete extension semantics in ABA as well as to the complete extension semantics and the complete argument labelling of the corresponding AA framework. This new assumption labelling approach has the advantage that rejected (OUT) assumptions and neutral assumptions which are neither accepted nor rejected (UNDEC) are distinguished. This distinction of non-accepted assumptions is important in applications, e.g. when using ABA for decision making, where it is necessary to know whether an assumption is rejected for sure or whether there is not enough evidence to definitely accept or reject it.

2. Background

An *Abstract Argumentation (AA) framework* [1] is a pair $\langle Ar, Att\rangle$, where Ar is a set of arguments and $Att \subseteq Ar \times Ar$ is a binary attack relation between arguments. A pair $(A, B) \in Att$ expresses that argument A attacks argument B. A set of arguments $Args \subseteq Ar$ attacks an argument $B \in Ar$ iff there is $A \in Args$ such that A attacks B. $Args^+ = \{A \in Ar \mid Args \textit{ attacks } A\}$ denotes the set of all arguments attacked by $Args$ [2].
Let $Args \subseteq Ar$ be a set of arguments.

- $Args$ *defends* $A \in Ar$ iff $Args$ attacks every $B \in Ar$ attacking A.
- $Args$ is a *complete argument extension* of $\langle Ar, Att\rangle$ iff $Args$ consists of all arguments it defends and $Args$ does not attack any $A \in Args$.

An equivalent way of expressing the extension semantics of an AA framework is in terms of argument labellings [8,3].

An *argument labelling* of $\langle Ar, Att\rangle$ is a total function $LabArg : Ar \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$. The set of all arguments labelled $\texttt{in}$ by $LabArg$ is denoted $\texttt{in}(LabArg) = \{A \in Ar \mid LabArg(A) = \texttt{in}\}$. The sets $\texttt{out}(LabArg)$ and $\texttt{undec}(LabArg)$ consist of all arguments labelled $\texttt{out}$ and $\texttt{undec}$, respectively.

An argument labelling $LabArg$ is a *complete argument labelling* of $\langle Ar, Att\rangle$ [3] iff for each argument $A \in Ar$ it holds that:

- if $LabArg(A) = \texttt{in}$ then for each $B \in Ar$ attacking A, $LabArg(B) = \texttt{out}$;
- if $LabArg(A) = \texttt{out}$ then there exists some $B \in Ar$ attacking A such that $LabArg(B) = \texttt{in}$;
- if $LabArg(A) = \texttt{undec}$ then there exists some $B \in Ar$ attacking A such that $LabArg(B) = \texttt{undec}$ and there exists no $C \in Ar$ attacking A such that $LabArg(C) = \texttt{in}$.

Complete argument extensions coincide with sets of arguments labelled $\texttt{in}$ [3].

An *Assumption-Based Argumentation (ABA) framework* [4,5] is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ }\rangle$, where

- $(\mathcal{L}, \mathcal{R})$ is a deductive system, with $\mathcal{L}$ a language and $\mathcal{R}$ a set of inference rules of the form $s_0 \leftarrow s_1, \ldots, s_n$ $(n \geq 0)$ where $s_0, \ldots, s_n \in \mathcal{L}$;
- $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set of *assumptions*;
- $\bar{\ }$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$ defining the *contrary* of assumptions, where $\overline{\alpha}$ denotes the contrary of $\alpha \in \mathcal{A}$.

We will here focus on *flat* ABA frameworks [4], where assumptions only occur on the right of the arrow in inference rules.

An *argument* $AP \vdash s$ for *conclusion* $s \in \mathcal{L}$ supported by *assumption-premises* $AP \subseteq \mathcal{A}$ is a finite tree, where every node holds a sentence in $\mathcal{L}$ or the sentence τ (where $\tau \notin \mathcal{L}$ stands for "true"), such that

- the root node holds s;
- for every node N
  - $*$ if N is a leaf then N holds either an assumption or τ;
  - $*$ if N is not a leaf and N holds the sentence t_0, then there is an inference rule $t_0 \leftarrow t_1, \ldots, t_m \in \mathcal{R}$ and either $m = 0$ and the only child node of N holds τ or $m > 0$ and N has m children holding $t_1, \ldots, t_m$;
- AP is the set of all assumptions held by leaf nodes.

We sometimes name arguments with capital letters, e.g. $A\!:\!AP \vdash s$ is an argument with name A. With an abuse of notation, the name of an argument is also used to refer to the whole argument.

Let $Asms, Asms_1 \subseteq \mathcal{A}$ be sets of assumptions and let $\alpha \in \mathcal{A}$.

- $Asms$ *attacks* α iff there exists an argument $AP \vdash \overline{\alpha}$ such that $AP \subseteq Asms$. Equivalently, we say α is attacked by $Asms$.
- $Asms$ *attacks* $Asms_1$ iff $Asms$ attacks some $\alpha \in Asms_1$.
- $Asms^+ = \{\alpha \in \mathcal{A} \mid Asms \textit{ attacks } \alpha\}$ [7].
- $Asms$ *defends* α iff $Asms$ attacks all sets of assumptions attacking α.
- $Asms$ is a *complete assumption extension* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ iff $Asms$ consists of all assumptions it defends and $Asms$ does not attack itself.

An ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ can be mapped onto a *corresponding AA framework* $\langle Ar_{ABA}, Att_{ABA} \rangle$ [6], where

- Ar_{ABA} is the set of all constructible arguments $AP \vdash s$ in $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$;
- $(AP_1 \vdash s_1,\ AP_2 \vdash s_2) \in Att_{ABA}$ iff s_1 is the contrary of some $\alpha \in AP_2$.

Given a complete assumption extension $Asms$ of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$, the set of all arguments supported by any subset of $Asms$ forms a complete argument extension of $\langle Ar_{ABA}, Att_{ABA} \rangle$. Conversely, given a complete argument extension $Args$ of $\langle Ar_{ABA}, Att_{ABA} \rangle$, the union of all assumptions supporting arguments in $Args$ is a complete assumption extension of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ [7].

3. Labelling Assumptions

Inspired by argument labellings in AA, we introduce a labelling for ABA, which assigns labels to single assumptions rather than whole arguments. In the remainder, and if clear from the context, we assume as given an ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$.

Definition 1. An *assumption labelling* is a total function
$LabAsm : \mathcal{A} \rightarrow \{\text{IN}, \text{OUT}, \text{UNDEC}\}$.

$\text{IN}(LabAsm) = \{\alpha \in \mathcal{A} \mid LabAsm(\alpha) = \text{IN}\}$ consists of all assumptions labelled IN. $\text{OUT}(LabAsm)$ and $\text{UNDEC}(LabAsm)$ are the sets of all assumptions labelled OUT and UNDEC, respectively.

Definition 2. Let $LabAsm$ be an assumption labelling. $LabAsm$ is a *complete assumption labelling* iff for each assumption $\alpha \in \mathcal{A}$ it holds that:

- if $LabAsm(\alpha)$ = IN then each set of assumptions attacking α contains some β such that $LabAsm(\beta) =$ OUT;
- if $LabAsm(\alpha)$ = OUT then there exists a set of assumptions AP attacking α such that $AP \subseteq$ IN$(LabAsm)$;
- if $LabAsm(\alpha) =$ UNDEC then each set of assumptions attacking α contains some β such that $LabAsm(\beta) \neq$ IN, and there exists a set of assumptions AP attacking α such that $AP \cap$ OUT$(LabAsm) = \emptyset$.

Example 1. Consider the following ABA framework, which we call ABA_1:

- $\mathcal{L} = \{a, b, c, \alpha, \beta, \gamma\}$
- $\mathcal{R} = \{a \leftarrow \alpha\ ;\ a \leftarrow \beta\ ;\ c \leftarrow \beta,\ ;\ b \leftarrow \gamma\}$
- $\mathcal{A} = \{\alpha, \beta, \gamma\}$
- $\overline{\alpha} = a\ ;\ \overline{\beta} = b\ ;\ \overline{\gamma} = c$

ABA_1 has three complete assumption labellings:

- IN$(LabAsm_1) = \emptyset$, OUT$(LabAsm_1) = \emptyset$, UNDEC$(LabAsm_1) = \{\alpha, \beta, \gamma\}$
- IN$(LabAsm_2) = \{\gamma\}$, OUT$(LabAsm_2) = \{\beta\}$, UNDEC$(LabAsm_2) = \{\alpha\}$
- IN$(LabAsm_3) = \{\beta\}$, OUT$(LabAsm_3) = \{\alpha, \gamma\}$, UNDEC$(LabAsm_3) = \emptyset$

Lemma 1. *Let $LabAsm$ be an assumption labelling. $LabAsm$ is a complete assumption labelling iff for each assumption $\alpha \in \mathcal{A}$ it holds that:*

- *if each set of assumptions attacking α contains some β such that $LabAsm(\beta)$ $=$ OUT, then $LabAsm(\alpha) =$ IN;*
- *if there exists a set of assumptions AP attacking α such that $AP \subseteq$ IN$(LabAsm)$, then $LabAsm(\alpha) =$ OUT;*
- *if each set of assumptions attacking α contains some β such that $LabAsm(\beta)$ $\neq$ IN, and there exists a set of assumptions AP attacking α such that $AP \cap$ OUT$(LabAsm) = \emptyset$, then $LabAsm(\alpha) =$ UNDEC.*

Proof. Omitted due to space limitations, but it is straightforward to prove that in each case α cannot have a different label. □

Similarly to AA, the notions of complete assumption labelling and complete assumption extension coincide.

Theorem 2. *Let $LabAsm$ be an assumption labelling. $LabAsm$ is a complete assumption labelling iff $Asms =$ IN$(LabAsm)$ is a complete assumption extension with $Asms^+ =$ OUT$(LabAsm)$ and $\mathcal{A} \setminus (Asms \cup Asms^+) =$ UNDEC$(LabAsm)$.*

Proof. We prove both directions.

1. From left to right:
   We first prove that IN$(LabAsm)$ is a complete assumption extension.

   - IN$(LabAsm)$ does not attack itself: Assume IN$(LabAsm)$ attacks itself. Then, there is a set $AP \subseteq$ IN$(LabAsm)$ attacking some $\alpha \in$ IN$(LabAsm)$. By Definition 2, each set attacking α contains some β labelled OUT. Hence, AP contains some β labelled OUT. Contradiction.

- $\textsc{in}(LabAsm)$ contains only assumptions defended by $\textsc{in}(LabAsm)$: Let $\alpha \in \textsc{in}(LabAsm)$. Then by Definition 2, each set attacking α contains some β labelled $\textsc{out}$. For each such β there exists a set $AP \subseteq \textsc{in}(LabAsm)$ attacking β. Hence, $\textsc{in}(LabAsm)$ defends α.
- All assumptions defended by $\textsc{in}(LabAsm)$ are in $\textsc{in}(LabAsm)$: Let α be defended by $\textsc{in}(LabAsm)$, i.e. for each AP_1 attacking α there exists some $AP_2 \subseteq \textsc{in}(LabAsm)$ which attacks AP_1. So in every AP_1 there is some β which is attacked by the respective $AP_2 \subseteq \textsc{in}(LabAsm)$. By Lemma 1, $LabAsm(\beta) = \textsc{out}$, so each AP_1 contains some β labelled $\textsc{out}$. By Lemma 1, $LabAsm(\alpha) = \textsc{in}$.

$Asms^+ = \{\alpha \in \mathcal{A} \mid Asms \textit{ attacks } \alpha\} = \{\alpha \in \mathcal{A} \mid \textsc{in}(LabAsm) \textit{ attacks } \alpha\}$
$= \{\alpha \in \mathcal{A} \mid \alpha \in \textsc{out}(LabAsm)\}$ (by Lemma 1) $= \textsc{out}(LabAsm)$
$\mathcal{A} \setminus (Asms \cup Asms^+) = \{\alpha \in \mathcal{A} \mid \alpha \notin \textsc{in}(LabAsm), \alpha \notin \textsc{out}(LabAsm)\}$
$= \{\alpha \in \mathcal{A} \mid \alpha \in \textsc{undec}(LabAsm)\} = \textsc{undec}(LabAsm)$

2. From right to left: We prove that $LabAsm$ satisfies Definition 2.

- Let $LabAsm(\alpha) = \textsc{in}$. Then $\alpha \in Asms$, so for all sets AP_1 attacking α there exists some $AP_2 \subseteq Asms$, i.e. $AP_2 \subseteq \textsc{in}(LabAsm)$, which attacks some $\beta \in AP_1$. By Lemma 1, $LabAsm(\beta) = \textsc{out}$, so each AP_1 attacking α contains some β labelled $\textsc{out}$.
- Let $LabAsm(\alpha) = \textsc{out}$. Then $\alpha \in Asms^+$, so there is $AP \subseteq Asms$ attacking α. Thus, there is $AP \subseteq \textsc{in}(LabAsm)$ attacking α.
- Let $LabAsm(\alpha) = \textsc{undec}$. Then $\alpha \notin Asms$ and $\alpha \notin Asms^+$, so α is not attacked and not defended by $Asms$. Thus, α is not attacked by any $AP \subseteq \textsc{in}(LabAsm)$, so each set attacking α contains some β such that $LabAsm(\beta) \neq \textsc{in}$. Furthermore, there exists a set AP_1 attacking α which is not attacked by any $AP_2 \subseteq Asms$, i.e. by $AP_2 \subseteq \textsc{in}(LabAsm)$. Hence, AP_1 does not contain a γ labelled $\textsc{out}$, so $AP_1 \cap \textsc{out}(LabAsm) = \emptyset$.

$\square$

Example 2. ABA_1 has three complete assumption extensions: $Asms_1 = \emptyset$ with $Asms_1^+ = \emptyset$, $Asms_2 = \{\gamma\}$ with $Asms_2^+ = \{\beta\}$, $Asms_3 = \{\beta\}$ with $Asms_3^+ = \{\alpha, \gamma\}$. These complete assumption extensions correspond to the three complete assumption labellings of ABA_1 as stated in Theorem 2 ($Asms_1 - LabAsm_1$, $Asms_2 - LabAsm_2$, $Asms_3 - LabAsm_3$).

4. Relationship with AA

We now examine the relationship between complete assumption labellings in ABA and complete argument labellings in AA. In the remainder, and if clear from the context, we assume as given an ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ and its corresponding AA framework $\langle Ar_{ABA}, Att_{ABA} \rangle$. We first investigate the relationship between sets of assumptions in ABA and sets of arguments in AA in general.

Lemma 3. *Let $Asms \subseteq \mathcal{A}$ and let $Args = \{AP \vdash s \in Ar_{ABA} \mid AP \subseteq Asms\}$ be the set of all arguments in Ar_{ABA} supported by any subset of $Asms$. Then*

- $Args^+ = \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \alpha \in Asms^+\}$;

- $Ar_{ABA} \setminus (Args \cup Args^{+})$
  $= \{AP \vdash s \in Ar_{ABA} \mid AP \nsubseteq Asms, \nexists \alpha \in AP \textit{ s.t. } \alpha \in Asms^{+}\}$.

Proof. We prove both statements:

- $Args^{+} = \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \alpha \in Asms^{+}\}$
  $= \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } Asms \textit{ attacks } \alpha\}$
  $= \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \exists AP_1 \vdash \overline{\alpha} \textit{ and } AP_1 \subseteq Asms\}$
  $= \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \exists AP_1 \vdash \overline{\alpha} \in Args\}$
  $= \{AP \vdash s \in Ar_{ABA} \mid Args \textit{ attacks } AP \vdash s\}$
  $= \{A \in Ar_{ABA} \mid Args \textit{ attacks } A\}$
- $Ar_{ABA} \setminus (Args \cup Args^{+})$
  $= \{AP \vdash s \in Ar_{ABA} \mid AP \nsubseteq Asms, \nexists \alpha \in AP \textit{ s.t. } \alpha \in Asms^{+}\}$
  $= \{AP \vdash s \in Ar_{ABA} \mid AP \vdash s \notin Args, AP \vdash s \notin Args^{+}\}$
  $= \{A \in Ar_{ABA} \mid A \notin Args, A \notin Args^{+}\}$ □

Due to Theorem 2 and the correspondence between complete assumption extensions and complete argument extensions [7], it is straightforward that complete assumption labellings and complete argument labellings coincide. Theorem 4 below characterises the complete argument labelling corresponding to a given complete assumption labelling. Conversely, Theorem 5 identifies the complete assumption labelling corresponding to a given complete argument labelling.

Theorem 4. *Let $LabAsm$ be an assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$. $LabAsm$ is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ iff $LabArg$ with*

- $\texttt{in}(LabArg) = \{AP \vdash s \in Ar_{ABA} \mid AP \subseteq \textsc{in}(LabAsm)\}$,
- $\texttt{out}(LabArg) = \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \alpha \in \textsc{out}(LabAsm)\}$,
- $\texttt{undec}(LabArg) = \{AP \vdash s \in Ar_{ABA} \mid \exists \alpha \in AP \textit{ s.t. } \alpha \in \textsc{undec}(LabAsm), AP \cap \textsc{out}(LabAsm) = \emptyset\}$

is a complete argument labelling of $\langle Ar_{ABA}, Att_{ABA} \rangle$.

Proof. We prove both directions of the statement.

1. From left to right:
   By Theorem 2: $Asms = \textsc{in}(LabAsm)$ is a complete assumption extension of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$, with $Asms^{+} = \textsc{out}(LabAsm)$ and $\mathcal{A} \setminus (Asms \cup Asms^{+}) = \textsc{undec}(LabAsm)$.
   By Theorem 6.1 in [7]: $Args = \{AP \vdash s \mid AP \subseteq \textsc{in}(LabAsm)\}$ is a complete argument extension of $\langle Ar_{ABA}, Att_{ABA} \rangle$.
   By Lemma 3: $Args^{+} = \{AP \vdash s \mid \exists \alpha \in AP \textit{ s.t. } \alpha \in \textsc{out}(LabAsm)\}$ and $Ar_{ABA} \setminus (Args \cup Args^{+}) = \{AP \vdash s \mid AP \nsubseteq \textsc{in}(LabAsm), \nexists \alpha \in AP \textit{ s.t. } \alpha \in \textsc{out}(LabAsm)\}$.
   By Theorem 10 in [3]: $\texttt{in}(LabArg) = Args$, $\texttt{out}(LabArg) = Args^{+}$, $\texttt{undec}(LabArg) = Ar_{ABA} \setminus (Args \cup Args^{+})$ is a complete argument labelling of $\langle Ar_{ABA}, Att_{ABA} \rangle$.
2. From right to left: We prove that $LabAsm$ satisfies Definition 2.
   - Let $\alpha \in \textsc{in}(LabAsm)$. Then $\{\alpha\} \vdash \alpha \in \texttt{in}(LabArg)$. Thus, all arguments $AP \vdash s$ attacking $\{\alpha\} \vdash \alpha$ are in $\texttt{out}(LabArg)$. So in each AP attacking α there is some β such that $\beta \in \textsc{out}(LabAsm)$.

- Let $\alpha \in \text{OUT}(LabAsm)$. Then $\{\alpha\} \vdash \alpha \in \texttt{out}(LabArg)$. Thus, there is an argument $AP \vdash s$ in $\texttt{in}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$. Hence, $AP \subseteq \text{IN}(LabAsm)$, so α is attacked by a set $AP \subseteq \text{IN}(LabAsm)$.
- Let $\alpha \in \text{UNDEC}(LabAsm)$. Then $\{\alpha\} \vdash \alpha \in \texttt{undec}(LabArg)$. Thus, there is no argument $AP_1 \vdash s_1$ in $\texttt{in}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$ and there exists an argument $AP_2 \vdash s_2$ in $\texttt{undec}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$. Hence, there is no $AP_1 \subseteq \text{IN}(LabAsm)$ attacking α, so all sets AP_1 attacking α contain a γ with $LabAsm(\gamma) \neq \text{IN}$. Furthermore, there exists a set AP_2 attacking α such that $AP_2 \cap \text{OUT}(LabAsm) = \emptyset$. □

Theorem 5. *Let $LabArg$ be an argument labelling of $\langle Ar_{ABA}, Att_{ABA}\rangle$. If $LabArg$ is a complete argument labelling of $\langle Ar_{ABA}, Att_{ABA}\rangle$ then $LabAsm$ with*

- $\text{IN}(LabAsm) = \{\alpha \mid AP \vdash s \in \texttt{in}(LabArg), \alpha \in AP\}$,
- $\text{OUT}(LabAsm) = \{\alpha \mid \{\alpha\} \vdash \alpha \in \texttt{out}(LabArg)\}$,
- $\text{UNDEC}(LabAsm) = \{\alpha \mid \{\alpha\} \vdash \alpha \in \texttt{undec}(LabArg)\}$

is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ }\rangle$.

Proof. $LabAsm$ satisfies Definition 2:

- By Theorem 9 in [3], $\texttt{in}(LabArg)$ is a complete argument extension of $\langle Ar_{ABA}, Att_{ABA}\rangle$. By Theorem 6.1 in [7], $\{\alpha \in \mathcal{A} | AP \vdash s \in \texttt{in}(LabArg), \alpha \in AP\}$ is a complete assumption extension of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ }\rangle$. So by Theorem 2, $\text{IN}(LabAsm) = \{\alpha \mid AP \vdash s \in \texttt{in}(LabArg), \alpha \in AP\}$.
- Let $\alpha \in \text{OUT}(LabAsm)$, that is $\{\alpha\} \vdash \alpha \in \texttt{out}(LabArg)$. Then there exists an argument $AP \vdash s \in \texttt{in}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$. Thus, α is attacked by a set $AP \subseteq \text{IN}(LabAsm)$, satisfying Definition 2.
- Let $\alpha \in \text{UNDEC}(LabAsm)$, that is $\{\alpha\} \vdash \alpha \in \texttt{undec}(LabArg)$. Then, there is no argument $AP_1 \vdash s_1 \in \texttt{in}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$ and there is an argument $AP_2 \vdash s_2 \in \texttt{undec}(LabArg)$ attacking $\{\alpha\} \vdash \alpha$. Consequently, there is no $AP_1 \subseteq \text{IN}(LabAsm)$ attacking α, so all sets AP_1 attacking α contain a $\beta \notin \text{IN}(LabAsm)$. By Theorem 4, there exists a set AP_2 attacking α such that $AP_2 \cap \text{OUT}(LabAsm) = \emptyset$, satisfying Definition 2. □

Example 3. The corresponding AA framework of ABA_1 is $\langle Ar_{ABA_1}, Att_{ABA_1}\rangle$:

- $Ar_{ABA_1} = \{A_1 : \{\alpha\} \vdash \alpha\, ;\ A_2 : \{\beta\} \vdash \beta\, ;\ A_3 : \{\gamma\} \vdash \gamma\, ;$
  $A_4 : \{\alpha\} \vdash a\, ;\ A_5 : \{\beta\} \vdash a\, ;\ A_6 : \{\beta\} \vdash c\, ;\ A_7 : \{\gamma\} \vdash b\}$
- $Att_{ABA_1} = \{(A_4, A_1), (A_4, A_4), (A_5, A_1), (A_5, A_4), (A_6, A_3), (A_6, A_7),$
  $(A_7, A_2), (A_7, A_5), (A_7, A_6)\}$

$\langle Ar_{ABA_1}, Att_{ABA_1}\rangle$ has three complete argument labellings, corresponding to the three complete assumption labellings (see Example 1):

- $\texttt{in}(LabArg_1) = \emptyset$, $\texttt{out}(LabArg_1) = \emptyset$,
  $\texttt{undec}(LabArg_1) = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$
- $\texttt{in}(LabArg_2) = \{A_3, A_7\}$, $\texttt{out}(LabArg_2) = \{A_2, A_5, A_6\}$,
  $\texttt{undec}(LabArg_2) = \{A_1, A_4\}$
- $\texttt{in}(LabArg_3) = \{A_2, A_5, A_6\}$, $\texttt{out}(LabArg_3) = \{A_1, A_3, A_4, A_7\}$,
  $\texttt{undec}(LabArg_3) = \emptyset$

5. Conclusion

We introduced a labelling approach for ABA as a new way to express the complete extension semantics in ABA, where labels are assigned to single assumptions as opposed to the labelling of whole arguments in AA. We proved correspondence of this complete assumption labelling with the complete extension semantics in ABA as well as with the complete extension semantics and the complete argument labelling in AA. In contrast to the complete extension semantics in ABA, which only distinguishes between accepted (IN) and non-accepted (not IN) assumptions, the complete assumption labelling divides the non-accepted assumptions further into the ones rejected for sure (OUT) and neutral ones which are neither accepted nor rejected (UNDEC). This is an advantage with respect to decision making, e.g. medical treatment decision making, where it is important to know whether an assumption, e.g. that the patient has a certain allergy, is definitely rejected or whether it can neither be accepted nor rejected.

The idea to express argumentation semantics in terms of labellings has received considerable attention. Argument labellings have for example been used to create algorithms computing the semantics of an AA framework [9,10]. We will investigate whether the labelling approach for ABA introduced here can help with the implementation of efficient algorithms for computing the complete semantics in ABA. Future work also includes extending the assumption labelling to other semantics in ABA and considering labellings in non-flat ABA frameworks.

Acknowledgements

We thank Martin Caminada for fruitful preliminary discussion.

References

[1] P.M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 1995.

[2] M. Caminada. Semi-stable semantics. In *COMMA*, 2006.

[3] M. Caminada and D.M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 2009.

[4] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 1997.

[5] P.M. Dung, R.A. Kowalski, and F. Toni. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*. Springer US, 2009.

[6] P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 2007.

[7] M. Caminada, S. Sà, J. Alcântara, and W. Dvořák. On the difference between assumption-based argumentation and abstract argumentation. Proceedings of BNAIC 2013.

[8] M. Caminada. On the issue of reinstatement in argumentation. In *Logics in Artificial Intelligence*. Springer Berlin Heidelberg, 2006.

[9] S. Modgil and M. Caminada. Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in Artificial Intelligence*. Springer US, 2009.

[10] T. Wakaki and K. Nitta. Computing argumentation semantics in answer set programming. In *New Frontiers in Artificial Intelligence*. Springer Berlin Heidelberg, 2009.