Journal of Artificial Intelligence Research 60 (2017) 779-825 Submitted 06/17; published 12/17

On the Equivalence between Assumption-Based Argumentation and Logic Programming

Martin Caminada CaminadaM@cardiff.ac.uk
Cardiff School of Computer Science & Informatics
Cardiff University, UK

Claudia Schulz schulz@ukp.informatik.tu-darmstadt.de
Ubiquitous Knowledge Processing (UKP) Lab
TU Darmstadt, Germany

Abstract

Assumption-Based Argumentation (ABA) has been shown to subsume various other non-monotonic reasoning formalisms, among them normal logic programming (LP). We re-examine the relationship between ABA and LP and show that normal LP also subsumes (flat) ABA. More precisely, we specify a procedure that given a (flat) ABA framework yields an associated logic program with almost the same syntax whose semantics coincide with those of the ABA framework. That is, the 3-valued stable (respectively well-founded, regular, 2-valued stable, and ideal) models of the associated logic program coincide with the complete (respectively grounded, preferred, stable, and ideal) assumption labellings and extensions of the ABA framework. Moreover, we show how our results on the translation from ABA to LP can be reapplied for a reverse translation from LP to ABA, and observe that some of the existing results in the literature are in fact special cases of our work. Overall, we show that (flat) ABA frameworks can be seen as normal logic programs with a slightly different syntax. This implies that methods developed for one of these formalisms can be equivalently applied to the other by simply modifying the syntax.

1. Introduction

The formal study of argumentation in Artificial Intelligence aims at modelling argumentative reasoning, including for example constructing arguments from given knowledge, evaluating conflicting arguments to determine argument validity, and handling preferences over arguments or given information (Rahwan & Simari, 2009; Besnard, García, Hunter, Modgil, Prakken, Simari, & Toni, 2014). *Assumption-Based Argumentation* (ABA) (Bondarenko, Dung, Kowalski, & Toni, 1997; Dung, Kowalski, & Toni, 2009; Toni, 2014) has become one of the leading approaches for formal argumentation which has proven useful in a wide range of application areas such as decision making (Matt, Toni, Stournaras, & Dimitrelos, 2008; Dung, Thang, & Toni, 2008), multi-agent dialogues (Fan, Toni, & Hussain, 2010; Fan & Toni, 2014), legal reasoning (Dung & Thang, 2009; Dung, Thang, & Hung, 2010), and medicine (Craven, Toni, Cadar, Hadad, & Williams, 2012). It provides methods for the construction of arguments from given inference rules and defeasible information, called *assumptions*, as well as for the identification of attacks between assumptions based on the notions of arguments and contraries of assumptions. The semantics of an ABA framework are given in terms of sets of acceptable assumptions, which are able to defend themselves against attacking assumptions. The most frequently studied fragment of ABA are *flat* ABA frameworks (e.g., Dung, Kowalski, & Toni, 2006; Dung, Mancarella, & Toni, 2007; Dung et al., 2009; Toni, 2013; Fan & Toni, 2014), where assumptions cannot be derived using inference rules.

ABA has a well-studied relationship to abstract argumentation (AA) (Dung, 1995b), where arguments and attacks between them are given rather than constructed from knowledge and semantics are defined in terms of sets of acceptable *arguments*, in that both flat ABA is an instance of AA (Dung et al., 2007; Toni, 2014; Caminada, Sá, Alcântara, & Dvořák, 2015b) and AA is an instance of ABA (Toni, 2012) under many well-studied semantics. In addition to the "normal" notion of acceptability used in abstract argumentation and many other argumentation formalisms, ABA is equipped with a dialectical notion of acceptability (Toni, 2013) which makes it particularly suitable for many applications.

In addition to its relationship with AA, it has been shown that ABA is powerful enough to capture various non-monotonic reasoning formalisms such as default logic, circumscription, autoepistemic logic, and – most importantly for this paper – logic programming (LP) (Bondarenko et al., 1997; Toni, 2007, 2008; Schulz & Toni, 2015). More precisely, the aforementioned formalisms can be translated into ABA frameworks in such a way that the semantics of the resulting ABA framework and of the respective formalism correspond. This allows to apply methods developed for ABA, e.g. the dialectical notion of acceptability, to the captured non-monotonic formalisms and has for example proven useful for explaining (Schulz & Toni, 2016) as well as visualizing (Schulz, 2015) logic programs under certain semantics. Importantly, the translation from a logic program yields a *flat* ABA framework.

In the current paper, we investigate the *opposite direction*, that is we define a translation from a (flat) ABA framework to an associated logic program and show that LP is powerful enough to capture the commonly used semantics of (flat) ABA. In particular, we study the complete, grounded, preferred, stable, and ideal ABA semantics, which we will refer to as *common ABA semantics*, and prove that they correspond to, respectively, the 3-valued stable, well-founded, regular, (2-valued) stable, and ideal model semantics for LP. Importantly, these results do not follow from previous work on translating LP to ABA (Bondarenko et al., 1997; Schulz & Toni, 2015).

Correspondence under certain semantics between an ABA framework and some translation into a logic program is not surprising since existing translations from ABA to AA (Dung et al., 2007; Caminada et al., 2015b; Schulz & Toni, 2017) and from AA to LP (Osorio, Zepeda, Nieves, & Cortés, 2005; Wu, Caminada, & Gabbay, 2009; Strass, 2013; Caminada, Sá, Alcântara, & Dvořák, 2015a) can be used to show correspondence under certain semantics between an ABA framework and the logic program resulting from concatenation. However, the concatenation may lead to an exponential blow-up as it requires the construction of all arguments from an ABA framework and is therefore not suitable for practical matters such as the computation of ABA semantics using LP tools. Furthermore, in the translation through concatenation, valuable information about the relation between sentences in ABA gets lost. In contrast, our novel translation yields a logic program that is syntactically very close to the inference rules of the ABA framework and thus preserves the information in an ABA framework. It furthermore does not suffer from exponential blow-up, making it suitable for the application of LP methods to ABA frameworks. Semantic correspondence between ABA frameworks and *our* translation does therefore not follow from existing correspondence results (as they concern a different translation). In addition, we also show that for a restricted class of ABA frameworks, the semi-stable ABA semantics and 3-valued L-stable semantics of the logic program obtained through our translation correspond. This result cannot be obtained through existing translations from ABA to AA and from AA to LP.

The implications of our novel translation and our correspondence results are twofold: Firstly, our results pave the way for the application of methods developed for LP to (flat) ABA frameworks since our translation prevents an exponential blow-up while preserving the semantics. Efficient computation methods for LP semantics could for instance be used to determine the semantics of ABA frameworks, which is a promising direction for future work. Secondly, our results illustrate not only that there exists a semantic correspondence between (flat) ABA and LP (as obtained through existing results), but that (flat) ABA can in fact be seen as LP with a slightly different syntax.

Our work contributes to an ongoing effort to compare non-monotonic reasoning formalisms and find translations between them that preserve the respective semantics. Starting with comparisons of different non-monotonic logics such as default logic, circumscription, and autoepistemic logic in the nineteen-nineties (Imielinski, 1987; Gottlob, 1995; Janhunen, 1999), attention has recently shifted towards the comparison of different extensions of abstract argumentation frameworks such as evidential argumentation frameworks and bipolar argumentation frameworks (Oren, Reed, & Luck, 2010; Cayrol & Lagasquie-Schiex, 2013; Polberg & Oren, 2014). Furthermore, the relationship between different structured argumentation frameworks as well as their relation to abstract argumentation and other non-monotonic reasoning formalisms has received considerable attention in recent years (Chesñevar, Dix, Stolzenburg, & Simari, 2003; Vesic, 2013; Dung & Thang, 2014; Schulz & Toni, 2015; Caminada et al., 2015b; Heyninck & Straßer, 2016; Young, Modgil, & Rodrigues, 2016; Grooters & Prakken, 2016).

This paper is structured as follows. First, we recall the definitions of ABA frameworks and logic programs as well as of their respective semantics in Section 2 and illustrate how the notion of an argument can be defined in terms of logic programs. In Section 3 we then provide a translation from an ABA framework to an associated logic program and show that the semantics of the two coincide. We subsequently show how our results on the translation from ABA to LP can be reapplied for a reverse translation from LP to ABA in Section 4 and observe that some of the existing results in the literature are in fact special cases of our work. In Section 5 we then discuss the relationship between LP and different fragments of ABA frameworks, as well as their relationship under other semantics. In Section 6 we conclude and discuss the implications of our work.[1]

2. Formal Preliminaries

In the current section, we provide a number of key definitions on Assumption-Based Argumentation (ABA) and Logic Programming (LP).

1. Parts of the results of this paper have been presented at the *first international workshop on argumentation and logic programming (ArgLP)* (Caminada & Schulz, 2015). The current paper provides an extended and thoroughly revised version of these results. In particular, we have extended our initial results, which hold for a very specialized fragment of ABA, to general *flat* ABA frameworks and we have added two appendices (A and B) containing full proofs.

2.1 Assumption-Based Argumentation

Definition 1. *An* Assumption-Based Argumentation (ABA) framework *is a tuple* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ *where* $\mathcal{L}$ *is a language,* $\mathcal{R}$ *is a set of inference rules based on this language,* $\mathcal{A} \subseteq \mathcal{L}$ *is a (non-empty) set of assumptions, and* $\bar{\ } : \mathcal{A} \rightarrow \mathcal{L}$ *is a function that maps each assumption* $\chi \in \mathcal{A}$ *to what is called its* contrary.

We denote by $\overline{\chi}$ the sentence in $\mathcal{L}$ that is the contrary of assumption χ according to the mapping $\bar{\ }$, for example $\overline{\chi} = \psi$ where $\psi \in \mathcal{L}$. As a convention, we will use lower case greek letters for assumptions and lower case Latin letters for non-assumption sentences. An ABA framework is said to be *flat* (Bondarenko et al., 1997) iff assumptions only occur in the body of the inference rules, and not in the head. From here onwards, and if not stated otherwise, we assume that ABA frameworks are flat. Furthermore, we notice that each assumption has just a single contrary. Although this deviates from some generalized work on ABA, where each assumption has a *set* of contraries (Gaertner & Toni, 2007, 2008; Fan & Toni, 2014, 2015), or where a set of sentences (containing at least one assumption) is associated with a set of sentences which together form the contrary (Toni, 2007) of the first set, in a lot of work on ABA (e.g., Dung et al., 2007, 2009; Toni, 2014; Schulz & Toni, 2014) it is common for the authors to restrict themselves to assumptions with single contraries as originally defined by Bondarenko et al. (1997). In addition, we will often restrict ourselves to a fragment of ABA where the contrary of an assumption cannot be an assumption itself (that is, $\bar{\ } : \mathcal{A} \rightarrow \mathcal{L} \setminus \mathcal{A}$), which we call *normal ABA frameworks*. This is the type of ABA framework that is for example obtained when translating a logic program to an ABA framework (Bondarenko et al., 1997; Schulz & Toni, 2015). In Section 5 we show that restricting ourselves to normal ABA frameworks does not affect the generality of our results since any flat ABA framework (including those where assumptions have sets of contraries) can be equivalently expressed as a normal ABA framework.

Definition 2. *Let* $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ *be an ABA framework. An* ABA argument $Asms \vdash x$ *for conclusion* $x \in \mathcal{L}$ *supported by assumptions* $Asms \subseteq \mathcal{A}$ *is a finite tree with nodes labelled with sentences in* $\mathcal{L}$ *or with the special symbol* `TRUE`[2]*, such that:*

- *the root is labelled with* x
- *for every node* N*:*
  - *if* N *is a leaf node, then* N *is labelled with an assumption or with* `TRUE`
  - *if* N *is not a leaf node and* $z \in \mathcal{L}$ *is the label of* N*, then there exists a rule in* $\mathcal{R}$ *of the form* $z \leftarrow y_1, \ldots, y_n$ *and either* $n = 0$ *and* N *has just a single child that is labelled with* `TRUE`*, or* $n > 0$ *and* N *has* n *children, labelled with* $y_1, \ldots, y_n$ *respectively*
  - $Asms$ *is the set of all assumptions labelling leaf nodes*

We say that an ABA argument $Asms \vdash \chi$ is *trivial* iff it consists of a single node, which implies that χ is an assumption and $Asms = \{\chi\}$.

2. We assume that the special symbol `TRUE`, just like the special symbols `FALSE` and `UNDEFINED` do not occur in any ABA framework. So $\texttt{TRUE}, \texttt{FALSE}, \texttt{UNDEFINED} \notin \mathcal{L}$.

Based on the definition of an ABA argument, we proceed to introduce ABA semantics. For this, we apply the notion of assumption labellings (Schulz & Toni, 2014) instead of extensions (Bondarenko et al., 1997). As we will observe later, the three labels of an assumption labelling provide a straightforward correspondence with the three values of logic programming models.

Definition 3. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework. An* assumption labelling *of $\mathcal{F}$ is a total function $\mathcal{L}ab : \mathcal{A} \rightarrow \{\textsc{in}, \textsc{out}, \textsc{undec}\}$. We denote by $\textsc{in}(\mathcal{L}ab)$ the set of all assumptions labelled* IN *by $\mathcal{L}ab$, and similarly by $\textsc{out}(\mathcal{L}ab)$ and $\textsc{undec}(\mathcal{L}ab)$ the sets of assumptions labelled* OUT *and* UNDEC*, respectively. An assumption labelling $\mathcal{L}ab$ is called a* complete assumption labelling *of $\mathcal{F}$ iff for each $\chi \in \mathcal{A}$ it holds that:*

1. *if $\mathcal{L}ab(\chi) = \textsc{in}$ then for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \cap \textsc{out}(\mathcal{L}ab) \neq \emptyset$*
2. *if $\mathcal{L}ab(\chi) = \textsc{out}$ then there exists an ABA argument $Asms \vdash \overline{\chi}$ such that $Asms \subseteq \textsc{in}(\mathcal{L}ab)$*
3. *if $\mathcal{L}ab(\chi) = \textsc{undec}$ then there exists an ABA argument $Asms \vdash \overline{\chi}$ such that $Asms \cap \textsc{out}(\mathcal{L}ab) = \emptyset$, and for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \nsubseteq \textsc{in}(\mathcal{L}ab)$*

We will often denote a labelling $\mathcal{L}ab$ as a triple $\langle \textsc{in}(\mathcal{L}ab), \textsc{out}(\mathcal{L}ab), \textsc{undec}(\mathcal{L}ab) \rangle$.

The following proposition states that we are free to change the direction of the three if-statements in Definition 3. This will be a useful property for some of the proofs.

Proposition 1. *(Schulz & Toni, 2014) Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework, and let $\mathcal{L}ab$ be an assumption labelling of $\mathcal{F}$. $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$ iff for each $\chi \in \mathcal{A}$ it holds that:*

1. *if for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \cap \textsc{out}(\mathcal{L}ab) \neq \emptyset$, then $\mathcal{L}ab(\chi) = \textsc{in}$*
2. *if there exists an ABA argument $Asms \vdash \overline{\chi}$ such that $Asms \subseteq \textsc{in}(\mathcal{L}ab)$, then $\mathcal{L}ab(\chi) = \textsc{out}$*
3. *if there exists an ABA argument $Asms \vdash \overline{\chi}$ such that $Asms \cap \textsc{out}(\mathcal{L}ab) = \emptyset$, and for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \nsubseteq \textsc{in}(\mathcal{L}ab)$, then $\mathcal{L}ab(\chi) = \textsc{undec}$*

The notion of assumption labellings has been extended from the complete semantics as introduced by Schulz and Toni (2014) to other well-known ABA semantics (Schulz & Toni, 2017), which were previously defined in terms of extensions rather than labellings (Bondarenko et al., 1997; Toni, 2014). Note that there exists a one-to-one correspondence between the assumption labellings and the assumption extensions of an ABA framework (Schulz & Toni, 2014, 2017). In essence, the set of IN-labelled assumptions of a complete (respectively grounded, preferred, stable, or ideal) assumption labelling constitutes a complete (respectively grounded, preferred, stable, or (maximal) ideal) assumption extension.

Definition 4. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework. A complete assumption labelling $\mathcal{L}ab$ of $\mathcal{F}$ is called:*

1. *a* grounded assumption labelling *iff* IN($\mathcal{L}ab$) *is minimal (w.r.t.* $\subseteq$*) among all complete assumption labellings of* $\mathcal{F}$

2. *a* preferred assumption labelling *iff* IN($\mathcal{L}ab$) *is maximal (w.r.t.* $\subseteq$*) among all complete assumption labellings of* $\mathcal{F}$

3. *a* stable assumption labelling *iff* UNDEC($\mathcal{L}ab$) $= \emptyset$

4. *an* ideal assumption labelling *iff* IN($\mathcal{L}ab$) *is maximal (w.r.t.* $\subseteq$*) among all complete assumption labellings of* $\mathcal{F}$ *satisfying that for all preferred assumption labellings* $\mathcal{L}ab_{pref}$ *of* $\mathcal{F}$*,* IN($\mathcal{L}ab$) $\subseteq$ IN($\mathcal{L}ab_{pref}$)

Any ABA framework has one or more complete assumption labellings, in particular, a unique grounded assumption labelling, one or more preferred assumption labellings, zero or more stable asstumption labellings, and a unique ideal assumption labelling.

As complete, grounded, preferred, stable, and ideal semantics are well-studied in the ABA literature (e.g., Čyras & Toni, 2015; Baláz, Frtús, Flouris, Homola, & Šefránek, 2014; Toni, 2014; Dunne, 2009), we refer to these as the *common ABA semantics*[3]. These different semantics have proven suitable for different applications. For example, the preferred semantics is used in the context of legal reasoning with ABA (Dung & Thang, 2009), whereas grounded and ideal semantics are suitable for modelling agent dialogues (Fan & Toni, 2014).

The following example illustrates the difference between the various common ABA semantics and will be used as a running example throughout this paper.

Example 1. *Let* $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ *be an ABA framework with*

- $\mathcal{A} = \{\alpha, \beta, \gamma, \delta, \epsilon, \phi, \lambda\}$
- $\mathcal{L} = \mathcal{A} \cup \{a, b, c, d, e, f, l\}$
- $\overline{\alpha} = f,\ \overline{\beta} = b,\ \overline{\gamma} = c,\ \overline{\delta} = d,\ \overline{\epsilon} = e,\ \overline{\phi} = f,\ \overline{\lambda} = l$
- $\mathcal{R} = \{a \leftarrow\ ;\ b \leftarrow a, \gamma\ ;\ c \leftarrow \beta\ ;\ d \leftarrow b, e, \delta\ ;\ e \leftarrow a, \alpha\ ;\ e \leftarrow a, \phi\ ;\ f \leftarrow \epsilon, \phi\}$

$\mathcal{F}$ has six complete assumption labellings:
$\mathcal{L}ab_1 = \langle \{\lambda\}, \emptyset, \{\alpha, \beta, \gamma, \delta, \epsilon, \phi\} \rangle$, $\mathcal{L}ab_2 = \langle \{\alpha, \phi, \lambda\}, \{\epsilon\}, \{\beta, \gamma, \delta\} \rangle$,
$\mathcal{L}ab_3 = \langle \{\gamma, \lambda\}, \{\beta\}, \{\alpha, \delta, \epsilon, \phi\} \rangle$, $\mathcal{L}ab_4 = \langle \{\alpha, \phi, \gamma, \lambda\}, \{\beta, \epsilon\}, \{\delta\} \rangle$,
$\mathcal{L}ab_5 = \langle \{\beta, \delta, \lambda\}, \{\gamma\}, \{\alpha, \epsilon, \phi\} \rangle$, $\mathcal{L}ab_6 = \langle \{\alpha, \beta, \delta, \phi, \lambda\}, \{\gamma, \epsilon\}, \emptyset \rangle$.
$\mathcal{L}ab_1$ is the grounded assumption labelling, $\mathcal{L}ab_4$ and $\mathcal{L}ab_6$ are the preferred assumption labellings, $\mathcal{L}ab_6$ is the only stable assumption labelling, and $\mathcal{L}ab_2$ is the ideal assumption labelling.

Next, we introduce a new result, namely that the set of IN assumptions of a complete assumption labelling $\mathcal{L}ab_1$ is a subset of the set of IN assumptions of another complete assumption labelling $\mathcal{L}ab_2$ iff the set of OUT assumptions of $\mathcal{L}ab_1$ is a subset of the set of OUT assumptions of $\mathcal{L}ab_2$.

3. We here restrict ourselves to semantics that can be defined based on the complete semantics and therefore do not deal with the admissible semantics.

Lemma 1. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ be an ABA framework, and let $\mathcal{L}ab_1$ and $\mathcal{L}ab_2$ be complete assumption labellings of $\mathcal{F}$. It holds that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$ iff $\text{OUT}(\mathcal{L}ab_1) \subseteq \text{OUT}(\mathcal{L}ab_2)$.*

Proof. "$\Rightarrow$": Assume that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$. Let $\chi \in \text{OUT}(\mathcal{L}ab_1)$. Then, by the definition of a complete assumption labelling (Definition 3) there exists an ABA argument $Asms \vdash \overline{\chi}$ with $Asms \subseteq \text{IN}(\mathcal{L}ab_1)$. Since $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$ it follows that $Asms \subseteq \text{IN}(\mathcal{L}ab_2)$. So by Proposition 1 (point 2), $\chi \in \text{OUT}(\mathcal{L}ab_2)$.
"$\Leftarrow$": Assume that $\text{OUT}(\mathcal{L}ab_1) \subseteq \text{OUT}(\mathcal{L}ab_2)$. Let $\chi \in \text{IN}(\mathcal{L}ab_1)$. Then, by the definition of a complete assumption labelling (Definition 3) it holds that each ABA argument $Asms \vdash \overline{\chi}$ has $Asms \cap \text{OUT}(\mathcal{L}ab_1) \neq \emptyset$. Since $\text{OUT}(\mathcal{L}ab_1) \subseteq \text{OUT}(\mathcal{L}ab_2)$ it follows that $Asms \cap \text{OUT}(\mathcal{L}ab_2) \neq \emptyset$. So by Proposition 1 (point 1), $\chi \in \text{IN}(\mathcal{L}ab_2)$. □

2.2 Logic Programming

We now shift our attention to logic programming. We start with formally introducing the notion of a logic program. For current purposes, we restrict ourselves to normal logic programs.[4]

Definition 5. *A* logic programming rule *is an expression $x \leftarrow y_1, \ldots, y_n, \texttt{not}\ z_1, \ldots, \texttt{not}\ z_m$ ($n \geq 0$, $m \geq 0$) where x, each y_i ($1 \leq i \leq n$) and each z_j ($1 \leq j \leq m$) is an atom, and* `not` *represents negation as failure (NAF). We say that x is the* head *of the rule, and $y_1, \ldots, y_n, \texttt{not}\ z_1, \ldots, \texttt{not}\ z_m$ the body of the rule. Moreover, we say that $y_1, \ldots, y_n$ is the strong part of the body, and $\texttt{not}\ z_1, \ldots, \texttt{not}\ z_m$ is the weak part of the body. We assume the presence of three special atoms* `TRUE`, `FALSE` *and* `UNDEFINED`, *which can only occur in the strong part of the body. A* NAF literal *is an expression $\texttt{not}\ w$, where w is an atom. We say a rule is NAF-free iff it does not contain any NAF literal (that is, iff $m = 0$). A* logic program *P consists of a set of logic programming rules. A logic program is NAF-free iff each of its rules is NAF-free. The Herbrand Base of a logic program P (written as HB_P) is the set of all atoms in P (excluding the special atoms* `TRUE`, `FALSE` *and* `UNDEFINED`*). We denote by $HB_P^{\texttt{not}} = \{\texttt{not}\ w \mid w \in HB_P\}$ the set of all NAF literals of atoms in the Herbrand Base.*

In the following, we recall the definitions of LP semantics.

Definition 6. *A* 3-valued interpretation *of a logic program P with respect to a set of atoms $Atms \supseteq HB_P$ is a pair $\langle T, F \rangle$ where $T, F \subseteq Atms$ and $T \cap F = \emptyset$.*

Definition 7. *A 3-valued interpretation $\langle T, F \rangle$ of a NAF-free logic program P w.r.t. $Atms \supseteq HB_P$ is a* 3-valued model *of P w.r.t. $Atms$ if for all logic programming rules $x \leftarrow y_1, \ldots, y_n$ in P it holds that*

- *$x \in T$ or*
- *$x \in F$ and $\exists i \in \{1, \ldots, n\} : y_i \in F \vee y_i = \texttt{FALSE}$ or*

4. We recall that a logic program is called *normal* iff it does not contain strong negation, and does not contain any disjunction in the head of any rule (e.g., Alferes & Pereira, 1992; Brogi, Lamma, Mancarella, & Mello, 1992).

- $x \in Atms \setminus (T \cup F)$ *and* $\exists i \in \{1, \ldots, n\} : y_i \in F \vee y_i \in Atms \setminus (T \cup F) \vee y_i = \texttt{FALSE} \vee y_i = \texttt{UNDEFINED}$

When P is a NAF-free logic program (possibly containing `TRUE`, `FALSE` or `UNDEFINED`), we write $\Phi_{Atms}(P)$ for its unique *minimal* 3-valued model $\langle T, F\rangle$ (w.r.t. *Atms*), i.e. $\langle T, F\rangle$ has minimal T and maximal F (w.r.t. $\subseteq$) among all 3-valued models of P w.r.t. *Atms*.

Definition 8. *The* reduct *of a logic program P w.r.t. a 3-valued interpretation $\mathcal{M}od = \langle T, F\rangle$, written as $P^{\mathcal{M}od}$, is obtained by replacing in every rule each NAF literal* `not` x *by* `TRUE` *if $x \in F$, by* `FALSE` *if $x \in T$, and by* `UNDEFINED` *otherwise.*

Since $P^{\mathcal{M}od}$ is a NAF-free program, it has a unique minimal 3-valued model, written as $\Phi_{HB_P}(P^{\mathcal{M}od})$.[5]

We now recall various logic programming semantics which are based on 3-valued interpretations (Przymusinski, 1990). Notice that although our definition of well-founded and regular models is slightly different from what is in the literature, equivalence is shown by Caminada et al. (2015a). We also define a new semantics based on 3-valued models, namely *ideal models*[6], inspired by the idea of ideal scenarios for logic programs (Alferes, Dung, & Pereira, 1993). In fact our ideal models coincide with ideal scenarios, as shown in Appendix B.

Definition 9. *Let P be a logic program and $\mathcal{M}od = \langle T, F\rangle$ a 3-valued interpretation of P w.r.t. HB_P. We say that $\mathcal{M}od$ is:*

- *a* 3-valued stable model *iff* $\Phi_{HB_P}(P^{\mathcal{M}od}) = \mathcal{M}od$
- *a* well-founded model *iff $\mathcal{M}od$ is a 3-valued stable model where T is minimal (w.r.t. $\subseteq$) among all 3-valued stable models of P*
- *a* regular model *iff $\mathcal{M}od$ is a 3-valued stable model where T is maximal (w.r.t. $\subseteq$) among all 3-valued stable models of P (Eiter, Leone, & Saccà, 1997)*
- *a* (2-valued) stable model *iff $\mathcal{M}od$ is a 3-valued stable model where $T \cup F = HB_P$*
- *an* ideal model *iff $\mathcal{M}od$ is a 3-valued stable model where T is maximal (w.r.t. $\subseteq$) among all 3-valued stable models of P satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$ of P, $T \subseteq T_{reg}$*

Any logic program has one or more 3-valued stable models, in particular, a unique well-founded model, one or more regular models, zero or more (2-valued) stable models, and a unique ideal model.

We sometimes refer to 3-valued stable, well-founded, regular, (2-valued) stable and ideal semantics as the *common LP semantics*.

5. Please be aware that we have made the formalization of Przymusinski (1990) a bit more precise by explicitly mentioning that the unique minimal 3-valued model is with respect to HB_P and not for instance with respect to $HB_{P^{\mathcal{M}od}}$. To see why this matters, consider the logic program $P = \{a \leftarrow \texttt{not}\ b\}$. When we take $\mathcal{M}od$ to be $\langle\{a\}, \{b\}\rangle$ it holds that $HB_P = \{a, b\}$ and $HB_{P^{\mathcal{M}od}} = \{a\}$ (as $P^{\mathcal{M}od} = \{a \leftarrow \texttt{TRUE}\}$) so $\Phi_{HB_P}(P^{\mathcal{M}od}) = \mathcal{M}od$ whereas $\Phi_{HB_{P^{\mathcal{M}od}}}(P^{\mathcal{M}od}) = \langle\{a\}, \emptyset\rangle \neq \mathcal{M}od$. This illustrates that for defining a 3-valued stable model in a meaningful way, one has to do so with respect to the Herbrand Base of the *original* program P (not the reduced one).
6. Note that our definition of ideal models is different from the one by Nieves and Osorio (2016) which is not inspired by the ideal scenario semantics.

Example 2. *Let P be the the logic program $\{a \leftarrow\ ;\ b \leftarrow a, \texttt{not}\ c\ ;\ c \leftarrow \texttt{not}\ b\ ;\ d \leftarrow b, e, \texttt{not}\ d\ ;\ e \leftarrow a, \texttt{not}\ f\ ;\ f \leftarrow \texttt{not}\ e, \texttt{not}\ f\}$. The common LP semantics of P are as follows. There are six 3-valued stable models:*
$\mathcal{M}od_1 = \langle \{a\}, \emptyset \rangle$, $\mathcal{M}od_2 = \langle \{a, e\}, \{f\} \rangle$, $\mathcal{M}od_3 = \langle \{a, b\}, \{c\} \rangle$,
$\mathcal{M}od_4 = \langle \{a, b, e\}, \{c, f\} \rangle$, $\mathcal{M}od_5 = \langle \{a, c\}, \{b, d\} \rangle$, $\mathcal{M}od_6 = \langle \{a, c, e\}, \{b, d, f\} \rangle$.
$\mathcal{M}od_1$ is the well-founded model, $\mathcal{M}od_4$ and $\mathcal{M}od_6$ are the regular models, $\mathcal{M}od_6$ is the only (2-valued) stable model, and $\mathcal{M}od_2$ is the ideal model.

Just as was done for ABA, we can also define arguments in the context of logic programming.[7]

Definition 10. *Let P be a logic program. An* LP argument *for $x \in HB_P$ (the conclusion) is a finite tree with nodes labelled with atoms in HB_P, NAF literals in $HB_P^{\texttt{not}}$, or with the special atoms* TRUE, FALSE *or* UNDEFINED *such that:*

- *the root is labelled with x*
- *for every node N:*
  - *if N is a leaf node, then N is labelled with a NAF literal or with one of the special atoms* TRUE, FALSE *or* UNDEFINED
  - *if N is not a leaf node and z is the label of N, then there exists a rule in P of the form $z \leftarrow y_1, \dots, y_n, \texttt{not}\ w_1, \dots, \texttt{not}\ w_m$ and either $m + n = 0$ and N has just a single child, that is labelled with* TRUE, *or $n + m > 0$ and N has $n + m$ children, labelled with $y_1, \dots, y_n, \texttt{not}\ w_1, \dots, \texttt{not}\ w_m$ respectively*

For NAF-free logic programs, atoms which are in T or F of the unique minimal 3-valued model have special LP arguments, as shown by the following proposition.

Proposition 2. *Let P be a NAF-free logic program, let $Atms \supseteq HB_P$, let $\langle T, F \rangle$ be $\Phi_{Atms}(P)$, and let $x \in HB_P$. It holds that:*

1. *$x \in T$ iff there exists an LP argument for x where every leaf node is labelled with* TRUE
2. *$x \in F$ iff each LP argument for x has at least one leaf node that is labelled with* FALSE

Proof. See Appendix A □

Example 3. *Let $P^{\mathcal{M}od_4}$ be the reduct of the logic program from Example 2 with respect to the 3-valued stable model $\mathcal{M}od_4 = \langle T_4, F_4 \rangle$ where $T_4 = \{a, b, e\}$ and $F_4 = \{c, f\}$. $P^{\mathcal{M}od_4}$ is the NAF-free logic program $\{a \leftarrow\ ;\ b \leftarrow a, \texttt{TRUE}\ ;\ c \leftarrow \texttt{FALSE}\ ;\ d \leftarrow b, e, \texttt{UNDEFINED}\ ;\ e \leftarrow a, \texttt{TRUE}\ ;\ f \leftarrow \texttt{FALSE}, \texttt{TRUE}\}$. Since $\mathcal{M}od_4$ is a 3-valued stable model of P, it holds that $\mathcal{M}od_4 = \Phi_{HB_P}(P^{\mathcal{M}od_4})$, so Proposition 2 applies to arguments of $P^{\mathcal{M}od_4}$ and its minimal 3-valued model $\mathcal{M}od_4$. Figure 1 depicts three such LP arguments: The only LP argument for b has all leaf nodes labelled* TRUE, *and $b \in T_4$; for $f \in F_4$ the only LP argument is such that some leaf node is labelled* FALSE; *and the only LP argument for d has not all leaf nodes labelled* TRUE *but has no leaf node labelled* FALSE, *so as stated in Proposition 2 $d \notin T_4$ and $d \notin F_4$.*

7. Our way of defining arguments in the context of LP is slightly different from what is done for instance by Dung (1995b), as our aim is to facilitate an easy translation between LP arguments and ABA arguments.

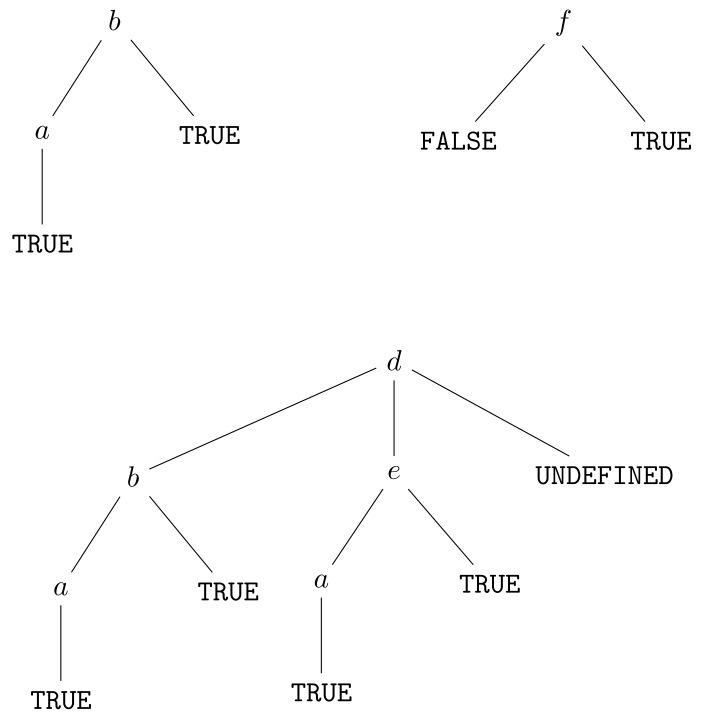

Figure 1: LP arguments constructed from $P^{\mathcal{M}od_4}$ (see Example 3).

3. Translating ABA Frameworks to Logic Programs

In order to compare ABA to logic programming, we first introduce a novel translation from a normal ABA framework to a logic program and then present semantic correspondence results. At the end of this section, we compare our new translation with a translation obtained from existing results.

3.1 A Novel Translation

The idea of our translation is to take the rules of the ABA framework and substitute each assumption by the NAF literal of the assumption's contrary. Note that this means that different assumptions might be substituted by the same NAF literal if they have the same contrary.

Definition 11. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ be a normal ABA framework. We define the* associated logic program $P_\mathcal{F}$ *as* $\{x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ \overline{\zeta_1}, \ldots, \mathtt{not}\ \overline{\zeta_m} \mid x \leftarrow y_1, \ldots, y_n, \zeta_1, \ldots, \zeta_m \in \mathcal{R}\}$.

Since we assume that no ABA framework contains the special symbols `TRUE`, `FALSE` or `UNDEFINED`, the associated logic program will not contain any of these symbols either. Note also that since $\mathcal{F}$ is a normal ABA framework, i.e. the contraries of assumptions are non-assumptions, $HB_{P_\mathcal{F}}$ contains no atoms which are assumptions in $\mathcal{F}$. As illustrated by the following example, an associated logic program may comprise less rules than the original

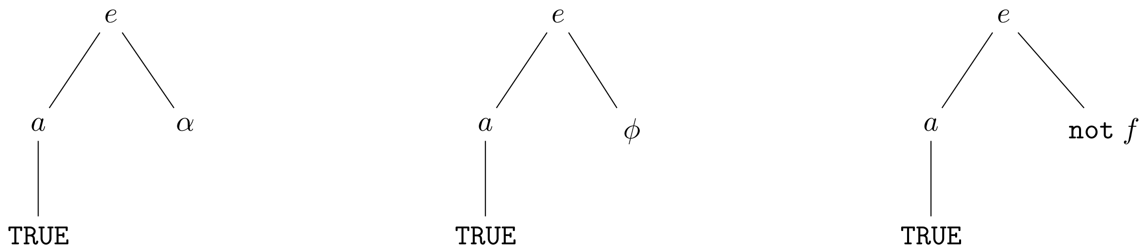

Figure 2: The two ABA arguments for e (left and middle) and the one LP argument for e (right) from Example 4.

ABA framework, which is due to the fact that different assumptions may have the same contrary.[8]

Example 4. *Consider again the ABA framework $\mathcal{F}$ from Example 1. The associated logic program $P_{\mathcal{F}}$ is the logic program introduced in Example 2. Note that $P_{\mathcal{F}}$ comprises one rule less than $\mathcal{F}$ since two ABA rules $e \leftarrow a, \alpha$ and $e \leftarrow a, \phi$ are translated to the same LP rule $e \leftarrow a, \texttt{not}\ f$. Furthermore, note that $l \in \mathcal{L}$ but $l \notin HB_{P_{\mathcal{F}}}$ since neither l nor λ occurs in an ABA rule.*

It can be observed that the translation from a normal ABA framework to a logic program can also be applied to translate ABA arguments (Definition 2) to LP arguments (Definition 10). For instance, in Example 4, an ABA argument for conclusion e has an associated LP argument for conclusion e. However, various ABA arguments may have the same associated LP argument, as illustrated in Figure 2. The reason is that different assumptions may be translated to the same NAF literal, e.g. α and ϕ in Example 4 are both translated to $\texttt{not}\ f$. In general, we observe that each non-trivial ABA argument in $\mathcal{F}$ has an associated LP argument in $P_{\mathcal{F}}$, and that each LP argument is associated to at least one ABA argument.

3.2 Semantic Correspondence

One of the main aims of the current paper is to examine how ABA semantics are related to logic programming semantics when translating an ABA framework to a logic program. For this, we introduce the functions `Lab2Mod` and `Mod2Lab` to convert between ABA assumption labellings and logic programming models.

To convert an assumption labelling to a 3-valued interpretation, we start by "inverting" the labelling. That is, we construct an interpretation $\langle T', F' \rangle$ where T' contains the contraries of the assumptions that are OUT, whereas F' contains the contraries of the assumptions that are IN. However, since we started with assumptions, this will only yield the status of atoms which are contraries of assumptions. In order to obtain the status of *all* atoms in the logic program (including those that are not the contrary of any assumption in the ABA framework) we perform a simple trick: apply the Gelfond-Lifschitz reduct.

8. Another subtle aspect of the translation is that ABA rules are composed of sentences whereas LP rules are composed of atoms (possibly inside of a NAF literal). In essence, the translation creates an LP atom for each (non-assumption) sentence that occurs in the ABA rules.

To convert a 3-valued interpretation to an assumption labelling, the idea is again to "invert" the interpretation. The assumptions whose contrary is in F will be labelled IN. The assumptions whose contrary is in T will be labelled OUT. The assumptions whose contrary is in the Herbrand Base, but not in T or F will be labelled UNDEC. The only remaining case is that of assumptions whose contrary is not in the Herbrand Base. This case occurs if there exist assumptions in the ABA framework which are not part of any inference rule and neither are their contraries. Thus, there are no ABA arguments for the contraries of these assumptions, so the assumptions can simply be labelled IN.

Definition 12. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be a normal ABA framework and let $P_{\mathcal{F}}$ be the associated logic program. We define a function* `Lab2Mod` *that, given a complete assumption labelling $\mathcal{L}ab$ of $\mathcal{F}$, yields the 3-valued interpretation $\Phi_{HB_{P_{\mathcal{F}}}}(P_{\mathcal{F}}^{\langle T',F' \rangle})$ where $T' = \{\overline{\chi} \mid \chi \in \text{OUT}(\mathcal{L}ab)\}$ and $F' = \{\overline{\chi} \mid \chi \in \text{IN}(\mathcal{L}ab)\} \cap HB_{P_{\mathcal{F}}}$.*[9] *We also define a function* `Mod2Lab` *that, given a 3-valued stable model $\langle T, F \rangle$ of $P_{\mathcal{F}}$, yields an assumption labelling $\mathcal{L}ab$ of $\mathcal{F}$ with $\text{IN}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in F\} \cup \{\chi \in \mathcal{A} \mid \overline{\chi} \notin HB_{P_{\mathcal{F}}}\}$, $\text{OUT}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in T\}$ and $\text{UNDEC}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in HB_{P_{\mathcal{F}}} \setminus (T \cup F)\}$.*

We observe that the functions `Lab2Mod` and `Mod2Lab` provide a one-to-one mapping between the complete assumption labellings of $\mathcal{F}$ and the 3-valued stable models of $P_{\mathcal{F}}$.

Theorem 2. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be a normal ABA framework and let $P_{\mathcal{F}}$ be the associated logic program. It holds that*

1. *if $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$ then* `Lab2Mod`*($\mathcal{L}ab$) is a 3-valued stable model of $P_{\mathcal{F}}$*
2. *if $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}}$ then* `Mod2Lab`*($\mathcal{M}od$) is a complete assumption labelling of $\mathcal{F}$*
3. `Lab2Mod` *and* `Mod2Lab` *are bijections which are each other's inverses*

Proof. See Appendix A ☐

Example 5. *Consider again $\mathcal{F}$ and $P_{\mathcal{F}}$ from Example 4. The six complete assumption labellings of $\mathcal{F}$ are given in Example 1, and the six 3-valued stable models of $P_{\mathcal{F}}$ in Example 2. It is easy to verify the correspondences between complete assumption labellings and 3-valued stable models, e.g. $\mathcal{M}od_1 =$* `Lab2Mod`*($\mathcal{L}ab_1$) and $\mathcal{L}ab_1 =$* `Mod2Lab`*($\mathcal{M}od_1$).*

Theorem 2 is important, since in ABA complete assumption labellings are the basis of various other semantics (like grounded, preferred, stable, and ideal), just like in logic programming 3-valued stable models are the basis of various other semantics (like well-founded, regular, (2-valued) stable, and ideal). For instance, where preferred semantics takes the complete assumption labellings and selects those with the maximal set of IN labelled assumptions, regular semantics takes the 3-valued stable models and selects those with maximal T. Hence, to prove equivalence between preferred semantics in ABA and regular semantics in logic programming, we need to show that there is an equivalence (through

9. Note that for any $\chi \in \text{OUT}(\mathcal{L}ab)$ it holds that there exists an inference rule with head $\overline{\chi}$, and therefore $\overline{\chi} \in HB_{P_{\mathcal{F}}}$. For T' it is thus not necessary to use the intersection with $HB_{P_{\mathcal{F}}}$.

the functions Lab2Mod and Mod2Lab) between the complete assumption labellings with maximal IN and the 3-valued stable models with maximal T. For this purpose, we first introduce the following lemma on the correspondence between sets of IN labelled assumptions of complete assumption labellings and sets T of 3-valued stable models.

Lemma 3. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be a normal ABA framework and let $P_{\mathcal{F}}$ be the associated logic program. Let $\mathcal{L}ab_1$ and $\mathcal{L}ab_2$ be complete assumption labellings of $\mathcal{F}$, and let $\mathcal{M}od_1 = \langle T_1, F_1 \rangle = \mathtt{Lab2Mod}(\mathcal{L}ab_1)$ and $\mathcal{M}od_2 = \langle T_2, F_2 \rangle = \mathtt{Lab2Mod}(\mathcal{L}ab_2)$. It holds that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$ iff $T_1 \subseteq T_2$.*

Proof. "$\Rightarrow$": Assume that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$. From $\mathcal{M}od_1 = \mathtt{Lab2Mod}(\mathcal{L}ab_1)$ it follows that $\langle T_1, F_1 \rangle = \Phi_{HB_{P_{\mathcal{F}}}}(P_{\mathcal{F}}^{\langle T'_1, F'_1 \rangle})$ with $T'_1 = \{\overline{\chi} \mid \chi \in \text{OUT}(\mathcal{L}ab_1)\} \cap HB_{P_{\mathcal{F}}}$ and $F'_1 = \{\overline{\chi} \mid \chi \in \text{IN}(\mathcal{L}ab_1)\} \cap HB_{P_{\mathcal{F}}}$. From $\mathcal{M}od_2 = \mathtt{Lab2Mod}(\mathcal{L}ab_2)$ it follows that $\langle T_2, F_2 \rangle = \Phi_{HB_{P_{\mathcal{F}}}}(P_{\mathcal{F}}^{\langle T'_2, F'_2 \rangle})$ with $T'_2 = \{\overline{\chi} \mid \chi \in \text{OUT}(\mathcal{L}ab_2)\} \cap HB_{P_{\mathcal{F}}}$ and $F'_2 = \{\overline{\chi} \mid \chi \in \text{IN}(\mathcal{L}ab_2)\} \cap HB_{P_{\mathcal{F}}}$. From the fact that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$ it follows (Lemma 1) that $\text{OUT}(\mathcal{L}ab_1) \subseteq \text{OUT}(\mathcal{L}ab_2)$, so we obtain that $T'_1 \subseteq T'_2$. Now, suppose that $x \in T_1$. From Proposition 2 it follows that there exists an LP argument for x in $P_{\mathcal{F}}^{\langle T'_1, F'_1 \rangle}$ such that each leaf node is labelled with TRUE. From the fact that $T'_1 \subseteq T'_2$ it then follows that the same LP argument also exists in $P_{\mathcal{F}}^{\langle T'_2, F'_2 \rangle}$. So $x \in T_2$.
"$\Leftarrow$": Since Lab2Mod and Mod2Lab are each other's inverses (point 3 of Theorem 2) it follows that $\mathcal{L}ab_1 = \mathtt{Mod2Lab}(\mathcal{M}od_1)$ and $\mathcal{L}ab_2 = \mathtt{Mod2Lab}(\mathcal{M}od_2)$. From the definition of Mod2Lab it then follows that $\text{OUT}(\mathcal{L}ab_1) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in T_1\}$ and $\text{OUT}(\mathcal{L}ab_2) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in T_2\}$. From $T_1 \subseteq T_2$ it then follows that $\text{OUT}(\mathcal{L}ab_1) \subseteq \text{OUT}(\mathcal{L}ab_2)$. From Lemma 1 it then follows that $\text{IN}(\mathcal{L}ab_1) \subseteq \text{IN}(\mathcal{L}ab_2)$. □

From the fact that for complete assumption labellings and 3-valued stable models Lab2Mod and Mod2Lab are each other's inverses, it follows that Lemma 3 can also be applied for two 3-valued stable models $\mathcal{M}od_1$ and $\mathcal{M}od_2$ of $P_{\mathcal{F}}$ and the associated assumption labellings $\mathcal{L}ab_1 = \mathtt{Mod2Lab}(\mathcal{M}od_1)$ and $\mathcal{L}ab_2 = \mathtt{Mod2Lab}(\mathcal{M}od_2)$ of $\mathcal{F}$.

The following theorem states the correspondence between the various semantics of ABA frameworks and their associated logic programs.

Theorem 4. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be a normal ABA framework and let $P_{\mathcal{F}}$ be the associated logic program. It holds that:*

1. *if $\mathcal{L}ab$ is a grounded assumption labelling of $\mathcal{F}$ then $\mathtt{Lab2Mod}(\mathcal{L}ab)$ is a well-founded model of $P_{\mathcal{F}}$*
2. *if $\mathcal{M}od$ is a well-founded model of $P_{\mathcal{F}}$ then $\mathtt{Mod2Lab}(\mathcal{M}od)$ is a grounded assumption labelling of $\mathcal{F}$*
3. *if $\mathcal{L}ab$ is a preferred assumption labelling of $\mathcal{F}$ then $\mathtt{Lab2Mod}(\mathcal{L}ab)$ is a regular model of $P_{\mathcal{F}}$*
4. *if $\mathcal{M}od$ is a regular model of $P_{\mathcal{F}}$ then $\mathtt{Mod2Lab}(\mathcal{M}od)$ is a preferred assumption labelling of $\mathcal{F}$*

5. *if $\mathcal{L}ab$ is a stable assumption labelling of $\mathcal{F}$ then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is a (2-valued) stable model of* $P_{\mathcal{F}}$

6. *if $\mathcal{M}od$ is a (2-valued) stable model of $P_{\mathcal{F}}$ then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is a stable assumption labelling of* $\mathcal{F}$

7. *if $\mathcal{L}ab$ is an ideal assumption labelling of $\mathcal{F}$ then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is an ideal model of* $P_{\mathcal{F}}$

8. *if $\mathcal{M}od$ is an ideal model of $P_{\mathcal{F}}$ then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is an ideal assumption labelling of* $\mathcal{F}$

Proof. See Appendix A. □

The reader can easily verify that there is a one-to-one correspondence between the common ABA semantics of the ABA framework $\mathcal{F}$ from Example 4 and the common LP semantics of its associated logic program $P_{\mathcal{F}}$. For example, the preferred assumption labellings of $\mathcal{F}$ (see Example 1) correspond to the regular models of $P_{\mathcal{F}}$ (see Example 2) in terms of `Lab2Mod` and `Mod2Lab`.

The semantics of an ABA framework were originally defined as assumption extensions, i.e. sets of "acceptable" assumptions (Bondarenko et al., 1997), rather than as assumption labellings. However, the two notions of ABA semantics coincide (Schulz & Toni, 2017), so 3-valued stable (respectively well-founded, regular, (2-valued) stable, ideal) models also correspond to complete (respectively grounded, preferred, stable, ideal) assumption extensions.

3.3 Translation from Existing Results

As pointed out in the introduction, it is not surprising that there exists a semantic correspondence for *some* translation from an ABA framework into a logic program due to the semantic correspondence of ABA and AA (Dung et al., 2007; Caminada et al., 2015b; Schulz & Toni, 2017) and AA and LP (Osorio et al., 2005; Wu et al., 2009; Strass, 2013; Caminada et al., 2015a). However, the translation obtained from concatenating the ABA to AA and the AA to LP translations is different from our translation, as illustrated in the following.

Example 6. *Let $\mathcal{F}$ be the ABA framework with:*

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

Translating $\mathcal{F}$ into an AA framework and then into a logic program using existing translations (Dung et al., 2007; Caminada et al., 2015b; Schulz & Toni, 2017; Osorio et al., 2005; Wu et al., 2009; Strass, 2013; Caminada et al., 2015a) yields the logic program $P = \{x \leftarrow\ ;\ y \leftarrow\ ;\ z \leftarrow \texttt{not}\ y\ ;\ v \leftarrow\ ;\ w \leftarrow \texttt{not}\ y\}$ (where the names of the atoms could be anything)[10]. In comparison, according to our translation we obtain the logic program $P_{\mathcal{F}} = \{a \leftarrow\ ;\ b \leftarrow a, \texttt{not}\ d\ ;\ c \leftarrow a, \texttt{not}\ b\}$, which is much closer to the underlying ABA framework. Note that apart from preserving the names of atoms, our translation preserves the dependencies between atoms, such as the dependency of b on a, which is lost in the concatenated translation. Thus, the semantic correspondence holding for the concatenated translation from ABA to LP does not straightforwardly carry over to *our* translation.

Another advantage of our translation is that it prevents exponential blow-up, which is important for future work on using LP tools for ABA.

Example 7. *Let $\mathcal{F}$ be the ABA framework with:*

- $\mathcal{A} = \{\alpha\}$
- $\mathcal{L} = \mathcal{A} \cup \{a_i, b_i \mid 0 \leq i \leq n\} \cup \{x\}$
- $\overline{\alpha} = x$
- $\mathcal{R} = \{a_0 \leftarrow\ ;\ b_0 \leftarrow\ ;\ a_{i+1} \leftarrow a_i\ ;\ a_{i+1} \leftarrow b_i\ ;\ b_{i+1} \leftarrow a_i\ ;\ b_{i+1} \leftarrow b_i\}$

where $n \in \mathbb{N}$

Translating $\mathcal{F}$ into an AA framework results in an exponential blow-up since the number of constructable arguments quadruples every time i is incremented. Then further translating it into a logic program thus yields a large logic program with exponential blow-up. In contrast, our translation results in a logic program with the same number of rules as $\mathcal{F}$.

4. Translating Logic Programs to ABA Frameworks

In the previous section, we studied a translation from normal ABA frameworks to logic programs, and observed that the various types of labellings of an ABA framework coincide with the various types of models of the associated logic program. In the current section, we go the other way around. That is, we examine a translation from a logic program to an ABA framework as done by Bondarenko et al. (1997) and Schulz and Toni (2015, 2016), and then show that existing correspondence results are generalised by our work.

4.1 Translation

Without loss of generality, we restrict the translation to logic programs without the special atoms `TRUE`, `FALSE` and `UNDEFINED`[11] (this is to prevent these atoms from occurring in the resulting ABA framework).

10. The AA framework obtained from $\mathcal{F}$ has the following arguments, where the letter in front of an argument denotes its name: $x : \{\} \vdash a$, $y : \{\alpha\} \vdash b$, $z : \{\beta\} \vdash c$, $v : \{\alpha\} \vdash \alpha$, $w : \{\beta\} \vdash \beta$.
11. Any logic program P containing these special atoms can be transformed to an equivalent logic program P' without these atoms by (1) removing each occurrence of `TRUE` from the bodies of the rules, (2) removing each rule that contains `FALSE` in its body, and (3) replacing each occurrence of `UNDEFINED` by a new atom u and adding a rule $u \leftarrow \texttt{not}\ u$. It can be verified that $\mathcal{M}od$ is a 3-valued stable (resp. well-founded, regular, 2-valued stable and ideal) model of P iff $\mathcal{M}od$ is a 3-valued stable (resp. well-founded, regular, 2-valued stable and ideal) model of P'. We refer to Appendix A (Proposition 3) for a formal proof.

Definition 13. *Let P be a logic program not containing the special atoms* TRUE, FALSE *and* UNDEFINED. *We define the* associated ABA framework $\mathcal{F}_P = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ *with* $\mathcal{A} = \{\mathtt{not\_w} \mid w \in HB_P\}$, $\mathcal{L} = HB_P \cup \mathcal{A}$, $\mathcal{R} = \{x \leftarrow y_1, \ldots, y_n, \mathtt{not\_z}_1, \ldots, \mathtt{not\_z}_m \mid x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ z_1, \ldots, \mathtt{not}\ z_m \in P\}$ *and* $\overline{\mathtt{not\_w}} = w$ *for every* $\mathtt{not\_w} \in \mathcal{A}$.

We define LP2ABA to be the function that, given a logic program P, yields the associated ABA framework $\mathcal{F}_P$ (Definition 13). Similarly, we define ABA2LP to be the function that, given a normal ABA framework $\mathcal{F}$, yields the associated logic program $P_{\mathcal{F}}$ (Definition 11).

We first show a syntactic feature of the two translations: translating a logic program to an associated ABA framework using LP2ABA, and then translating this ABA framework back to a logic program using ABA2LP yields the original logic program.

Lemma 5. *Let P be a normal logic program. It holds that* $\mathtt{ABA2LP}(\mathtt{LP2ABA}(P)) = P$.

Proof. Let $\mathcal{F}_P = \langle \mathcal{L}_P, \mathcal{R}_P, \mathcal{A}_P, \bar{\ } \rangle$ be $\mathtt{LP2ABA}(P)$.
"$\subseteq$": Let $x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ z_1, \ldots, \mathtt{not}\ z_m$ be a logic programming rule that is part of $\mathtt{ABA2LP}(\mathtt{LP2ABA}(P))$. Then from the definition of ABA2LP it follows that there exists an ABA rule $x \leftarrow y_1, \ldots, y_n, \zeta_1, \ldots, \zeta_m$ in $\mathcal{R}_P$ with $\zeta_i \in \mathcal{A}_P$ and $\overline{\zeta_i} = z_i$ $(1 \leq i \leq m)$. From the definition of LP2ABA it then follows that $\zeta_i = \mathtt{not\_z}_i$ (because $\mathtt{not\_z}_i$ is the only assumption in $\mathcal{A}_P$ that has z_i as its contrary $(1 \leq i \leq m)$) and that there exists a rule $x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ z_1, \ldots, \mathtt{not}\ z_m$ in P.
"$\supseteq$": Let $x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ z_1, \ldots, \mathtt{not}\ z_m$ be a logic programming rule in P. Then from the definition of LP2ABA it follows that $\mathcal{R}_P$ contains a rule $x \leftarrow y_1, \ldots, y_n, \mathtt{not\_z}_1, \ldots, \mathtt{not\_z}_m$ with $\overline{\mathtt{not\_z}_i} = z_i$ $(1 \leq i \leq m)$. From the definition of ABA2LP this then implies that $\mathtt{ABA2LP}(\mathtt{LP2ABA}(P))$ contains a rule $x \leftarrow y_1, \ldots, y_n, \mathtt{not}\ z_1, \ldots, \mathtt{not}\ z_m$. ☐

Example 8. *Let P be the logic program from Example 2. The associated ABA framework* $\mathtt{LP2ABA}(P)$ *is* $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ *with* $\mathcal{A} = \{\mathtt{not\_a}, \mathtt{not\_b}, \mathtt{not\_c}, \mathtt{not\_d}, \mathtt{not\_e}, \mathtt{not\_f}\}$, $\mathcal{L} = \{a, b, c, d, e, f\} \cup \mathcal{A}$, $\mathcal{R} = \{a \leftarrow\ ;\ b \leftarrow a, \mathtt{not\_c}\ ;\ c \leftarrow \mathtt{not\_b}\ ;\ d \leftarrow b, e, \mathtt{not\_d}\ ;\ e \leftarrow a, \mathtt{not\_f}\ ;\ f \leftarrow \mathtt{not\_e}, \mathtt{not\_f}\}$, *and* $\overline{\mathtt{not\_a}} = a$, $\overline{\mathtt{not\_b}} = b$, *etc. The associated logic program of this ABA framework is then* $\mathtt{ABA2LP}(\mathtt{LP2ABA}(P)) = \{a \leftarrow\ ;\ b \leftarrow a, \mathtt{not}\ c\ ;\ c \leftarrow \mathtt{not}\ b\ ;\ d \leftarrow b, e, \mathtt{not}\ d\ ;\ e \leftarrow a, \mathtt{not}\ f\ ;\ f \leftarrow \mathtt{not}\ e, \mathtt{not}\ f\}$

4.2 Semantic Correspondence

Based on this syntactic equivalence, Theorem 6 and Theorem 7 point out that our results on semantic correspondence regarding the translation from ABA to LP, as stated in the previous section, can be reused for the translation from LP to ABA. Hence, our work generalizes the results by Schulz and Toni (2015) and Bondarenko et al. (1997), where only the LP to ABA direction is considered. In particular, the function Lab2Mod from Definition 12 can be applied to assumption labellings of an ABA framework $\mathcal{F}_P$, which is associated with a logic program P, to yield models of P. Conversely, Mod2Lab can be applied to models of a logic program P to yield assumption labellings of the associated ABA framework $\mathcal{F}_P$.

Theorem 6. *Let P be a logic program and let* $\mathcal{F}_P = \mathtt{LP2ABA}(P)$ *be its associated ABA framework. It holds that:*

1. *if $\mathcal{M}od$ is a 3-valued stable model of P, then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is a complete assumption labelling of* $\mathcal{F}_P$

2. *if $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}_P$, then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is a 3-valued stable model of P*

Proof. [12] Let $P_{\mathcal{F}_P}$ be the associated logic program of $\mathcal{F}_P$, i.e. $P_{\mathcal{F}_P} = \texttt{ABA2LP}(\mathcal{F}_P)$. From $\mathcal{F}_P = \texttt{LP2ABA}(P)$ it then follows that $P_{\mathcal{F}_P} = \texttt{ABA2LP}(\texttt{LP2ABA}(P))$. It then follows from Lemma 5 that $P_{\mathcal{F}_P} = P$.

1. Let $\mathcal{M}od$ be a 3-valued stable model of P. As $P = P_{\mathcal{F}_P}$, it directly follows that $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}_P}$. From Theorem 2 (point 2) it then follows that $\texttt{Mod2Lab}(\mathcal{M}od)$ is a complete assumption labelling of $\mathcal{F}_P$.

2. Let $\mathcal{L}ab$ be a complete assumption labelling of $\mathcal{F}_P$. From Theorem 2 (point 1) it then follows that $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a 3-valued stable model of $P_{\mathcal{F}_P}$. From the fact that $P_{\mathcal{F}_P} = P$, it then directly follows that $\texttt{Lab2Mod}(\mathcal{L}ab)$ is also a 3-valued stable model of P.

□

We now extend the correspondence results from Theorem 6 to common ABA and LP semantics.

Theorem 7. *Let P be a logic program and let $\mathcal{F}_P = \texttt{LP2ABA}(P)$ be its associated ABA framework. It holds that:*

1. *if $\mathcal{M}od$ is a well-founded model of P, then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is a grounded assumption labelling of* $\mathcal{F}_P$

2. *if $\mathcal{L}ab$ is a grounded assumption labelling of $\mathcal{F}_P$, then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is a well-founded model of P*

3. *if $\mathcal{M}od$ is a regular model of P, then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is a preferred assumption labelling of* $\mathcal{F}_P$

4. *if $\mathcal{L}ab$ is a preferred assumption labelling of $\mathcal{F}_P$, then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is a regular model of P*

5. *if $\mathcal{M}od$ is a (2-valued) stable model of P, then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is a stable assumption labelling of* $\mathcal{F}_P$

6. *if $\mathcal{L}ab$ is a stable assumption labelling of $\mathcal{F}_P$, then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is a (2-valued) stable model of P*

7. *if $\mathcal{M}od$ is an ideal model of P, then* $\texttt{Mod2Lab}(\mathcal{M}od)$ *is an ideal assumption labelling of* $\mathcal{F}_P$

12. This was proven by Schulz and Toni (2015) for a simplified version of `Mod2Lab` and `Lab2Mod`. Here, we use our results from Section 3 for the proof which also serves as an illustration for the proofs of Theorem 7.

8. *if $\mathcal{L}ab$ is an ideal assumption labelling of $\mathcal{F}_P$, then* $\texttt{Lab2Mod}(\mathcal{L}ab)$ *is an ideal model of* P

Proof. Let $P_{\mathcal{F}_P}$ be the associated logic program of $\mathcal{F}_P$, i.e. $P_{\mathcal{F}_P} = \texttt{ABA2LP}(\mathcal{F}_P)$. From $\mathcal{F}_P = \texttt{LP2ABA}(P)$ it then follows that $P_{\mathcal{F}_P} = \texttt{ABA2LP}(\texttt{LP2ABA}(P))$. It then follows from Lemma 5 that $P_{\mathcal{F}_P} = P$.

1. Let $\mathcal{M}od$ be a well-founded model of P. As $P = P_{\mathcal{F}_P}$, it directly follows that $\mathcal{M}od$ is a well-founded model of $P_{\mathcal{F}_P}$. From point 2 of Theorem 4 it follows that $\texttt{Mod2Lab}(\mathcal{M}od)$ is a grounded assumption labelling of $\mathcal{F}_P$.

2. Let $\mathcal{L}ab$ be a grounded labelling of $\mathcal{F}_P$. From point 1 of Theorem 4 it then follows that $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a well-founded model of $P_{\mathcal{F}_P}$. From the fact that $P_{\mathcal{F}_P} = P$, it then directly follows that $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a well-founded model of P.

3. Similar to point 1, but using point 4 of Theorem 4.

4. Similar to point 2, but using point 3 of Theorem 4.

5. Similar to point 1, but using point 6 of Theorem 4.

6. Similar to point 2, but using point 5 of Theorem 4.

7. Similar to point 1, but using point 8 of Theorem 4.

8. Similar to point 2, but using point 7 of Theorem 4.

□

The reader can verify that the common LP semantics of the logic program P from Example 2 correspond to the common ABA semantics of the associated ABA framework $\mathcal{F}_P$ given in Example 8.

If it is possible to reuse our results from the ABA to LP translation for the LP to ABA translation, then would the reverse also be possible? In other words, is it possible to reuse some of the existing work on the LP to ABA translation (e.g., Bondarenko et al., 1997; Schulz & Toni, 2015) to obtain similar results for the ABA to LP translation? The short answer is no, at least not in any obvious way. Our ability to reuse the results from the ABA to LP translation for the LP to ABA translation critically depends on the fact that $\texttt{ABA2LP}(\texttt{LP2ABA}(P)) = P$ as stated in Lemma 5. To reuse the results of the LP to ABA direction for the ABA to LP direction would thus require the property that for any ABA framework $\mathcal{F}$, $\texttt{LP2ABA}(\texttt{ABA2LP}(\mathcal{F})) = \mathcal{F}$. However, this property does *not* hold, since in the translation from ABA to LP some information gets lost (like the precise set of assumptions, some of which may not occur in any rule) as in essence only its set of rules $\mathcal{R}$ gets translated.

Example 9. *Consider the ABA framework $\mathcal{F}$ from Example 1 and its associated logic program* $\texttt{ABA2LP}(\mathcal{F}) = P_{\mathcal{F}}$*, which is the logic program from Example 2. Translating $P_{\mathcal{F}}$ back to an ABA framework yields* $\texttt{LP2ABA}(P_{\mathcal{F}}) = \mathcal{F}'$*, where $\mathcal{F}'$ is the ABA framework introduced as* $\texttt{LP2ABA}(P)$ *in Example 8. The most obvious difference between $\mathcal{F}$ and $\mathcal{F}'$ is in the set of assumptions. Even though* `not_b` *corresponds to the original assumption β, and* `not_c` *to γ,* not_a *does not correspond to the original assumption* α *since they have different contraries. Furthermore,* $\mathcal{F}'$ *is missing one of the rules from* $\mathcal{F}$*, and the fact that two of the assumptions (*α *and* ϕ*) originally had the same contrary is lost. Furthermore, the non-used sentences (*l *and* λ*) in the language of* $\mathcal{F}$ *are not present in* $\mathcal{F}'$*.*

Furthermore, the two functions Lab2Mod and Mod2Lab, which convert between assumption labellings and LP models for ABA frameworks and their associated logic programs (as introduced in Section 3), can be directly applied to logic programs and their associated ABA frameworks (see Theorem 6). In contrast, functions converting between LP and ABA semantics for logic programs and their associated ABA frameworks (as introduced in existing work (e.g., Bondarenko et al., 1997; Schulz & Toni, 2015) cannot be applied the other way around (i.e. for ABA frameworks and their associated logic programs), as shown in the following.

We recall the conversion functions by Schulz and Toni (2015) (applied to an ABA framework and its associated logic program), which are in fact special cases of the functions we give in Definition 12.

Definition 14. *Let* $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ *be a normal ABA framework and let* $P_\mathcal{F}$ *be the associated logic program. Given an assumption labelling* $\mathcal{L}ab$ *of* $\mathcal{F}$*, the function* $\mathtt{Lab2Mod}'$ *yields the 3-valued interpretation* $\langle T, F \rangle$ *where* $T = \{\overline{\chi} \mid \chi \in \text{OUT}(\mathcal{L}ab)\}$ *and* $F = \{\overline{\chi} \mid \chi \in \text{IN}(\mathcal{L}ab)\}$*. Given a 3-valued interpretation* $\langle T, F \rangle$ *of* $P_\mathcal{F}$*, the function* $\mathtt{Mod2Lab}'$ *yields an assumption labelling* $\mathcal{L}ab$ *of* $\mathcal{F}$ *with* $\text{IN}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in F\}$*,* $\text{OUT}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in T\}$ *and* $\text{UNDEC}(\mathcal{L}ab) = \{\chi \in \mathcal{A} \mid \overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T \cup F)\}$*.*

These translations between assumption labellings and LP models do not preserve the ABA and LP semantics as is the case when using Mod2Lab and Lab2Mod, i.e. Theorems 2 and 4 does not hold when applying $\mathtt{Mod2Lab}'$ or $\mathtt{Lab2Mod}'$.

Example 10. *Consider the ABA framework* $\mathcal{F}$ *from Example 4 and its associated logic program* $P_\mathcal{F}$ *from Example 2.* $\mathtt{Lab2Mod}'(\mathcal{L}ab_1) = \langle \emptyset, \{l\} \rangle$*, which is not a 3-valued stable model of* $P_\mathcal{F}$*. Similarly,* $\mathtt{Mod2Lab}'(\mathcal{M}od_1) = \langle \emptyset, \emptyset, \{\alpha, \beta, \gamma, \delta, \epsilon, \phi\} \rangle$ *is not a complete assumption labelling of* $\mathcal{F}$*.*

Thus, existing results on the correspondence between ABA and LP semantics when translating LP to ABA cannot be applied to the translation from ABA to LP.

5. Going Beyond Normal ABA Frameworks and Common Semantics

So far, we have only studied a special type of ABA frameworks, namely flat ABA frameworks where each assumption has a single contrary which is not an assumption itself. Furthermore, we have only considered common ABA semantics, that is complete, grounded, preferred, stable, and ideal semantics. In the current section, we briefly examine what happens when we try to go beyond these restrictions, that is, if we use either ABA frameworks that are not normal or ABA semantics that are not common. Note that the translation from a logic program to an ABA framework always yields a normal ABA framework, so we only consider the translation from an ABA framework which is not normal to a logic program.

5.1 Beyond Normal ABA Frameworks

The most obvious way to go beyond normal ABA frameworks is to consider the type of ABA frameworks used in most of the literature, namely *flat* ABA frameworks, i.e. to drop the restriction that the contrary of an assumption has to be a non-assumption. Using `ABA2LP` to translate such an ABA framework to a logic program can result in NAF literals of the form $\texttt{not}\ \alpha$, where α is an assumption. Since there is no rule with head α in the ABA framework, and thus neither in the associated logic program, $\texttt{not}\ \alpha$ will always be satisfied, which does not always reflect the situation in the underlying ABA framework.

Example 11. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be the ABA framework with $\mathcal{A} = \{\alpha, \beta\}$, $\mathcal{L} = \{a, b\} \cup \mathcal{A}$, $\overline{\alpha} = \beta$, $\overline{\beta} = a$, and $\mathcal{R} = \{a \leftarrow \alpha;\ b \leftarrow \beta\}$. This ABA framework has three complete assumption labellings: $\langle \emptyset, \emptyset, \{\alpha, \beta\} \rangle$, $\langle \{\alpha\}, \{\beta\}, \emptyset \rangle$, and $\langle \{\beta\}, \{\alpha\}, \emptyset \rangle$. Translating $\mathcal{F}$ to a logic program using* `ABA2LP` *yields $P_{\mathcal{F}} = \{a \leftarrow \texttt{not}\ \beta;\ b \leftarrow \texttt{not}\ a\}$, where $HB_{P_{\mathcal{F}}} = \{a, b, \beta\}$, so an assumption is now an atom in the Herbrand Base of the associated logic program. $P_{\mathcal{F}}$ has only one 3-valued stable model, namely $\langle \{a\}, \{b, \beta\} \rangle$, which corresponds to the second complete assumption labelling in terms of* `Lab2Mod` *and* `Mod2Lab`*. However, there is no corresponding 3-valued stable model for the other two complete assumption labellings.*

We now show that a flat ABA framework (where contraries can be assumptions) can always be translated to an equivalent normal ABA framework, that is, to a normal ABA framework that has the same complete (respectively grounded, preferred, stable and ideal) labellings as the flat ABA framework. The idea is that every assumption that has an assumption as its contrary gets a new non-assumption as its contrary. A rule is then added with the new contrary as its head and the old contrary as its body.

Definition 15. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework where contraries are allowed to be assumptions or non-assumptions (that is: $^{-} : \mathcal{A} \to \mathcal{L}$). The* corresponding normal ABA framework *is $\mathcal{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}, ^{\sim} \rangle$ with $\mathcal{L}' = \mathcal{L} \cup \{\overline{\chi}^* \mid \chi \in \mathcal{A}$ and $\overline{\chi} \in \mathcal{A}\}$, $\mathcal{R}' = \mathcal{R} \cup \{\overline{\chi}^* \leftarrow \overline{\chi} \mid \chi \in \mathcal{A}$ and $\overline{\chi} \in \mathcal{A}\}$ and $\tilde{\chi}$ being $\overline{\chi}$ if $\overline{\chi} \in \mathcal{L} \setminus \mathcal{A}$ or $\overline{\chi}^*$ if $\overline{\chi} \in \mathcal{A}$.*

The following theorem shows that the contrary of an assumption has corresponding arguments in $\mathcal{F}$ and $\mathcal{F}'$. To distinguish arguments constructed from different ABA frameworks, we use the notation $Asms \vdash_{\mathcal{F}} \overline{\chi}$ to denote that the argument is constructed from the ABA framework $\mathcal{F}$.

Theorem 8. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework where contraries are allowed to be assumptions or non-assumptions and let $\mathcal{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}, ^{\sim} \rangle$ be the corresponding normal ABA framework.*
It holds that:

1. *no assumption $\chi \in \mathcal{A}$ has $\tilde{\chi} \in \mathcal{A}$*
2. *let $\chi \in \mathcal{A}$ and $Asms \subseteq \mathcal{A}$. $Asms \vdash_{\mathcal{F}} \overline{\chi}$ iff $Asms \vdash_{\mathcal{F}'} \tilde{\chi}$.*

Proof. See Appendix A. □

Since complete assumption labellings only depend on the arguments for the contrary of an assumption in question, it follows that the assumption labellings of $\mathcal{F}$ and $\mathcal{F}'$ are the same.

Corollary 1. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an ABA framework where contraries are allowed to be assumptions or non-assumptions and let $\mathcal{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}, \tilde{} \rangle$ be the corresponding normal ABA framework. Then $\mathcal{L}ab$ is a complete (respectively grounded, preferred, stable or ideal) assumption labelling of $\mathcal{F}$ iff it is a complete (respectively grounded, preferred, stable or ideal) assumption labelling of $\mathcal{F}'$.*

Thus, given that any flat ABA framework can be translated to a corresponding normal ABA framework, which can then be translated to a logic program by means of `ABA2LP`, it holds that the 3-valued stable (respectively well-founded, regular, (2-valued) stable, ideal) models of the resulting logic program coincide (in terms of `Mod2Lab` and `Lab2Mod`) with the complete (respectively grounded, preferred, stable, ideal) assumption labellings of the flat ABA framework. Hence, the results presented in this paper hold not just for normal ABA frameworks, but also for flat ABA frameworks in general.

Observation 1. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be a flat ABA framework, $\mathcal{F}'$ the corresponding normal ABA framework, and $P_{\mathcal{F}'}$ the associated logic program of $\mathcal{F}'$.*

1. *if $\mathcal{L}ab$ is a complete (respectively grounded, preferred, stable, ideal) assumption labelling of $\mathcal{F}$ then* `Lab2Mod`*($\mathcal{L}ab$) is a 3-valued stable (respectively well-founded, regular, (2-valued) stable, ideal) model of $P_{\mathcal{F}'}$.*
2. *if $\mathcal{M}od$ is a 3-valued stable (respectively well-founded, regular, (2-valued) stable, ideal) model of $P_{\mathcal{F}'}$ then* `Mod2Lab`*($\mathcal{M}od$) is a complete (respectively grounded, preferred, stable, ideal) assumption labelling of $\mathcal{F}$.*

Example 12. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be the ABA framework from Example 11. The corresponding normal ABA framework is $\mathcal{F}' = \langle \mathcal{L}, \mathcal{R}', \mathcal{A}, \tilde{} \rangle$ with $\mathcal{L}' = \mathcal{L} \cup \{\beta^*\}$, $\mathcal{R}' = \mathcal{R} \cup \{\beta^* \leftarrow \beta\}$, and $\tilde{\alpha} = \beta^*$ and $\tilde{\beta} = a$. $\mathcal{F}'$ has the same three complete assumption labellings as $\mathcal{F}$ (see Example 11).*

Translating $\mathcal{F}'$ to a logic program using `ABA2LP` *yields $P_{\mathcal{F}'} = \{a \leftarrow \texttt{not}\ \beta^*;\ b \leftarrow \texttt{not}\ a;\ \beta^* \leftarrow \texttt{not}\ a\}$, where $HB_{P_{\mathcal{F}'}} = \{a, b, \beta^*\}$. $P_{\mathcal{F}'}$ has three 3-valued stable models, which correspond to the three complete assumption labellings of $\mathcal{F}'$ in terms of* `Lab2Mod` *and* `Mod2Lab`*: $\langle \emptyset, \emptyset \rangle$, $\langle \{a\}, \{b, \beta^*\} \rangle$, $\langle \{b, \beta^*\}, \{a\} \rangle$.*

Another way to go beyond normal ABA frameworks is to allow assumptions that have a set of contraries (e.g., Gaertner & Toni, 2007, 2008; Fan & Toni, 2014, 2015), instead of just a single contrary (e.g., Bondarenko et al., 1997; Dung et al., 2009, 2007; Toni, 2014; Schulz & Toni, 2014). We will refer to these kind of ABA frameworks as *multiple contraries ABA frameworks*, to distinguish them from the *single contrary ABA frameworks* we have studied so far. It turns out that a multiple contraries ABA framework can always be translated to an equivalent single contrary ABA framework, using a procedure first described by Gaertner and Toni (2008). The idea is that for each assumption α with $\overline{\alpha} = \{a_1, \ldots, a_n\}$ a new sentence a^* is added to $\mathcal{L}$ as well as a new rule $a^* \leftarrow a_i$ for each $i \in \{1, \ldots, n\}$ to $\mathcal{R}$, and $\overline{\alpha}$ is set to a^*. The resulting ABA framework is clearly single contrary. Furthermore, for every assumption α, there exists an ABA argument $Asms \vdash a_i$ for some $a_i \in \overline{\alpha}$ in the multiple contraries ABA framework iff there exists an ABA argument $Asms \vdash a$ with $a = \overline{\alpha}$ in the translated single contrary ABA framework. This implies that the assumption labellings of the two ABA frameworks are the same. Thus, in order to translate a multiple contraries ABA framework to a logic program, the ABA framework can first be translated to a single contrary ABA framework, which is subsequently translated to the associated logic program using `ABA2LP`. Since the semantics of the multiple contraries and the single contrary ABA frameworks are the same, the semantics of the multiple contraries ABA framework and the associated logic program coincide as stated in Theorems 2 and 4.

When it comes to broadening our results to more general classes of ABA frameworks, the only remaining constraint is that the ABA framework needs to be flat. If we were to consider non-flat ABA frameworks, that is, ABA frameworks where the head of a rule can be an assumption, then the translation to logic programming could yield a logic program where the head of a rule can be a NAF literal. Clearly, this would go beyond the syntax of a normal logic program, and many of the LP semantics that were applied in the current paper have simply not been defined in the context of such a non-normal logic program. Overall, we therefore observe that our results hold for flat ABA frameworks only. We do want to emphasize, however, that most of the recent research on ABA is restricted to flat ABA frameworks (Dung et al., 2006, 2007, 2009; Toni, 2013; Fan & Toni, 2014).

5.2 Beyond Common ABA Semantics

Apart from going beyond normal ABA frameworks, one could also examine what happens when going beyond common ABA semantics. Take for instance the semi-stable semantics (Verheij, 1996; Caminada, 2006), which has recently been formulated in the context of ABA (Caminada et al., 2015b; Schulz & Toni, 2015).

Definition 16 (Schulz & Toni, 2015)**.** *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ be an ABA framework. We say that an assumption labelling $\mathcal{L}ab$ is a* semi-stable assumption labelling *of $\mathcal{F}$ iff $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$ where* $\text{UNDEC}(\mathcal{L}ab)$ *is minimal among all complete assumption labellings of $\mathcal{F}$.*

It has been shown that there exists a one-to-one relationship between semi-stable assumption labellings and semi-stable assumption extensions in the sense defined by Caminada et al. (2015b), with the set of IN labelled assumptions in a semi-stable assumption labelling constituting a semi-stable assumption extension. It has also been observed that semi-stable semantics for ABA behaves in a way that is very similar to semi-stable semantics for abstract argumentation (Caminada et al., 2015b). For instance, each stable assumption labelling (extension) is also a semi-stable assumption labelling (extension), and each semi-stable assumption labelling (extension) is also a preferred assumption labelling (extension). Moreover, for ABA frameworks that have at least one stable assumption labelling (extension), it holds that each semi-stable assumption labelling (extension) is also a stable assumption labelling (extension).

In the context of logic programming, a concept somewhat similar to semi-stable semantics exists under the name of L-stable semantics (Eiter et al., 1997).

Definition 17. *Let P be a logic program. We say that a 3-valued interpretation $\langle T, F \rangle$ of P is an* L-stable model *of P iff it is a 3-valued stable model of P where $T \cup F$ is maximal among all 3-valued stable models of P.*

So where a semi-stable assumption labelling tries to minimize the set of assumptions that are not labelled IN or OUT, an L-stable model tries to minimize the set of atoms that are not in T or F. Due to the similar formulation of these semantics and the fact that L-stable models of a logic program coincide with semi-stable assumption labellings of the associated ABA framework (Schulz & Toni, 2015), one might expect that the semi-stable assumption labellings of an ABA framework $\mathcal{F}$ also coincide with the L-stable models of the associated logic program $P_{\mathcal{F}}$. However, this turns out not to be the case, as is illustrated by the following example.

Example 13. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ be the ABA framework with $\mathcal{A} = \{\alpha, \beta, \gamma\}$, $\mathcal{L} = \mathcal{A} \cup \{a, b, c, d\}$, $\overline{\alpha} = a$, $\overline{\beta} = b$, and $\overline{\gamma} = c$ and $\mathcal{R} = \{a \leftarrow \beta;\ b \leftarrow \alpha;\ c \leftarrow \gamma;\ d \leftarrow b, c\}$. $\mathcal{F}$ has three complete assumption labellings: $\mathcal{L}ab_1 = \langle \emptyset, \emptyset, \{\alpha, \beta, \gamma\} \rangle$, $\mathcal{L}ab_2 = \langle \{\alpha\}, \{\beta\}, \{\gamma\} \rangle$, and $\mathcal{L}ab_3 = \langle \{\beta\}, \{\alpha\}, \{\gamma\} \rangle$. The last two of these are semi-stable assumption labellings. The associated logic program $P_{\mathcal{F}}$ is $\{a \leftarrow \texttt{not}\ b;\ b \leftarrow \texttt{not}\ a;\ c \leftarrow \texttt{not}\ c;\ d \leftarrow b, c\}$. $P_{\mathcal{F}}$ has three 3-valued stable models: $\mathcal{M}od_1 = \langle \emptyset, \emptyset \rangle$, $\mathcal{M}od_2 = \langle \{b\}, \{a\} \rangle$ and $\mathcal{M}od_3 = \langle \{a\}, \{b, d\} \rangle$. Only the last one is an L-stable model. Hence, we have that the semi-stable assumption labellings of $\mathcal{F}$ do not coincide with the L-stable models of $P_{\mathcal{F}}$.*

Although Example 13 illustrates that in general ABA under semi-stable semantics does not coincide with logic programming under L-stable semantics, equivalence is restored when the ABA framework is *assumption-spanning*, i.e. if each non-assumption is the contrary of some assumption.

Definition 18. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ be an ABA framework. We say that $\mathcal{F}$ is* assumption-spanning *iff for each $x \in \mathcal{L} \setminus \mathcal{A}$ there exists a $\chi \in \mathcal{A}$ such that $\overline{\chi} = x$.*

We first observe that the ABA framework of Example 13 is not assumption-spanning, because there is no assumption that has d as its contrary. In the following example, we make the ABA framework from Example 13 assumption-spanning.

Example 14. *Let $\mathcal{F}' = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ be the assumption-spanning ABA framework with $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$, $\mathcal{L} = \mathcal{A} \cup \{a, b, c, d\}$, $\overline{\alpha} = a$, $\overline{\beta} = b$, $\overline{\gamma} = c$, $\overline{\delta} = d$ and $\mathcal{R} = \{a \leftarrow \beta;\ b \leftarrow \alpha;\ c \leftarrow \gamma;\ d \leftarrow b, c\}$. $\mathcal{F}'$ has three complete assumption labellings: $\langle \emptyset, \emptyset, \{\alpha, \beta, \gamma, \delta\} \rangle$, $\langle \{\alpha\}, \{\beta\}, \{\gamma, \delta\} \rangle$, and $\langle \{\beta, \delta\}, \{\alpha\}, \{\gamma\} \rangle$. Only the last one is a semi-stable assumption labelling. The associated logic program $P_{\mathcal{F}'}$ is $\{a \leftarrow \texttt{not}\ b;\ b \leftarrow \texttt{not}\ a;\ c \leftarrow \texttt{not}\ c;\ d \leftarrow b, c\}$, which is the same as $P_{\mathcal{F}}$ from Example 13. $P_{\mathcal{F}'}$ thus has the same three 3-valued stable models as $P_{\mathcal{F}}$: $\langle \emptyset, \emptyset \rangle$, $\langle \{b\}, \{a\} \rangle$ and $\langle \{a\}, \{b, d\} \rangle$. Again, only the last one is an L-stable model. Hence, we observe that when we make the ABA framework $\mathcal{F}$ (from Example 13) assumption-spanning (by adding an assumption δ with $\overline{\delta} = d$) semi-stable assumption labellings of the resulting ABA framework $\mathcal{F}'$ coincide with the L-stable models of the associated logic program $P_{\mathcal{F}'}$.*

The fact that for an assumption-spanning ABA framework $\mathcal{F}$, the semi-stable assumption labellings of $\mathcal{F}$ coincide with the L-stable models of $P_{\mathcal{F}}$ is not restricted to the particular ABA framework of Example 14, but holds in general. To prove this, we need the following lemma.

Lemma 9. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an assumption-spanning normal ABA framework, and let $P_\mathcal{F}$ be its associated logic program. Let $\mathcal{L}ab_1$ and $\mathcal{L}ab_2$ be complete assumption labellings of $\mathcal{F}$, let $\mathcal{M}od_1 = \langle T_1, F_1 \rangle$ be $\texttt{Lab2Mod}(\mathcal{L}ab_1)$ and let $\mathcal{M}od_2 = \langle T_2, F_2 \rangle$ be $\texttt{Lab2Mod}(\mathcal{L}ab_2)$. It holds that $\text{UNDEC}(\mathcal{L}ab_1) \subseteq \text{UNDEC}(\mathcal{L}ab_2)$ iff $HB_{P_\mathcal{F}} \setminus (T_1 \cup F_1) \subseteq HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)$.*

Proof. "$\Rightarrow$": Suppose $\text{UNDEC}(\mathcal{L}ab_1) \subseteq \text{UNDEC}(\mathcal{L}ab_2)$. Let $x \in HB_{P_\mathcal{F}} \setminus (T_1 \cup F_1)$. From the fact that $\mathcal{F}$ is assumption-spanning, it follows that there exists an assumption χ with $\overline{\chi} = x$. As $\mathcal{L}ab_1 = \texttt{Mod2Lab}(\mathcal{M}od_1)$ (Theorem 2, point 3) it follows from the definition of $\texttt{Mod2Lab}$ that $\chi \in \text{UNDEC}(\mathcal{L}ab_1)$. From the fact that $\text{UNDEC}(\mathcal{L}ab_1) \subseteq \text{UNDEC}(\mathcal{L}ab_2)$ it then follows that $\chi \in \text{UNDEC}(\mathcal{L}ab_2)$. As $\mathcal{L}ab_2 = \texttt{Mod2Lab}(\mathcal{M}od_2)$ (Theorem 2, point 3) it follows that $\text{UNDEC}(\mathcal{L}ab_2) = \{\zeta \in \mathcal{A} \mid \overline{\zeta} \in HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)\}$, so from $\chi \in \text{UNDEC}(\mathcal{L}ab_2)$ it follows that $x \in HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)$.
"$\Leftarrow$": Suppose that $HB_{P_\mathcal{F}} \setminus (T_1 \cup F_1) \subseteq HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)$. Let $\chi \in \text{UNDEC}(\mathcal{L}ab_1)$. As $\mathcal{L}ab_1 = \texttt{Mod2Lab}(\mathcal{M}od_1)$ (Theorem 2, point 3) it follows that $\text{UNDEC}(\mathcal{L}ab_1) = \{\zeta \in \mathcal{A} \mid \overline{\zeta} \in HB_{P_\mathcal{F}} \setminus (T_1 \cup F_1)\}$ so from $\chi \in \text{UNDEC}(\mathcal{L}ab_1)$ it follows that $\overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T_1 \cup T_2)$. From $HB_{P_\mathcal{F}} \setminus (T_1 \cup F_1) \subseteq HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)$ it then follows that $\overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T_2 \cup F_2)$. As $\mathcal{L}ab_2 = \texttt{Mod2Lab}(\mathcal{M}od_2)$ (Theorem 2, point 3) it follows that $\chi \in \text{UNDEC}(\mathcal{L}ab_2)$. ☐

Note that Lemma 9 critically relies on the ABA framework being assumption-spanning. For instance, in Example 13 $\text{UNDEC}(\mathcal{L}ab_2) \subseteq \text{UNDEC}(\mathcal{L}ab_3)$, but $HB_{P_\mathcal{F}} \setminus T_2 \cup F_2 \nsubseteq HB_{P_\mathcal{F}} \setminus T_3 \cup F_3$, as $\mathcal{F}$ is not assumption-spanning.

Lemma 9 allows us to make the connection between the semi-stable assumption labellings of an ABA framework and the L-stable models of its associated logic program (as long as the ABA framework is assumption-spanning).

Theorem 10. *Let $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^{-} \rangle$ be an assumption-spanning normal ABA framework, and let $P_\mathcal{F}$ be its associated logic program. It holds that:*

1. *if $\mathcal{L}ab$ is a semi-stable assumption labelling of $\mathcal{F}$ then $\texttt{Lab2Mod}(\mathcal{L}ab)$ is an L-stable model of $P_\mathcal{F}$*
2. *if $\mathcal{M}od$ is an L-stable model of $P_\mathcal{F}$ then $\texttt{Mod2Lab}(\mathcal{M}od)$ is a semi-stable assumption labelling of $\mathcal{F}$*

Proof. See Appendix A ☐

Notice that when translating a logic program to an ABA framework (using $\texttt{LP2ABA}$) the resulting ABA framework is always assumption-spanning. This allows us to obtain the following result.

Theorem 11. *Let P be a logic program and let $\mathcal{F}_P - \texttt{LP2ABA}(P)$ be its associated ABA framework. It holds that:*

1. *if $\mathcal{M}od$ is an L-stable model of P then $\texttt{Mod2Lab}(\mathcal{M}od)$ is a semi-stable assumption labelling of $\mathcal{F}_P$*
2. *If $\mathcal{L}ab$ is a semi-stable assumption labelling of $\mathcal{F}_P$ then $\texttt{Lab2Mod}(\mathcal{L}ab)$ is an L-stable model of P*

Proof. Let $P_{\mathcal{F}_P}$ be the associated logic program of $\mathcal{F}_P$, i.e. $P_{\mathcal{F}_P} = \mathtt{ABA2LP}(\mathcal{F}_P)$. From $\mathcal{F}_P = \mathtt{LP2ABA}(P)$ it follows that $P_{\mathcal{F}_P} = \mathtt{ABA2LP}(\mathtt{LP2ABA}(P))$. It then follows from Lemma 5 that $P_{\mathcal{F}_P} = P$.

1. Let $\mathcal{M}od$ be an L-stable model of P. As $P = P_{\mathcal{F}_P}$ it directly follows that $\mathcal{M}od$ is an L-stable model of $P_{\mathcal{F}_P}$. As $\mathcal{F}_P$ is an assumption-spanning normal ABA framework, it holds that (Theorem 10 point 2) $\mathtt{Mod2Lab}(\mathcal{M}od)$ is a semi-stable labelling of $\mathcal{F}_P$.

2. Let $\mathcal{L}ab$ be a semi-stable assumption labelling of $\mathcal{F}_P$. As $\mathcal{F}_P$ is an assumption-spanning normal ABA framework, it holds that (Theorem 10 point 1) $\mathtt{Lab2Mod}(\mathcal{L}ab)$ is an L-stable model of $P_{\mathcal{F}_P}$. As $P_{\mathcal{F}_P} = P$, it follows that $\mathtt{Lab2Mod}(\mathcal{L}ab)$ is an L-stable model of P.

□

Hence, the results by Schulz and Toni (2015, Thms. 4, 5) are subsumed by our results.

6. Discussion

In the current paper we re-examined the relationship between ABA and LP and found that the most frequently studied fragment of ABA, namely flat ABA frameworks, does not only subsume normal logic programming as previously shown (Bondarenko et al., 1997; Toni, 2007, 2008; Schulz & Toni, 2015), but that it is in fact also subsumed by normal logic programming itself through a straightforward translation. That is, flat ABA can be seen as LP with a slightly different syntax since the outcome that is yielded by a flat ABA framework under common ABA semantics (that is: complete, grounded, preferred, stable and ideal) is the same as the outcome yielded by its (syntactically nearly identical) associated normal logic program under common LP semantics, and vice versa. The only difference is that in ABA the outcome is defined in terms of assumptions (which correspond to NAF literals in the associated logic program) whereas in logic programming the outcome is defined in terms of *all* the literals in the logic program (NAF as well as non-NAF). However, since the status of the non-NAF literals is determined solely by the status of the NAF-literals, flat ABA and normal LP are semantically equivalent. Importantly, correspondence for our novel translation does neither follow from existing results on the correspondence of ABA and LP semantics nor from correspondence results between ABA and AA and between AA and LP.

Although most of our formal results are restricted to flat ABA frameworks under common ABA semantics, their applicability is broader than that. For instance, as was explained in Section 5, it is always possible to translate a multiple contraries ABA framework to an equivalent single contrary ABA framework. Similarly, one could weaken the restriction that the semantics has to be *common* and for instance apply semi-stable semantics. As was observed in Section 5, for ABA frameworks that are assumption-spanning (i.e. every non-assumption is the contrary of some assumption), ABA is still subsumed by logic programming, even under semi-stable semantics. Overall, our results on translating ABA to LP combined with existing ones on translating LP to ABA yield *equivalence* between (flat) ABA and (normal) logic programming under every ABA semantics studied here.

This equivalence allows not only to carry over techniques developed in the context of ABA to the context of LP, but also to apply LP techniques for ABA, e.g. to apply the computation of LP semantics for finding ABA labellings. Moreover, recent results on the equivalence between ASPIC+ and ABA (Heyninck & Straßer, 2016) (which show that ASPIC+ without argument preferences is subsumed by ABA) can be reapplied through our translation from ABA to logic programming to show that ASPIC+ (without preferences) is subsumed by logic programming.

A by-product of our work concerns the ideal semantics for ABA. When the ideal semantics was first introduced for ABA (Dung et al., 2007), the authors were inspired by the idea of the ideal scenario semantics for LP. Our correspondence results between ideal assumption labellings and ideal models, along with the correspondence between ideal models and ideal scenarios as proven in Appendix B, imply that the ideal semantics for ABA in fact *coincides* with ideal scenario semantics for LP.[13]

We chose to apply the assumption labelling semantics (Schulz & Toni, 2014, 2017) for ABA instead of the assumption extension semantics (Bondarenko et al., 1997) since it classifies assumptions as accepted (IN), unaccepted (OUT), and neither accepted nor unaccepted (UNDEC), rather than only as accepted and not accepted as done by assumption extensions. This yields a straightforward relation with the three values a literal can be assigned by a logic programming model.

Although (flat) ABA and (normal) logic programming are technically equivalent, there is an important conceptual difference between them. Logic programming over the years has evolved mostly into a formalism for "constraint satisfaction", by expanding the (2-valued) stable model semantics for normal logic programs to answer sets for logic programs with strong negation, disjunction, and other constructs (so-called answer set programs), as evidenced by the current popularity of answer set programming (e.g., Brewka, Eiter, & Truszczynski, 2011; Calimeri, Ianni, Krennwallner, & Ricca, 2012; Calimeri, Gebser, Maratea, & Ricca, 2016; Gebser, Kaufmann, Kaminski, Ostrowski, Schaub, & Schneider, 2011; Alviano, Dodaro, Leone, & Ricca, 2015; Leone, Pfeifer, Faber, Eiter, Gottlob, Perri, & Scarcello, 2006; Liu, Janhunen, & Niemelä, 2012; Lin & Zhao, 2004). The idea is for a particular problem (say, a sudoku puzzle) to be represented as an answer set program, so that the resulting answer sets correspond to the solutions of the original problem. Also, if the problem is such that no solutions exist, one would like to obtain no answer sets either. This helps to explain the current popularity of (2-valued) stable model (or answer set) semantics, as (unlike 3-valued stable, well-founded, regular, ideal or L-stable) it allows for the absence of models. ABA, on the other hand, is concerned not so much with "constraint satisfaction", but with reasoning using rules of thumb or with other forms of imperfect information, e.g. to model human-style reasoning and dialogue in artificial agents (e.g., Dung & Thang, 2009; Craven et al., 2012; Toni, 2012; Fan & Toni, 2014; Toni, 2014). In this context, one would like to avoid having small imperfections causing a total collapse of all entailment (the absence of any model or labelling). Hence, one would like to go beyond

13. The only related work that we are aware of regarding the ideal semantics in LP and argumentation is by Nieves and Osorio (2016), who translate abstract argumentation frameworks to logic programs and define two different types of ideal semantics for logic programs. However, their definition of ideal models is completely different from ours and their translation from argumentation to LP is a lot more high-level and explicitly encodes acceptability of arguments in the logic program.

stable semantics, and apply for instance complete, grounded, preferred, ideal or semi-stable semantics instead. Hence, although ABA and LP are technically equivalent, they differ in the types of problems that they aim to deal with, as well as in the semantics that are most appropriate for doing so.

There are two major lines for future work. Firstly, it will be interesting to see which applications benefit most from our findings, in other words, which applications of both LP and ABA benefit most when applying results from the other formalism. ABA methods have already been successfully applied to LP, e.g. for explaining (Schulz & Toni, 2016) as well as visualizing (Schulz, 2015) logic programs under certain semantics. LP methods which may prove useful for ABA include computational methods (e.g., Ruiz & Minker, 1994; Janhunen, Niemelä, Seipel, Simons, & You, 2006; Gebser et al., 2011). Secondly, we are planning to investigate extensions of ABA and LP, for example ABA and LP with preferences (Toni, 2008; Cyras & Toni, 2016; Greco, Trubitsyna, & Zumpano, 2007; Heyninck, Pardo, & Straßer, 2017), or non-flat ABA frameworks (Bondarenko et al., 1997).

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC, UK), grant ref. EP/J012084/1 (SAsSy project).

Appendix A. Proofs

Proposition 2

Proof. $\Phi_{Atms}(P)$ is the least fixed point of an "immediate consequence operator" Ψ and $\Phi_{Atms}(P)$ can be obtained by iterating Ψ ω times, starting with the 3-valued interpretation $\langle \emptyset, Atms \rangle$ (Przymusinski, 1990). We recall the definition of Ψ (Przymusinski, 1990).
For a 3-valued interpretation $\langle T, F \rangle$ w.r.t. $Atms \supseteq HB_P$, $\Psi(\langle T, F \rangle) = \langle T', F' \rangle$ is the 3-valued interpretation w.r.t $Atms$ such that for every $x \in Atms$ it holds that:

- $x \in T'$ if there exists a rule $x \leftarrow y_1, \ldots, y_n$ in P such that $\forall i \in \{1, \ldots, n\} : y_i \in T$ or $y_i = \texttt{TRUE}$;
- $x \notin T' \cup F'$ if
  - there does not exists a rule $x \leftarrow y_1, \ldots, y_n$ in P such that $\forall i \in \{1, \ldots, n\} : y_i \in T$ or $y_i = \texttt{TRUE}$, and
  - there exists a rule $x \leftarrow y_1, \ldots, y_n$ in P such that $\forall i \in \{1, \ldots, n\} : y_i \in T$ or $y_i \notin T \cup F$ or $y_i = \texttt{TRUE}$ or $y_i = \texttt{UNDEFINED}$;
- $x \in F'$ otherwise.

$(*)$ Note that for all $x \in Atms \setminus HB_P$ it holds that $x \in F'$ since there exists no rule with head x in P.

Let $\langle T_1, F_1 \rangle = \Psi(\langle \emptyset, Atms \rangle)$ be the 3-valued interpretation w.r.t. $Atms$ after the first iteration and let $\langle T_{i+1}, F_{i+1} \rangle = \Psi(\langle T_i, F_i \rangle)$ be the 3-valued interpretation w.r.t. $Atms$ after the $i+1$-th iteration. Furthermore let $\langle T, F \rangle$ be the least fixed point of Ψ, obtained by

iterating ω times the operator Ψ (starting with $\langle\emptyset, Atms\rangle$ and with respect to the standard ordering of 3-valued interpretations where $\langle T, F\rangle \preceq \langle T', F'\rangle$ iff $T \subseteq T'$ and $F' \subseteq F$), so $\Psi(\langle T, F\rangle) = \langle T, F\rangle = \Phi_{Atms}(P)$.
We recall that $x \in HB_P$. "$\Rightarrow$":

1. Proof by induction over the iterations of Ψ.
   Basis: Let $x \in T_1$. By definition of Ψ, there exists a rule with head x and either empty body or only `TRUE` occurring in the body. Thus, there exists an argument for x where every leaf node is labelled `TRUE`.
   Induction Hypothesis: Let $\langle T_i, F_i\rangle = \Psi(\langle T_{i-1}, F_{i-1}\rangle)$ and $x \in T_i$. Then there exists an argument for x where every leaf node is labelled `TRUE`.
   Inductive Step: Let $x \in T_{i+1}$. Then there exists a rule $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\} : y_j \in T_i$ or $y_j =$ `TRUE`, so $y_j \in HB_P$ by $(*)$. By the Induction Hypothesis, for every $y_j \neq$ `TRUE` there exists an argument for y_j where every leaf node is labelled `TRUE`, so there exists an argument for x where all leaf nodes are labelled `TRUE`.

2. Proof of the contraposition by induction over the depth of the argument for x.
   Basis: Let there be an argument for x with depth 1 such that no leaf node is labelled `FALSE`. Then there exists a rule $x \leftarrow$ or $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\}$: $y_j =$ `TRUE` or $y_j =$ `UNDEFINED`. By definition of Ψ, in the first case $x \in T$, in the second case $x \in T$ (if all $y_j =$ `TRUE`) or $x \notin T \cup F$ (if some $y_j =$ `UNDEFINED`). Thus, $x \notin F$.
   Induction Hypothesis: If there exists an argument for $x \in HB_P$ with depth at most i such that no leaf node is labelled `FALSE`, then $x \notin F$.
   Inductive Step: Let there be an argument for x with depth at most $i + 1$ such that no leaf node is labelled `FALSE`. Then there exists a rule $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\} : y_j \in HB_P$ or $y_j =$ `TRUE` or $y_j =$ `UNDEFINED`. Then for all $y_j \in HB_P$ there exists an argument for y_j of depth at most i such that no leaf node is labelled `FALSE`. By the Induction Hypothesis, $y_j \notin F$, so $\forall j \in \{1, \ldots, n\} : y_j \in T$ or $y_j \notin T \cup F$. By definition of Ψ, $x \in T$ (if all $y_j \in T$) or $x \notin T \cup F$ (if some $y_j \notin T \cup F$), so $x \notin F$.

"$\Leftarrow$":

1. Proof by induction over the depth of the argument for x.
   Basis: Let there be an argument for x with depth 1 where every leaf node is labelled `TRUE`. Then there exists a rule $x \leftarrow$ or $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\} : y_j =$ `TRUE`. By definition of Ψ, in both cases $x \in T$.
   Induction Hypothesis: If there exists an argument for $x \in HB_P$ with depth at most i where every leaf node is labelled `TRUE`, then $x \in T$.
   Inductive Step: Let there be an argument for x with depth at most $i + 1$ where every leaf node is labelled `TRUE`. Then there exists a rule $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\} : y_j \in HB_P$ or $y_j =$ `TRUE`. Thus, for all $y_j \in HB_P$ there exists an argument for y_j of depth at most i where every leaf node is labelled `TRUE`. By the Induction Hypothesis, $y_j \in T$, so by definition of Ψ, $x \in T$.

2. Proof of the contraposition by induction over the iterations of Ψ.
   Basis: Let $x \notin F_1$. By definition of Ψ there exists a rule $x \leftarrow$ or $x \leftarrow y_1, \ldots, y_n$ such

that $\forall j \in \{1, \ldots, n\} : y_j =$ `TRUE` or $y_j =$ `UNDEFINED`. Thus, there exists an argument for x such that all leaves are labelled `TRUE` or `UNDEFINED`, so no leaf is labelled `FALSE`.
Induction Hypothesis: If $\langle T_i, F_i \rangle = \Psi(\langle T_{i-1}, F_{i-1} \rangle)$ and $x \notin F_i$, then there exists an argument for x such that no leaf node is labelled `FALSE`.
Inductive Step: Let $x \notin F_{i+1}$. By definition of Ψ there exists a rule $x \leftarrow y_1, \ldots, y_n$ such that $\forall j \in \{1, \ldots, n\} : y_j \in T_i$ or $y_j \in HB_P \setminus (T_i \cup F_i)$ or $y_j =$ `TRUE` or $y_j =$ `UNDEFINED`, so $y_j \notin F_i$ and $y_j \in HB_P$ by $(*)$. Then by the Inductive Hypothesis, for all y_j there exists an argument such that no leaf node is labelled `FALSE`. Thus, there exists an argument for x such that no leaf node is labelled `FALSE`.

□

Proposition 3. *Let P be a logic program, possibly containing the special atoms* `TRUE`*,* `FALSE` *and* `UNDEFINED`*. Let P' be the logic program derived from P by*

- *removing each occurrence of* `TRUE` *from the bodies of the rules*
- *removing each rule that contains* `FALSE` *in its body*
- *replacing each occurrence of* `UNDEFINED` *by a new atom u and adding a rule $u \leftarrow$* `not` *u*

It holds that $\mathcal{M}od$ is a 3-valued stable (resp. well-founded, regular, 2-valued stable or ideal) model of P iff $\mathcal{M}od$ is a 3-valued stable (resp. well-founded, regular, 2-valued stable or ideal) model of P'.

Proof. We first prove equivalence under the 3-valued stable model semantics.
"$\Rightarrow$": Let $\mathcal{M}od = \langle T, F \rangle$ be a 3-valued stable model of P. That is, $\Phi_{HB_P}(P^{\langle T,F \rangle}) = \langle T, F \rangle$. Let $\Phi_{HB_{P'}}(P'^{\langle T,F \rangle}) = \langle T_{P'}, F_{P'} \rangle$. We need to prove that $\langle T, F \rangle = \langle T_{P'}, F_{P'} \rangle$.

1. Let $x \in T$. Then (Proposition 2) there exists an LP argument for x under $P^{\langle T,F \rangle}$ where each leaf node is labelled `TRUE`. But then there also exists an associated LP argument for x under $P'^{\langle T,F \rangle}$ where each leaf node is labelled `TRUE`. Hence (Proposition 2) $x \in T_{P'}$.

2. Let $x \in T_{P'}$. Then (Proposition 2) there exists an LP argument for x under $P'^{\langle T,F \rangle}$ where each leaf node is labelled `TRUE`. But then there also exists an associated LP argument for x under $P^{\langle T,F \rangle}$ where each leaf node is labelled `TRUE`. Hence (Proposition 2 $x \in T_P$.

3. Let $x \in F$. Then (Proposition 2) each LP argument for x under $P^{\langle T,F \rangle}$ has at least one leaf node that is labelled `FALSE`. But then also each LP argument for x under $P'^{\langle T,F \rangle}$ has at least one leaf node that is labelled `FALSE`. Hence (Proposition 2 $x \in F_{P'}$.

4. Let $x \in F_{P'}$. Then (Proposition 2) each LP argument for x under $P'^{\langle T,F \rangle}$ has at least one leaf node that is labelled `FALSE`. But then also each LP argument for x under $P^{\langle T,F \rangle}$ has at least one leaf node that is labelled `FALSE`. Hence (Proposition 2) $x \in F$.

"$\Leftarrow$": This is similar to "$\Rightarrow$".
As we just established, P and P' have the same 3-valued stable models. It then follows that P and P' also have the same regular (resp. well-founded and 2-valued stable) models, from which it follows that P and P' also have the same ideal model. □

Theorem 2

Proof. 1. Let $\mathcal{L}ab$ be a complete assumption labelling of $\mathcal{F}$ and let $\langle T, F\rangle$ be $\texttt{Lab2Mod}(\mathcal{L}ab)$. That is, $\langle T, F\rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T',F'\rangle})$ where $T' = \{\overline{\chi} \mid \chi \in \textsc{out}(\mathcal{L}ab)\} \cap HB_{P_\mathcal{F}}$ and $F' = \{\overline{\chi} \mid \chi \in \textsc{in}(\mathcal{L}ab)\} \cap HB_{P_\mathcal{F}}$. $\langle T', F'\rangle$ is a well-defined 3-valued interpretation of $P_\mathcal{F}$ since $T', F' \subseteq HB_{P_\mathcal{F}}$ and $T' \cap F' = \emptyset$, the latter following from the facts that $\textsc{in}(\mathcal{L}ab) \cap \textsc{out}(\mathcal{L}ab) = \emptyset$ and that two assumptions which have the same contrary have the same label in $\mathcal{L}ab$. It follows that also $\langle T, F\rangle$ is a well-defined 3-valued interpretation of $P_\mathcal{F}$.

We proceed to show that applying the Gelfond-Lifschitz reduct (w.r.t. $\langle T', F'\rangle$) does not change the status of any of the NAF literals in $P_\mathcal{F}$. Let "$\texttt{not}\ x$" be a NAF literal in some rule of $P_\mathcal{F}$. We distinguish three cases:

(a) $x \in T'$. Then there exists an assumption χ with $\overline{\chi} = x$ and $\chi \in \textsc{out}(\mathcal{L}ab)$. Since $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$, there exists an ABA argument $Asms \vdash x$ with $Asms \subseteq \textsc{in}(\mathcal{L}ab)$. Moreover, as x is not an assumption, this argument is non-trivial. It follows that there exists an LP argument for x under $P_\mathcal{F}$ where every leaf node is labelled $\texttt{TRUE}$ or "$\texttt{not}\ z$", where $z \in F'$. Thus, there exists an LP argument for x under $P_\mathcal{F}^{\langle T',F'\rangle}$ where every leaf node is labelled $\texttt{TRUE}$. By Proposition 2, $x \in T$.

(b) $x \in F'$. Then,there exists an assumption χ with $\overline{\chi} = x$ and $\chi \in \textsc{in}(\mathcal{L}ab)$. Since $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$, for each ABA argument $Asms \vdash x$ (which has to be non-trivial) it holds that there is a $\zeta \in Asms$ with $\zeta \in \textsc{out}(\mathcal{L}ab)$. Then each LP argument for x under $P_\mathcal{F}$ has a leaf node "$\texttt{not}\ z$" with $z \in T'$. Thus, each LP argument for x under $P_\mathcal{F}^{\langle T',F'\rangle}$ has a leaf node labelled $\texttt{FALSE}$. By Proposition 2, $x \in F$.

(c) $x \in HB_{P_\mathcal{F}} \setminus (T' \cup F')$.
Since $x \notin T'$ for each assumption χ with $\overline{\chi} = x$ it holds that $\chi \notin \textsc{out}(\mathcal{L}ab)$. Since $\mathcal{L}ab$ is a complete assumption labelling it follows from point 2 of Proposition 1 that there is no ABA argument $Asms \vdash x$ with $Asms \subseteq \textsc{in}(\mathcal{L}ab)$. Hence, for each ABA argument $Asms \vdash x$ (which has to be non-trivial) there is a $\zeta \in Asms$ with $\zeta \notin \textsc{in}(\mathcal{L}ab)$. Then each LP argument for x under $P_\mathcal{F}$ has a leaf node labelled "$\texttt{not}\ z$" with $z \notin F'$. Thus, each LP argument for x in $P_\mathcal{F}^{\langle T',F'\rangle}$ has a leaf node that is not labelled $\texttt{TRUE}$. By Proposition 2, $x \notin T$.
Since $x \notin F'$ there is no assumption χ with $\overline{\chi} = x$ and $\chi \in \textsc{in}(\mathcal{L}ab)$. That is, for each assumption χ with $\overline{\chi} = x$ it holds that $\chi \notin \textsc{in}(\mathcal{L}ab)$. Since $\mathcal{L}ab$ is a complete assumption labelling it follows from point 1 of Proposition 1 that there exists an ABA argument $Asms \vdash x$ (which has to be non-trivial) such that there exists no $\zeta \in Asms$ with $\zeta \in \textsc{out}(\mathcal{L}ab)$. Thus, there exists an LP argument for x under $P_\mathcal{F}$ without any leaf node that is labelled "$\texttt{not}\ z$" where $z \in T'$. Then there exists an argument for x in $P_\mathcal{F}^{\langle T',F'\rangle}$ without any leaf node that is labelled $\texttt{FALSE}$. By Proposition 2, $x \notin F$.
Thus, $x \in HB_{P_\mathcal{F}} \setminus (T \cup F)$.

So, overall we obtained that: if $x \in T'$ then $x \in T$, if $x \in F'$ then $x \in F$, and if $x \in HB_{P_\mathcal{F}} \setminus (T' \cup F')$ then $x \in HB_{P_\mathcal{F}} \setminus (T \cup F)$. So $\langle T', F' \rangle$ and $\langle T, F \rangle$ agree on the NAF literals of $P_\mathcal{F}$. It should be noted that whenever two 3-valued interpretations $\mathcal{M}od_1$ and $\mathcal{M}od_2$ of some logic program P agree on the NAF literals of P, the respective reducts $P^{\mathcal{M}od_1}$ and $P^{\mathcal{M}od_2}$ are equal. Hence, we have that $P_\mathcal{F}^{\langle T', F' \rangle} = P_\mathcal{F}^{\langle T, F \rangle}$, so also $\Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T', F' \rangle}) = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$. From $\Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T', F' \rangle}) = \langle T, F \rangle$ it then directly follows that $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$, so $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T', F' \rangle})$ is a 3-valued stable model of $P_\mathcal{F}$.

2. Let $\mathcal{M}od = \langle T, F \rangle$ be a 3-valued stable model of $P_\mathcal{F}$ (i.e. $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$) and let $\mathcal{L}ab = \texttt{Mod2Lab}(\mathcal{M}od)$. Let $\chi \in \mathcal{A}$. We distinguish three cases:
   (a) $\chi \in \text{IN}(\mathcal{L}ab)$. Then $\overline{\chi} \in F$ or $\overline{\chi} \notin HB_{P_\mathcal{F}}$.
      i. $\overline{\chi} \in F$. Since $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$ it follows from Proposition 2 that each LP argument for $\overline{\chi}$ under $P_\mathcal{F}^{\langle T, F \rangle}$ has a leaf node labelled `FALSE`. Then each LP argument for $\overline{\chi}$ under $P_\mathcal{F}$ has a leaf node labelled with some "`not` z" where $z \in T$. Thus, for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \cap \text{OUT}(\mathcal{L}ab) \neq \emptyset$.
      ii. $\overline{\chi} \notin HB_{P_\mathcal{F}}$. That is, $\overline{\chi}$ does not occur in any rule of $P_\mathcal{F}$, so also not in any rule of $\mathcal{F}$. Therefore, there is no ABA argument for $\overline{\chi}$. Hence, trivially, for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \cap \text{OUT}(\mathcal{L}ab) \neq \emptyset$.
   (b) $\chi \in \text{OUT}(\mathcal{L}ab)$. Then $\overline{\chi} \in T$. Since $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$ it follows from Proposition 2 there exists an LP argument for $\overline{\chi}$ under $P_\mathcal{F}^{\langle T, F \rangle}$ where each leaf node is labelled `TRUE`. Then there exists an LP argument for $\overline{\chi}$ under $P_\mathcal{F}$ where each leaf node is labelled with `TRUE` or with some "`not` z" where $z \in F$. Thus, there exists an ABA argument for $\overline{\chi}$ where each leaf node is labelled with either `TRUE` or with some assumption ζ with $\zeta \in \text{IN}(\mathcal{L}ab)$. That is, there exists an ABA argument $Asms \vdash \overline{\chi}$ with $Asms \subseteq \text{IN}(\mathcal{L}ab)$.
   (c) $\chi \in \text{UNDEC}(\mathcal{L}ab)$. Then $\overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T \cup F)$.
   Since $\overline{\chi} \notin T$ and $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$, it follows from Proposition 2 that there is no LP argument for $\overline{\chi}$ under $P_\mathcal{F}^{\langle T, F \rangle}$ where each leaf node is labelled `TRUE`. Then there is no LP argument for $\overline{\chi}$ under $P_\mathcal{F}$ where each leaf node is labelled with `TRUE` or with some "`not` z" where $z \in F$. Thus, there is no ABA argument $Asms \vdash \overline{\chi}$ with $Asms \subseteq \text{IN}(\mathcal{L}ab)$.
   Similarly, since $\overline{\chi} \notin F$ and $\langle T, F \rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T, F \rangle})$, it follows from Proposition 2 that there exists an LP argument for $\overline{\chi}$ under $P_\mathcal{F}^{\langle T, F \rangle}$ that does not have any leaf labelled `FALSE`, so there exists an LP argument for $\overline{\chi}$ under $P_\mathcal{F}$ that does not have any leaf labelled "`not` z" where $z \in T$. Thus, there exists an ABA argument $Asms \vdash \overline{\chi}$ with $Asms \cap \text{OUT}(\mathcal{L}ab) = \emptyset$.

To sum up, we have observed that: if $\chi \in \text{IN}(Lab)$ then for each ABA argument $Asms \vdash \overline{\chi}$ it holds that $Asms \cap \text{OUT}(\mathcal{L}ab)$, if $\chi \in \text{OUT}(\mathcal{L}ab)$ then there exists an ABA argument $Asms \vdash \overline{\chi}$ with $Asms \subseteq \text{IN}(\mathcal{L}ab)$, and if $\chi \in \text{UNDEC}(\mathcal{L}ab)$ then there is no ABA argument $Asms \vdash \overline{\chi}$ with $Asms \subseteq \text{IN}(\mathcal{L}ab)$ and there is an ABA argument $Asms \vdash \overline{\chi}$ with $Asms \cap \text{OUT}(\mathcal{L}ab) = \emptyset$. This means the conditions of Definition 3 are satisfied, so $\mathcal{L}ab$ is a complete assumption labelling.

3. It suffices to prove that: if $\mathcal{L}ab$ is a complete assumption labelling then it follows that `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)) $= \mathcal{L}ab$, and if $\mathcal{M}od$ is is a 3-valued stable model then `Lab2Mod`(`Mod2Lab`($\mathcal{M}od$)) $= \mathcal{M}od$.
As for the first equivalence that has to be proved, let $\mathcal{L}ab$ be a complete assumption labelling of $\mathcal{F}$, and let $\chi \in \mathcal{A}$. Let $\langle T, F\rangle =$ `Lab2Mod`($\mathcal{L}ab$) and T' and F' be as in the definition of `Lab2Mod` (Definition 12). We distinguish four cases.

   (a) $\chi \in \text{IN}(\mathcal{L}ab)$ and $\overline{\chi} \in HB_{P_\mathcal{F}}$. Then, by definition of F' (w.r.t. `Lab2Mod`($\mathcal{L}ab$)) it follows that $\overline{\chi} \in F'$. As we have observed earlier (in point 1 of the current theorem) it holds that if the contrary of a particular assumption is in F', then it is also in F. Hence, $\overline{\chi} \in F$. From the definition of `Mod2Lab` it then follows that χ is labelled IN by `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)).
   (b) $\chi \in \text{IN}(\mathcal{L}ab)$ and $\overline{\chi} \notin HB_{P_\mathcal{F}}$. We first observe that, also in this case, `Lab2Mod`($\mathcal{L}ab$) is well-defined. From the definition of `Mod2Lab` it then follows that χ is labelled IN by `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)).
   (c) $\chi \in \text{OUT}(\mathcal{L}ab)$. From point 2 of Definition 3 it follows that there exists an ABA argument for $\overline{\chi}$. Thus, there is an LP argument for $\overline{\chi}$ under $P_\mathcal{F}$, so $\overline{\chi} \in HB_{P_\mathcal{F}}$. Then $\overline{\chi} \in T'$. As we have observed earlier (in point 1 of the current theorem) it holds that if the contrary of a particular assumption is in T' then it is also in T. Hence, $\overline{\chi} \in T$. From the definition of `Mod2Lab` it then follows that χ is labelled OUT by `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)).
   (d) $\chi \in \text{UNDEC}(\mathcal{L}ab)$. From point 3 of Definition 3 it follows that there exists an ABA argument for $\overline{\chi}$ under $\mathcal{F}$. Thus, there is an LP argument for $\overline{\chi}$ under $P_\mathcal{F}$, so $\overline{\chi} \in HB_{P_\mathcal{F}}$. Then $\overline{\chi} \notin T'$ and $\overline{\chi} \notin F'$. Hence, $\overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T' \cup F')$. As we have observed earlier (in point 1 of the current theorem) it holds that if the contrary of a particular assumption is in $HB_{P_\mathcal{F}} \setminus (T' \cup F')$ then it is also in $HB_{P_\mathcal{F}} \setminus (T \cup F)$. Hence, $\overline{\chi} \in HB_{P_\mathcal{F}} \setminus (T \cup F)$. From the definition of `Mod2Lab` it then follows that χ is labelled UNDEC by `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)).

So overall, we observe that if χ is labelled IN (respectively OUT or UNDEC) by $\mathcal{L}ab$ then χ is labelled IN (respectively OUT or UNDEC) by `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)). Furthermore, `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)) does not assign any additional labels other than the ones assigned by $\mathcal{L}ab$. It can easily be verified that $\mathcal{L}ab$ and `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)) label the same set of assumptions $\mathcal{A}$. Then, since $\text{IN}(\mathcal{L}ab) \cup \text{OUT}(\mathcal{L}ab) \cup \text{UNDEC}(\mathcal{L}ab) = \mathcal{A}$, it follows that `Mod2Lab`(`Lab2Mod`($\mathcal{L}ab$)) $= \mathcal{L}ab$.
We now proceed to the second equivalence that has to be proved. Let $\mathcal{M}od = \langle T_{\mathcal{M}od}, F_{\mathcal{M}od}\rangle$ be a 3-valued stable model of $P_\mathcal{F}$, so $\langle T_{\mathcal{M}od}, F_{\mathcal{M}od}\rangle = \Phi_{HB_{P_\mathcal{F}}}(P_\mathcal{F}^{\langle T_{\mathcal{M}od}, F_{\mathcal{M}od}\rangle})$. Let $\langle T, F\rangle =$ `Lab2Mod`(`Mod2Lab`($\mathcal{M}od$)) and T' and F' as in the definition of `Lab2Mod` (w.r.t. `Lab2Mod`(`Mod2Lab`($\mathcal{M}od$))). Let $x \in HB_{P_\mathcal{F}}$. We distinguish three cases.

   (a) $x \in T_{\mathcal{M}od}$. By Proposition 2, there is an LP argument for x under $P_\mathcal{F}^{\langle T_{\mathcal{M}od}, F_{\mathcal{M}od}\rangle}$ where each leaf node is labelled `TRUE`. Thus, there exists an LP argument for

x under $P_{\mathcal{F}}$ where each leaf node is labelled with `TRUE` or with "`not` z" where $z \in F_{\mathcal{M}od}$. Then there exists an ABA argument for x under $\mathcal{F}$ where each leaf node is labelled with `TRUE` or with some assumption ζ that is labelled IN by `Mod2Lab`($\mathcal{M}od$). For each such ζ it holds that $\overline{\zeta} \in F'$. Then there is an LP argument for x under $P_{\mathcal{F}}^{\langle T',F'\rangle}$ where each leaf node is labelled `TRUE`. By Proposition 2, $x \in T$.

(b) $x \in F_{\mathcal{M}od}$. By Proposition 2, each LP argument for x under $P_{\mathcal{F}}^{\langle T_{\mathcal{M}od},F_{\mathcal{M}od}\rangle}$ has a leaf node labelled `FALSE`. Thus, each LP argument for x under $P_{\mathcal{F}}$ has a leaf node labelled with some "`not` z" where $z \in T_{\mathcal{M}od}$. Then each ABA argument for x has a leaf node labelled with some assumption ζ that is labelled OUT by `Mod2Lab`($\mathcal{M}od$). It follows that for each such ζ it holds that $\overline{\zeta} \in T'$. Then each argument for x under $P_{\mathcal{F}}^{\langle T',F'\rangle}$ has at least one leaf node labelled `FALSE`. By Proposition 2, $x \in F$.

(c) $x \in HB_{P_{\mathcal{F}}} \setminus (T_{\mathcal{M}od} \cup F_{\mathcal{M}od})$. From Proposition 2 (points 1 and 2) it follows that (1) there is no LP argument for x under $P_{\mathcal{F}}^{\langle T_{\mathcal{M}od},F_{\mathcal{M}od}\rangle}$ where each leaf node is labelled `TRUE`, and (2) there is an LP argument for x under $P_{\mathcal{F}}^{\langle T_{\mathcal{M}od},F_{\mathcal{M}od}\rangle}$ where no leaf node is labelled `FALSE`.
By (1), each LP argument for x under $P_{\mathcal{F}}^{\langle T_{\mathcal{M}od},F_{\mathcal{M}od}\rangle}$ has a leaf node that is not labelled `TRUE`. Then each LP argument for x under $P_{\mathcal{F}}$ has a leaf node that is labelled with "`not` z" where $z \notin F_{\mathcal{M}od}$. Thus, each ABA argument for x has a leaf node that is labelled with an assumption ζ that is not labelled IN by `Mod2Lab`($\mathcal{M}od$). Hence, $\overline{\zeta} \notin F'$. Then each LP argument for x under $P_{\mathcal{F}}^{\langle T',F'\rangle}$ has a leaf node that is not labelled `TRUE`. By Proposition 2, $x \notin T$.
By (2) and the definition of $P_{\mathcal{F}}^{\langle T_{\mathcal{M}od},F_{\mathcal{M}od}\rangle}$, there is an LP argument for x under $P_{\mathcal{F}}$ without any leaf being labelled with some "`not` z" with $z \in T_{\mathcal{M}od}$. Then there exists an ABA argument for x without any leaf node being labelled with some assumption that is labelled OUT by `Mod2Lab`($\mathcal{M}od$). That is, there exists an ABA argument for x where each assumption ζ is labelled IN or UNDEC by `Mod2Lab`($\mathcal{M}od$). For each such ζ, $\overline{\zeta} \notin T'$. Hence, there exists an LP argument for x under $P_{\mathcal{F}}^{\langle T',F'\rangle}$ without any leaf node labelled `FALSE`. By Proposition 2, $x \notin F$.
So, overall we obtain that $x \in HB_{P_{\mathcal{F}}} \setminus (T \cup F)$.

□

Theorem 4

Proof. 1. Let $\mathcal{L}ab$ be a grounded assumption labelling of $\mathcal{F}$, and let $\mathcal{M}od =$ `Lab2Mod`($\mathcal{L}ab$) $= \langle T, F\rangle$. By Theorem 2, $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}}$. We prove that T is *minimal* among all 3-valued stable models of $P_{\mathcal{F}}$. Let $\mathcal{M}od^* = \langle T^*, F^*\rangle$ be an arbitrary 3-valued stable model of $P_{\mathcal{F}}$. Suppose $T^* \subseteq T$. By Lemma 3, IN($\mathcal{L}ab^*$) $\subseteq$ IN($\mathcal{L}ab$), with $\mathcal{L}ab^* =$ `Mod2Lab`($\mathcal{M}od^*$). Since $\mathcal{L}ab$ is a grounded assumption labelling of $\mathcal{F}$, IN($\mathcal{L}ab$) is minimal among all complete assumption labellings of $\mathcal{F}$. Hence, IN($\mathcal{L}ab^*$) $=$ IN($\mathcal{L}ab$), so IN($\mathcal{L}ab^*$) $\supseteq$ IN($\mathcal{L}ab$). By Lemma 3, $T^* \supseteq T$, which together with $T^* \subseteq T$ implies that $T^* = T$. Hence, $\mathcal{M}od$ is a well-founded model of $P_{\mathcal{F}}$.

2. Let $\mathcal{M}od = \langle T, F \rangle$ be a well-founded model of $P_{\mathcal{F}}$, and let $\mathcal{L}ab = \texttt{Mod2Lab}(\mathcal{M}od)$. By Theorem 2, $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$. We prove that $\text{IN}(\mathcal{L}ab)$ is *minimal* among all complete assumption labellings of $\mathcal{F}$. Let $\mathcal{L}ab^*$ be an arbitrary complete assumption labelling of $\mathcal{F}$. Suppose $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab)$. By Lemma 3, $T^* \subseteq T$, with $\langle T^*, F^* \rangle = \mathcal{M}od^* = \texttt{Lab2Mod}(\mathcal{L}ab^*)$. Since $\mathcal{M}od$ is a well-founded model of $P_{\mathcal{F}}$, T is minimal among all 3-valued stable models of $P_{\mathcal{F}}$. Hence, $T^* = T$, so $T^* \supseteq T$. By Lemma 3, $\text{IN}(\mathcal{L}ab^*) \supseteq \text{IN}(\mathcal{L}ab)$, which together with $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab)$ implies that $\text{IN}(\mathcal{L}ab^*) = \text{IN}(\mathcal{L}ab)$. Hence, $\mathcal{L}ab$ is a grounded assumption labelling of $\mathcal{F}$.

3. Similar to point 1.

4. Similar to point 2.

5. Let $\mathcal{L}ab$ be a stable assumption labelling of $\mathcal{F}$. Then (Definition 4) $\text{UNDEC}(\mathcal{L}ab) = \emptyset$ so $\text{IN}(\mathcal{L}ab) \cup \text{OUT}(\mathcal{L}ab) = \mathcal{A}$. Let T' and F' be as in Definition 12 w.r.t. $\texttt{Lab2Mod}(\mathcal{L}ab)$. It then holds that for every NAF literal "$\texttt{not}\ z$", either $z \in T'$ or $z \in F'$. Let $\langle T, F \rangle = \texttt{Lab2Mod}(\mathcal{L}ab) = \Phi_{HB_{P_{\mathcal{F}}}}(P_{\mathcal{F}}^{\langle T', F' \rangle})$. Assume towards a contradiction that $T \cup F \neq HB_P$. This means that there exists an $x \in HB_P$ such that $x \notin T \cup F$. The fact that $x \notin T$ implies that (Proposition 2) there exists no LP argument for x under $P_{\mathcal{F}}^{\langle T', F' \rangle}$ where each leaf node is labelled TRUE, so each LP argument for x under $P_{\mathcal{F}}^{\langle T', F' \rangle}$ has a leaf node that is not labelled TRUE (so FALSE or UNDEFINED), so (1) each LP argument for x under $P_{\mathcal{F}}$ has a leaf node that is labelled "$\texttt{not}\ z$" for some $z \notin F'$. The fact that $x \notin F$ implies that (Proposition 2) not each LP argument for x under $P_{\mathcal{F}}^{\langle T', F' \rangle}$ has at least one leaf node that is labelled FALSE, so there exists an LP argument for x under $P_{\mathcal{F}}^{\langle T', F' \rangle}$ that has no leaf node labelled FALSE, so (2) there exists an LP argument for x under $P_{\mathcal{F}}$ that has no leaf node labelled "$\texttt{not}\ z$" with $z \in T'$. From (1) and (2) together, it follows that there exists an LP argument for x under $P_{\mathcal{F}}$ that has a leaf node labelled "$\texttt{not}\ z$" with $z \notin T'$ and $z \notin F'$. Contradiction. Therefore, $T \cup F = HB_P$. So $\langle T, F \rangle$ is a 3-valued stable model (Theorem 2, point 1) with $T \cup F = HB_P$, so a (2-valued) stable model.

6. Let $\mathcal{M}od = \langle T, F \rangle$ be a (2-valued) stable model of $P_{\mathcal{F}}$, and let $\mathcal{L}ab = \texttt{Mod2Lab}(\mathcal{M}od)$. By Theorem 2, $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$. Since $HB_{P_{\mathcal{F}}} \setminus (T \cup F) = \emptyset$, $\text{UNDEC}(\mathcal{L}ab) = \emptyset$, so $\mathcal{L}ab$ is a stable assumption labelling of $\mathcal{F}$.

7. Let $\mathcal{L}ab$ be an ideal assumption labelling of $\mathcal{F}$ and let $\mathcal{M}od = \texttt{Lab2Mod}(\mathcal{L}ab) = \langle T, F \rangle$. By Theorem 2 $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}}$. Since for all preferred assumption labellings $\mathcal{L}ab_{pref}$ of $\mathcal{F}$ it holds that $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$, by Lemma 3 $T \subseteq T_{reg}$ for all $\mathcal{M}od_{reg} = \texttt{Lab2Mod}(\mathcal{L}ab_{pref}) = \langle T_{reg}, F_{reg} \rangle$. Furthermore, by Theorem 4 (point 3) all $\mathcal{M}od_{reg}$ are regular models of $P_{\mathcal{F}}$. Thus, $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}}$ satisfying that for all regular models $\langle T_{reg}, F_{reg} \rangle$ of $P_{\mathcal{F}}$, $T \subseteq T_{reg}$. To show that in addition T is *maximal* among all 3-valued stable models of $P_{\mathcal{F}}$ satisfying that for all regular models $\langle T_{reg}, F_{reg} \rangle$ of $P_{\mathcal{F}}$, $T \subseteq T_{reg}$, let $\mathcal{M}od^* = \langle T^*, F^* \rangle$ be a 3-valued stable model of $P_{\mathcal{F}}$ satisfying that for all regular models $\langle T_{reg}, F_{reg} \rangle$ of $P_{\mathcal{F}}$, $T^* \subseteq T_{reg}$. Suppose $T^* \supseteq T$. Since for every regular model $\mathcal{M}od_{reg} = \langle T_{reg}, F_{reg} \rangle$ of $P_{\mathcal{F}}$ it holds that

$T^* \subseteq T_{reg}$, by Lemma 3 $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab_{pref})$ where $\mathcal{L}ab^* = \texttt{Mod2Lab}(\mathcal{M}od^*)$ and $\mathcal{L}ab_{pref} = \texttt{Mod2Lab}(\mathcal{M}od_{reg})$. Furthermore by Theorem 4 (point 4), all $\mathcal{L}ab_{pref}$ are preferred assumption labellings of $\mathcal{F}$. Thus, for all preferred assumption labellings $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab_{pref})$. By Lemma 3 also $\text{IN}(\mathcal{L}ab^*) \supseteq \text{IN}(\mathcal{L}ab)$. Since $\text{IN}(\mathcal{L}ab)$ is *maximal* among all complete assumption labellings of $\mathcal{F}$ with $\text{IN}(\mathcal{L}ab) \subseteq \{\text{IN}(\mathcal{L}ab_{pref}$ is a preferred assumption labelling of $\mathcal{F}\}$, it follows that $\text{IN}(\mathcal{L}ab^*) = \text{IN}(\mathcal{L}ab)$, so $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab)$. By Lemma 3, $T^* \subseteq T$, which together with $T^* \supseteq T$ implies that $T^* = T$ Hence, T is maximal among all 3-valued stable models of $P_{\mathcal{F}}$ satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$ of $P_{\mathcal{F}}$, $T \subseteq T_{reg}$, and thus $\mathcal{M}od$ is an ideal model of $P_{\mathcal{F}}$.

8. Let $\mathcal{M}od = \langle T, F\rangle$ be an ideal model of $P_{\mathcal{F}}$ and let $\mathcal{L}ab = \texttt{Mod2Lab}(\mathcal{M}od)$. By Theorem 2, $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$. Since for all regular models $\mathcal{M}od_{reg} = \langle T_{reg}, F_{reg}\rangle$ of $P_{\mathcal{F}}$ it holds that $T \subseteq T_{reg}$, by Lemma 3 $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$ with $\mathcal{L}ab_{pref} = \texttt{Mod2Lab}(\mathcal{M}od_{reg})$. By Theorem 4 (point 4), all $\mathcal{L}ab_{pref}$ are preferred assumption labellings of $\mathcal{F}$. Thus, $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$ satisfying that for all preferred assumption labellings $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$. To show that in addition $\text{IN}(\mathcal{L}ab)$ is *maximal* among all complete assumption labellings of $\mathcal{F}$ satisfying that for all preferred assumption labellings $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$, let $\mathcal{L}ab^*$ be a complete assumption labelling of $\mathcal{F}$ satisfying that for all preferred assumption labellings $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab_{pref})$. Suppose $\text{IN}(\mathcal{L}ab^*) \supseteq \text{IN}(\mathcal{L}ab)$. Since for every preferred assumption labelling $\mathcal{L}ab_{pref}$ of $\mathcal{F}$ it holds that $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab_{pref})$, by Lemma 3 $T^* \subseteq T_{reg}$ where $\mathcal{M}od^* = \langle T^*, F^*\rangle = \texttt{Lab2Mod}(\mathcal{L}ab^*)$ and $\mathcal{M}od_{reg} = \langle T_{reg}, F_{reg}\rangle = \texttt{Lab2Mod}(\mathcal{L}ab_{pref})$. By Theorem 4 (point 3), all $\mathcal{M}od_{reg}$ are regular models of $P_{\mathcal{F}}$. Thus, for all regular models $\langle T_{reg}, F_{reg}\rangle$ of $P_{\mathcal{F}}$, $T^* \subseteq T_{reg}$. By Lemma 3 also $T^* \supseteq T$. Since T is *maximal* among all 3-valued stable models of $P_{\mathcal{F}}$ satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$ of $P_{\mathcal{F}}$, $T \subseteq T_{reg}$, it follows that $T^* = T$, so $T^* \subseteq T$. By Lemma 3, $\text{IN}(\mathcal{L}ab^*) \subseteq \text{IN}(\mathcal{L}ab)$, which together with $\text{IN}(\mathcal{L}ab^*) \supseteq \text{IN}(\mathcal{L}ab)$ implies that $\text{IN}(\mathcal{L}ab^*) = \text{IN}(\mathcal{L}ab)$. Hence, $\text{IN}(\mathcal{L}ab)$ is maximal among all complete assumption labellings of $\mathcal{F}$ satisfying that for all preferred assumption labelling $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$, and thus $\mathcal{L}ab$ is an ideal assumption labelling of $\mathcal{F}$.

□

Theorem 8

Proof. 1. Let $\chi \in \mathcal{A}$. We distinguish two cases:

(a) $\overline{\chi} \in \mathcal{L} \setminus \mathcal{A}$. Then $\tilde{\chi} = \overline{\chi} \in \mathcal{L} \setminus \mathcal{A} \subseteq \mathcal{L}' \setminus \mathcal{A}$.

(b) $\overline{\chi} \in \mathcal{A}$. Then $\tilde{\chi} = \overline{\chi}^* \in \mathcal{L}' \setminus \mathcal{A}$.

2. "⇒": Suppose $Asms \vdash_{\mathcal{F}} \overline{\chi}$. We distinguish two cases:

(a) $Asms \vdash_{\mathcal{F}} \overline{\chi}$ is a trivial argument, i.e. $Asms = \{\overline{\chi}\}$ and $\overline{\chi}$ is an assumption. Then $\tilde{\chi} = \overline{\chi}^*$ and there exists a rule $\overline{\chi}^* \leftarrow \overline{\chi}$ in $\mathcal{R}'$. Thus, there is an argument under $\mathcal{F}'$ that starts with assumption $\overline{\chi}$ and applies the rule $\overline{\chi}^* \leftarrow \overline{\chi}$ to obtain conclusion $\overline{\chi}^*$. So $\{\overline{\chi}\} \vdash_{\mathcal{F}'} \overline{\chi}^*$. Since $\overline{\chi}^* = \tilde{\chi}$ it follows that $\{\overline{\chi}\} \vdash_{\mathcal{F}'} \tilde{\chi}$.

(b) $Asms \vdash_{\mathcal{F}} \overline{\chi}$ is a non-trivial argument, i.e. $\overline{\chi}$ is the consequent of a rule in $\mathcal{R}$. As $\mathcal{F}$ is flat, $\overline{\chi}$ is not an assumption, so $\tilde{\chi} = \overline{\chi}$. Since $\mathcal{R} \subseteq \mathcal{R}'$, we can construct the same argument under $\mathcal{F}'$, so $Asms \vdash_{\mathcal{F}'} \overline{\chi}$. As $\overline{\chi} = \tilde{\chi}$ it follows that $Asms \vdash_{\mathcal{F}'} \tilde{\chi}$.

"$\Leftarrow$": Suppose $Asms \vdash_{\mathcal{F}'} \tilde{\chi}$. We distinguish two cases:

(a) $\tilde{\chi} = \overline{\chi}$. This means $\overline{\chi} \in \mathcal{L} \setminus \mathcal{A}$, so the top rule used to construct the argument is in $\mathcal{R}$ and therefore all sentences in the antecedent are in $\mathcal{L}$. Thus all rules applied to construct the arguments for the sentences in the antecedent are in $\mathcal{R}$, and so on. So each rule used in the construction of $\tilde{\chi} = \overline{\chi}$ must be in $\mathcal{R}$. Therefore, the argument can also be constructed under $\mathcal{F}$, so So $Asms \vdash_{\mathcal{F}} \tilde{\chi}$. As $\tilde{\chi} = \overline{\chi}$ it follows that $Asms \vdash_{\mathcal{F}} \overline{\chi}$.

(b) $\tilde{\chi} = \overline{\chi}^*$. This means $\overline{\chi} \in \mathcal{A}$. As $\overline{\chi}^*$ is not an assumption in $\mathcal{F}'$, it follows that the argument has a top-rule $\overline{\chi}^* \leftarrow \overline{\chi}$. As $\overline{\chi}$ is an assumption and $\mathcal{F}'$ is flat, the argument cannot have any other rules. Hence, $Asms = \{\overline{\chi}\}$. This means that under $\mathcal{F}$ there exists an argument $\{\overline{\chi}\} \vdash_{\mathcal{F}} \overline{\chi}$.

□

Theorem 10

Proof. 1. Let $\mathcal{L}ab$ be a semi-stable assumption labelling of $\mathcal{F}$ and $\mathcal{M}od = \mathtt{Lab2Mod}(\mathcal{L}ab) = \langle T, F \rangle$. By Theorem 2, $\mathcal{M}od$ is a 3-valued stable model of $P_{\mathcal{F}}$. We prove that $HB_{P_{\mathcal{F}}} \setminus (T \cup F)$ is *minimal* among all 3-valued stable models of $P_{\mathcal{F}}$. Let $\mathcal{M}od^* = \langle T^*, F^* \rangle$ be an arbitrary 3-valued stable model of $P_{\mathcal{F}}$. Suppose $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) \subseteq HB_{P_{\mathcal{F}}} \setminus (T \cup F)$. By Theorem 2 and Lemma 9, $\textsc{undec}(\mathcal{L}ab^*) \subseteq \textsc{undec}(\mathcal{L}ab)$, with $\mathcal{L}ab^* = \mathtt{Mod2Lab}(\mathcal{M}od^*)$. Since $\mathcal{L}ab$ is a semi-stable assumption labelling of $\mathcal{F}$, $\textsc{undec}(\mathcal{L}ab)$ is minimal among all complete assumption labellings of $\mathcal{F}$. Hence, $\textsc{undec}(\mathcal{L}ab^*) = \textsc{undec}(\mathcal{L}ab)$, so $\textsc{undec}(\mathcal{L}ab^*) \supseteq \textsc{undec}(\mathcal{L}ab)$. By Theorem 2 and Lemma 9, $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) \supseteq HB_{P_{\mathcal{F}}} \setminus (T \cup F)$, which together with $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) \subseteq HB_{P_{\mathcal{F}}} \setminus (T \cup F)$ implies that $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) = HB_{P_{\mathcal{F}}} \setminus (T \cup F)$. Hence, $\mathcal{M}od$ is an L-stable model of $P_{\mathcal{F}}$.

2. Let $\mathcal{M}od = \langle T, F \rangle$ be an L-stable model of $P_{\mathcal{F}}$, and let $\mathcal{L}ab = \mathtt{Mod2Lab}(\mathcal{M}od)$. By Theorem 2, $\mathcal{L}ab$ is a complete assumption labelling of $\mathcal{F}$. We prove that $\textsc{undec}(\mathcal{L}ab)$ is *minimal* among all complete assumption labellings of $\mathcal{F}$. Let $\mathcal{L}ab^*$ be an arbitrary complete assumption labelling of $\mathcal{F}$. Suppose $\textsc{undec}(\mathcal{L}ab^*) \subseteq \textsc{undec}(\mathcal{L}ab)$. By Theorem 2 and Lemma 9, $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) \subseteq HB_{P_{\mathcal{F}}} \setminus (T \cup F)$, with $\langle T^*, F^* \rangle = \mathcal{M}od^* = \mathtt{Lab2Mod}(\mathcal{L}ab^*)$. Since $\mathcal{M}od$ is an L-stable model of $P_{\mathcal{F}}$, $HB_{P_{\mathcal{F}}} \setminus (T \cup F)$ is minimal among all 3-valued stable models of $P_{\mathcal{F}}$. Hence, $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) = HB_{P_{\mathcal{F}}} \setminus (T \cup F)$, so $HB_{P_{\mathcal{F}}} \setminus (T^* \cup F^*) \supseteq HB_{P_{\mathcal{F}}} \setminus (T \cup F)$. By Theorem 2 and Lemma 9, $\textsc{undec}(\mathcal{L}ab^*) \supseteq \textsc{undec}(\mathcal{L}ab)$, which together with $\textsc{undec}(\mathcal{L}ab^*) \subseteq \textsc{undec}(\mathcal{L}ab)$ implies that $\textsc{undec}(\mathcal{L}ab^*) = \textsc{undec}(\mathcal{L}ab)$. Hence, $\mathcal{L}ab$ is a semi-stable assumption labelling of $\mathcal{F}$.

□

Appendix B. Ideal Semantics

The ideal semantics for logic programs was first introduced by Alferes et al. (1993) in terms of scenaria, i.e. as the union of a logic program and a set of NAF literals (from which atoms can be derived using the T_P operator), rather than in terms of models. We recall the definition of scenaria.

Definition 19. *Let P be a logic program and $H \subseteq HB_P^{\texttt{not}}$ a set of NAF literals. $\vdash_{T_P}$ denotes derivation in the sense of the standard T_P operator for logic programs when treating NAF literals as atoms. Then*[14]

- *$P \cup H$ is a* scenario *of P.*
- *$P \cup H$ is* consistent *iff $\forall \texttt{not}\ x \in H$ it holds that $P \cup H \nvdash_{T_P} x$.*
- *A NAF literal $\texttt{not}\ x \in HB_P^{\texttt{not}}$ is* acceptable *w.r.t. $P \cup H$ iff $\forall H' \subseteq HB_P^{\texttt{not}}$ such that $P \cup H' \vdash_{T_P} x$ it holds that $\exists \texttt{not}\ y \in H'$ such that $P \cup H \vdash_{T_P} y$. The set of all NAF literals which are acceptable w.r.t. $P \cup H$ is denoted $Acc(H)$.*
- *$P \cup H$ is an* admissible scenario *iff it is consistent and $H \subseteq Acc(H)$.*
- *$P \cup H$ is the* ideal scenario *if H is the maximal (w.r.t. $\subseteq$) set satisfying: For all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario.*
- *$P \cup H$ is a* complete scenario *iff it is consistent and $H = Acc(H)$.*
- *$P \cup H$ is a* preferred extension *iff it is a maximal (w.r.t $\subseteq$) complete scenario.*

For a set of atoms $S \subseteq HB_P$, the set of all NAF literals corresponding to atoms in S is denoted $S_{HB^{\texttt{not}}} = \{\texttt{not}\ x \mid x \in S\}$. Conversely, for a set of NAF literals $H \subseteq HB_P^{\texttt{not}}$, the set of all atoms corresponding to NAF literals in H is denoted $H_{HB} = \{x \mid \texttt{not}\ x \in H\}$. Given a scenario $P \cup H$, $der(P \cup H) = \{x \mid P \cup H \vdash_{T_P} x\}$ denotes the set of atoms derivable from $P \cup H$.

We prove in Theorem 22 that the ideal semantics for logic programs as originally defined (Definition 19) is equivalent to the one given in Definition 9. The proof requires some further results, presented in the following.

Note that ideal scenaria are defined to be unique, whereas ideal models do not have a uniqueness condition in their definition. However, since the ideal semantics in ABA is unique (Dung et al., 2007), it follows from the correspondence between ideal assumption labellings and ideal models (see Theorems 4 and 7) that ideal models are unique.

Lemma 12. *Let P be a logic program and let $P \cup H_1$ and $P \cup H_2$ be two scenaria.*

1. *If $H_1 \subseteq H_2$ then $Acc(H_1) \subseteq Acc(H_2)$.*

14. Note that these definitions are simplified from Alferes et al. (1993) since we only consider normal logic programs without strong negation. Originally $Mand(H) = \{\texttt{not}\ x \mid P \cup H \cup \{\neg x' \rightarrow \texttt{not}\ x' | x' \in HB_P\} \vdash_{T_P} \texttt{not}\ x\}$ is taken into consideration for all semantics, but since no strong negation occurs in our logic programs, here $\forall H \subseteq HB_P^{\texttt{not}} : Mand(H) = \emptyset$. Furthermore, we do not need to make use of the transformation of a logic program into an "intended logic program" to encode the relation between strong negation and NAF.

2. $Acc(H_1) \cup Acc(H_2) \subseteq Acc(H_1 \cup H_2)$.

3. *If $P \cup H_1$ and $P \cup H_2$ are admissible scenaria and $P \cup H_1 \cup H_2$ is consistent then $P \cup H_1 \cup H_2$ is an admissible scenario.*

Proof. 1. Let $\texttt{not}\ x \in Acc(H_1)$. Then $\forall H' \subseteq HB_P^{\texttt{not}}$ with $P \cup H' \vdash_{T_P} x$ it holds that $\exists \texttt{not}\ y \in H'$ such that $P \cup H_1 \vdash_{T_P} y$. Since $H_1 \subseteq H_2$ it follows that $P \cup H_2 \vdash_{T_P} y$, so $\texttt{not}\ x \in Acc(H_2)$.

2. Let $\texttt{not}\ x \in Acc(H_1) \cup Acc(H_2)$, i.e. $\texttt{not}\ x \in Acc(H_1)$ or $\texttt{not}\ x \in Acc(H_2)$. In the first case, $\forall H' \subseteq HB_P^{\texttt{not}}$ with $P \cup H' \vdash_{T_P} x$ it holds that $\exists \texttt{not}\ y \in H'$ such that $P \cup H_1 \vdash_{T_P} y$. Then $P \cup H_1 \cup H_2 \vdash_{T_P} y$, so $\texttt{not}\ x \in Acc(H_1 \cup H_2)$. The same applies in the second case.

3. Since $P \cup H_1$ and $P \cup H_2$ are admissible scenaria, $H_1 \subseteq Acc(H_1)$ and $H_2 \subseteq Acc(H_2)$. Thus, $H_1 \cup H_2 \subseteq Acc(H_1) \cup Acc(H_2)$. By Lemma 12 (point 2), $Acc(H_1) \cup Acc(H_2) \subseteq Acc(H_1 \cup H_2)$, so $H_1 \cup H_2 \subseteq Acc(H_1 \cup H_2)$. Since $P \cup H_1 \cup H_2$ is consistent, $P \cup H_1 \cup H_2$ is an admissible scenario.

□

Lemma 13. *Let P be a logic program and $P \cup H$ a scenario. $P \cup H$ is a maximal (w.r.t. $\subseteq$) complete scenario iff $P \cup H$ is a maximal (w.r.t. $\subseteq$) admissible scenario.*

Proof. "⇒": Let $P \cup H$ be a maximal complete scenario. Then $P \cup H$ is an admissible scenario. Assume there exists a maximal admissible scenario $P \cup H'$ such that $H \subsetneq H'$. Since $P \cup H$ is a maximal complete scenario, $P \cup H'$ is not a complete scenario, so $H' \neq Acc(H')$, which together with $H' \subseteq Acc(H')$ implies that $H' \subsetneq Acc(H')$. By the Fundamental Lemma (Dung, 1991, 1995a), $P \cup Acc(H')$ is an admissible scenario, so $P \cup H'$ is not a maximal admissible scenario. Contradiction, so $P \cup H$ is a maximal admissible scenario.
"⇐": Let $P \cup H$ be a maximal admissible scenario. Assume that $P \cup H$ is not a complete scenario, so $H \neq Acc(H)$, which together with $H \subseteq Acc(H)$ implies that $H \subsetneq Acc(H)$. By the Fundamental Lemma (Dung, 1991, 1995a) $P \cup Acc(H')$ is an admissible scenario, so $P \cup H$ is not a maximal admissible scenario. Contradiction, so $P \cup H$ is a complete scenario. It is also a maximal complete scenario since any complete scenario $P \cup H'$ with $H \subsetneq H'$ is also an admissible scenario, but $P \cup H'$ is the maximal admissible scenario. □

Lemma 14. *Let P be a logic program and $P \cup H$ a scenario. $P \cup H$ is an admissible scenario satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$, iff for all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario.*

Proof. "⇒": Let $P \cup H$ be an admissible scenario satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$, and let $P \cup H'$ be an admissible scenario. By Theorem 1 in (Dung, 1991, 1995a), there exists a preferred extension $P \cup H_{pref}$ such that $H' \subseteq H_{pref}$. We know that $H \subseteq H_{pref}$, so $H' \cup H \subseteq H_{pref}$ and therefore $P \cup H' \cup H$ is consistent. By Lemma 12 (point 3), $P \cup H' \cup H$ is an admissible scenario.
"⇐": Let $P \cup H$ be a scenario satisfying: For all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario. Consider first the minimal admissible scenario, which is $P \cup \emptyset$ as observed by Alferes et al. (1993, p. 339). Then $P \cup \emptyset \cup H$ is an admissible scenario, so

$P \cup H$ is an admissible scenario. Now consider the maximal admissible scenaria, i.e. by Lemma 13 the preferred extensions. For all preferred extensions $P \cup H_{pref}$, $P \cup H_{pref} \cup H$ has to be an admissible scenario. Since $P \cup H_{pref}$ is a maximal admissible scenario, it follows that $H \subseteq H_{pref}$ for all preferred scenaria $P \cup H_{pref}$. Thus, for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. □

Even though the definition of ideal scenaria states that it is unique, this property has not been proven by Alferes et al. (1993).

Lemma 15. *Let P be a logic program. There exists a unique ideal scenario $P \cup H$.*

Proof. We first note that there exists a scenario $P \cup H$ such that for all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario, in particular $P \cup \emptyset$ fulfils this property. Thus, either $P \cup \emptyset$ is the only scenario fulfilling this property, making it maximal (w.r.t. $\subseteq$) among all scenaria satisfying this property, or there exists some $H \neq \emptyset$ such that $P \cup H$ is maximal among all scenaria fulfilling this property.
Let $P \cup H$ be a scenario which is maximal among all scenaria satisfying: For all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario. Assume $P \cup H_1$ ($H \neq H_1$) is also a scenario that is maximal among all scenaria satisfying this property. Then $P \cup H$ and $P \cup H_1$ are admissible scenaria since $P \cup \emptyset \cup H$ and $P \cup \emptyset \cup H_1$ have to be admissible scenaria. Furthermore, for all preferred extensions $P \cup H_{pref}$, $P \cup H_{pref} \cup H$ and $P \cup H_{pref} \cup H_1$ have to be admissible scenaria, so $H, H_1 \subseteq H_{pref}$ since H_{pref} is a maximal admissible scenario. Therefore, $H \cup H_1 \subseteq H_{pref}$ for every preferred extension $P \cup H_{pref}$ and thus $P \cup H \cup H_1$ is a consistent scenario. By Lemma 12 (point 3), $P \cup H \cup H_1$ is an admissible scenario. Since for every admissible scenario $P \cup H'$ there exists a preferred extension $P \cup H_{pref}$ such that $H' \subseteq H_{pref}$ (Theorem 1 in (Dung, 1991, 1995a)) and since $H \cup H_1 \subseteq H_{pref}$, it follows that $H' \cup (H \cup H_1) \subseteq H_{pref}$ and therefore $P \cup H' \cup (H \cup H_1)$ is a consistent scenario. Hence, by Lemma 12 (point 3), $P \cup H' \cup (H \cup H_1)$ is an admissible scenario (for every admissible scenario $P \cup H'$). Since $P \cup H$ and $P \cup H_1$ are maximal among all scenaria satisfying this property, it follows that $P \cup H' \cup (H \cup H_1) \subseteq P \cup H$ and $P \cup H' \cup (H \cup H_1) \subseteq P \cup H_1$. Then, $P \cup H' \cup (H \cup H_1) = P \cup H$ and $P \cup H' \cup (H \cup H_1) = P \cup H_1$, so $P \cup H = P \cup H_1$ and therefore $H = H_1$. Contradiction. Therefore, $P \cup H$ is the unique maximal scenario among all scenaria satisfying that for all admissible scenaria $P \cup H'$, $P \cup H' \cup H$ is again an admissible scenario. □

Lemma 16 follows directly from Lemma 14 and Lemma 15.

Lemma 16. *Let P be a logic program. There exists a unique admissible scenario $P \cup H$ such that H is maximal (w.r.t. $\subseteq$) among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$.*

The correspondence stated in the following corollary was mentioned by Alferes et al. (1993) but not proven. It follows from Lemmas 14, 15, and 16.

Lemma 17. *Let P be a logic program and $P \cup H$ a scenario. $P \cup H$ is the ideal scenario according to Definition 19 iff $P \cup H$ is the admissible scenario such that H is maximal (w.r.t. $\subseteq$) among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$.*

Lemma 18. *Let P be a logic program and $P \cup H$ a scenario. $P \cup H$ is an admissible scenario such that H is maximal (w.r.t. $\subseteq$) among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$, iff $P \cup H$ is a complete scenario such that H is maximal (w.r.t. $\subseteq$) among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$.*

Proof. "$\Rightarrow$": Let $P \cup H$ be an admissible scenario such that H is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Assume that $P \cup H$ is not a complete scenario, so $\exists \texttt{not}\ x \in Acc(H)$ such that $\texttt{not}\ x \notin H$. Since for every preferred extension $P \cup H_{pref}$ it holds that $H \subseteq H_{pref}$, by Lemma 12 (point 1) $\texttt{not}\ x \in Acc(H_{pref})$ for all $P \cup H_{pref}$. Since all preferred extensions are complete scenaria, $\texttt{not}\ x \in H_{pref}$ for all $P \cup H_{pref}$. Since $\texttt{not}\ x \in Acc(H)$, $P \cup H \cup \{\texttt{not}\ x\}$ is an admissible scenario by the Fundamental Lemma in (Dung, 1991, 1995a). Clearly, $H \cup \{\texttt{not}\ x\} \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$, so H is not maximal. Contradiction, so $P \cup H$ is a complete scenario. Furthermore, there cannot be a complete scenario $P \cup H'$ such that $H \subsetneq H'$ and $H' \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$ since every complete scenario is an admissible scenario, so $P \cup H'$ would also be an admissible scenario with $H \subsetneq H'$ and $H' \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$. Contradiction, so $P \cup H$ is a complete scenario such that H is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$.

"$\Leftarrow$": Let $P \cup H$ be a complete scenario such that H is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Then $P \cup H$ is an admissible scenario. Assume that there exists an admissible scenario $P \cup H'$ such that $H \subsetneq H'$ and H' is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H' \subseteq H_{pref}$. Then $P \cup H'$ is not a complete scenario since H is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Thus, $H' \subsetneq Acc(H')$, i.e. $\exists \texttt{not}\ x \in Acc(H')$ such that $\texttt{not}\ x \notin H'$. By the same reasoning as in the first part of this proof, $H' \cup \{\texttt{not}\ x\}$ is an admissible scenario and $H' \cup \{\texttt{not}\ x\} \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$. Contradiction, since this implies that H is not maximal. Thus, $P \cup H$ is an admissible scenario such that H is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. □

Lemma 19. *Let P be a logic program and let $P \cup H_1$ and $P \cup H_2$ be two complete scenaria.*

1. *$H_1 \subsetneq H_2$ iff $der(P \cup H_1) \subsetneq der(P \cup H_2)$.*
2. *$H_1 = H_2$ iff $der(P \cup H_1) = der(P \cup H_2)$.*

Proof. 1. "$\Rightarrow$": Let $H_1 \subsetneq H_2$. Then $\forall \texttt{not}\ x_1 \in H_1$ it holds that $\texttt{not}\ x_1 \in H_2$ and $\exists \texttt{not}\ x_2 \in H_2$ such that $\texttt{not}\ x_2 \notin H_1$. Furthermore, $\forall x \in der(P \cup H_1)$ it clearly holds that $x \in der(P \cup H_2)$, so $der(P \cup H_1) \subseteq der(P \cup H_2)$. Since $P \cup H_1$ is a complete scenario, $\texttt{not}\ x_2 \notin Acc(H_1)$ but $\texttt{not}\ x_2 \in Acc(H_2)$. Thus, $\exists H \subseteq HB_P^{\texttt{not}}$ with $P \cup H \vdash_{T_P} x_2$ and $\nexists \texttt{not}\ y \in H$ such that $P \cup H_1 \vdash_{T_P} y$, but $\exists \texttt{not}\ y' \in H$ such that $P \cup H_2 \vdash_{T_P} y'$. Consequently, $der(P \cup H_1) \subsetneq der(P \cup H_2)$.

"$\Leftarrow$": If $der(P \cup H_1) \subsetneq der(P \cup H_2)$ then $Acc(H_1) \subseteq Acc(H_2)$. Since $Acc(H_1) = H_1$ and $Acc(H_2) = H_2$ it follows that $H_1 \subseteq H_2$. If $H_1 = H_2$ then clearly $der(P \cup H_1) = der(P \cup H_2)$ (contradiction). Thus, $H_1 \subsetneq H_2$.

2. "⇒": Trivial.
"⇐": If $der(P \cup H_1) = (P \cup H_2)$ then $Acc(H_1) = Acc(H_2)$. Since $Acc(H_1) = H_1$ and $Acc(H_2) = H_2$ it follows that $H_1 = H_2$.

□

Lemma 20. *Let P be a logic program and $P \cup H$ a complete scenario. Then $H \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$ iff $der(P \cup H) \subseteq der(P \cup H_{pref})$ for all preferred extensions $P \cup H_{pref}$.*

Proof. "⇒": Let $H \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$. By Lemma 19 it follows that for all preferred extensions $P \cup H_{pref}$, $der(P \cup H) \subseteq der(P \cup H_{pref})$.
"⇐": Analogous to "⇒" □

Lemma 21. *Let P be a logic program and $P \cup H$ a complete scenario. H is maximal (w.r.t $\subseteq$) among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$, iff $der(P \cup H)$ is maximal (w.r.t $\subseteq$) among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $der(P \cup H) \subseteq der(P \cup H_{pref})$.*

Proof. "⇒": Let H be maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Then by Lemma 20, $der(P \cup H) \subseteq der(P \cup H_{pref})$ for all preferred extensions $P \cup H_{pref}$. Assume there exists a complete scenario $P \cup H'$ such that $der(P \cup H') \subseteq der(P \cup H_{pref})$ for all preferred extensions $P \cup H_{pref}$ and $der(P \cup H) \subsetneq der(P \cup H')$. Then by Lemma 19, $H \subsetneq H'$ and by Lemma 20 $H' \subseteq H_{pref}$ for all preferred extensions $P \cup H_{pref}$. Contradiction since H is maximal among all complete scenaria satisfying this condition. Thus, $der(P \cup H)$ is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $der(P \cup H) \subseteq der(P \cup H_{pref})$.
"⇐": Analogous to "⇒" □

Theorem 22. *Let P be a logic program.*

1. *If $\langle T, F\rangle$ is an ideal model of P then $P \cup F_{HB^{\text{not}}}$ is the ideal scenario of P with $T = der(P \cup F_{HB^{\text{not}}})$.*

2. *If $P \cup H$ is the ideal scenario of P then $\langle der(P \cup H), H_{HB}\rangle$ is an ideal model of P.*

Proof. 1. Let $\langle T, F\rangle$ be an ideal model. By Definition 9, $\langle T, F\rangle$ is a 3-valued stable model such that T is maximal among all 3-valued stable models satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$, $T \subseteq T_{reg}$. By Theorem 2.1 in (You & Yuan, 1995), $\langle T_{reg}, F_{reg}\rangle$ is a regular model iff $P \cup F_{reg\,HB^{\text{not}}}$ is a preferred extension where $T_{reg} = der(P \cup F_{reg\,HB^{\text{not}}})$. Furthermore, by Theorem 3.1 in (Przymusinski, 1991) and Corollary 4.16 in (Brogi et al., 1992), $P \cup F_{HB^{\text{not}}}$ is a complete scenario with $T = der(P \cup F_{HB^{\text{not}}})$. Assume there exists a complete scenario $P \cup H$ such that for all preferred extensions $P \cup H_{pref}$, $der(P \cup H) \subseteq der(P \cup H_{pref})$ and $F_{HB^{\text{not}}} \subsetneq H$. Then by Lemma 19, $der(P \cup F_{HB^{\text{not}}}) \subsetneq der(P \cup H)$. By Theorem 3.1 in (Przymusinski, 1991) and Corollary 4.16 in (Brogi et al., 1992) $\langle der(P \cup H), H_{HB}\rangle$ is a 3-valued stable model. Furthermore, $der(P \cup H) \subseteq T_{reg}$ for all regular models $\langle T_{reg}, F_{reg}\rangle$. Contradiction since $\langle T, F\rangle$ with $T = der(P \cup F_{HB^{\text{not}}})$ is a 3-valued stable model such that T is maximal among all 3-valued stable models satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$, $T \subseteq T_{reg}$.

Thus, $P \cup F_{HB^{\text{not}}}$ is a complete scenario such that $der(P \cup F_{HB^{\text{not}}})$ is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $der(P \cup F_{HB^{\text{not}}}) \subseteq der(P \cup H_{pref})$. By Lemma 21, $P \cup F_{HB^{\text{not}}}$ is a complete scenario such that $F_{HB^{\text{not}}}$ is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $F_{HB^{\text{not}}} \subseteq H_{pref}$. By Lemma 18, $P \cup F_{HB^{\text{not}}}$ is an admissible scenario such that $F_{HB^{\text{not}}}$ is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $F_{HB^{\text{not}}} \subseteq H_{pref}$. By Lemma 16 $P \cup F_{HB^{\text{not}}}$ is the unique admissible scenario satisfying this property. Then by Lemma 17, $P \cup F_{HB^{\text{not}}}$ is the ideal scenario.

2. Let $P \cup H$ be the ideal scenario. By Lemma 17, $P \cup H$ is the admissible scenario such that H is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Then by Lemma 18, $P \cup H$ is a complete scenario such that H is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. By Lemma 21, $P \cup H$ is a complete scenario such that $der(P \cup H)$ is maximal among all complete scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $der(P \cup H) \subseteq der(P \cup H_{pref})$. By Theorem 3.1 in (Przymusinski, 1991) and Corollary 4.16 in (Brogi et al., 1992), $\langle der(P \cup H), H_{HB}\rangle$ is a 3-valued stable model. Furthermore, by Theorem 2.1 in (You & Yuan, 1995), $\langle T_{reg}, F_{reg}\rangle$ is a regular model iff $P \cup F_{reg\,HB^{\text{not}}}$ is a preferred extension where $T_{reg} = der(P \cup F_{reg\,HB^{\text{not}}})$. Thus, $\langle der(P \cup H), H_{HB}\rangle$ is a 3-valued stable model satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$, $der(P \cup H) \subseteq T_{reg}$. Assume there exists a 3-valued stable model $\langle T, F\rangle$ satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$, $T \subseteq T_{reg}$ and $der(P \cup H) \subsetneq T$. By Theorem 3.1 in (Przymusinski, 1991) and Corollary 4.16 in (Brogi et al., 1992), $P \cup F_{HB^{\text{not}}}$ is a complete scenario with $T = der(P \cup F_{HB^{\text{not}}})$. Furthermore, $F_{HB^{\text{not}}} \subseteq der(P \cup H_{pref})$ for all preferred extensions $P \cup H_{pref}$. By Lemma 19, $H \subsetneq F_{HB^{\text{not}}}$. Contradiction since H is maximal among all admissible scenaria satisfying that for all preferred extensions $P \cup H_{pref}$, $H \subseteq H_{pref}$. Thus, $\langle der(P \cup H), H_{HB}\rangle$ is a 3-valued stable model such that $der(P \cup H)$ is maximal among all 3-valued stable models satisfying that for all regular models $\langle T_{reg}, F_{reg}\rangle$, $der(P \cup H) \subseteq T_{reg}$. Thus, $\langle der(P \cup H), H_{HB}\rangle$ is an ideal model.

□

References

Alferes, J. J., Dung, P. M., & Pereira, L. M. (1993). Scenario semantics of extended logic programs. In Nerode, A., & Pereira, L. (Eds.), *Proceedings of the 2nd International Workshop on Logic Programming and Non-monotonic Reasoning (LPNMR)*, pp. 334–348. MIT Press.

Alferes, J. J., & Pereira, L. M. (1992). On logic program semantics with two kinds of negation. In *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP)*, pp. 574–588.

Alviano, M., Dodaro, C., Leone, N., & Ricca, F. (2015). Advances in WASP. In *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pp. 40–54.

Baláz, M., Frtús, J., Flouris, G., Homola, M., & Šefránek, J. (2014). Conflict Resolution in Assumption-Based Frameworks. In *Revised Selected Papers of the 12th European Conference on Multi-Agent Systems (EUMAS)*, pp. 360–369.

Besnard, P., García, A. J., Hunter, A., Modgil, S., Prakken, H., Simari, G. R., & Toni, F. (2014). Introduction to structured argumentation. *Argument & Computation*, *5*(1), 1–4.

Bondarenko, A., Dung, P. M., Kowalski, R. A., & Toni, F. (1997). An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, *93*, 63–101.

Brewka, G., Eiter, T., & Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, *54*(12), 92–103.

Brogi, A., Lamma, E., Mancarella, P., & Mello, P. (1992). Normal logic programs as open positive programs. In Apt, K. R. (Ed.), *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP)*, pp. 783–797. MIT Press.

Calimeri, F., Gebser, M., Maratea, M., & Ricca, F. (2016). Design and results of the fifth answer set programming competition. *Artificial Intelligence*, *231*, 151–181.

Calimeri, F., Ianni, G., Krennwallner, T., & Ricca, F. (2012). The answer set programming competition. *AI Magazine*, *33*(4), 114–118.

Caminada, M. W. A. (2006). Semi-stable semantics. In Dunne, P. E., & Bench-Capon, T. M. (Eds.), *Proceedings of Computational Models of Argument (COMMA)*, pp. 121–130. IOS Press.

Caminada, M. W. A., Sá, S., Alcântara, J., & Dvořák, W. (2015a). On the equivalence between logic programming semantics and argumentation semantics. *International Journal of Approximate Reasoning*, *58*, 87–111.

Caminada, M. W. A., Sá, S., Alcântara, J., & Dvořák, W. (2015b). On the difference between assumption-based argumentation and abstract argumentation. *The IfCoLog Journal of Logics and their Applications*, *2*(1), 16–34.

Caminada, M. W. A., & Schulz, C. (2015). On the Equivalence between Assumption-Based Argumentation and Logic Programming. In *Proceedings of the 1st International Workshop on Argumentation and Logic Programming (ArgLP)*.

Cayrol, C., & Lagasquie-Schiex, M.-C. (2013). Bipolarity in Argumentation Graphs: Towards a Better Understanding. *International Journal of Approximate Reasoning*, *54*(7), 876–899.

Chesñevar, C. I., Dix, J., Stolzenburg, F., & Simari, G. R. (2003). Relating defeasible and normal logic programming through transformation properties. *Theoretical Computer Science*, *290*(1), 499 – 529.

Craven, R., Toni, F., Cadar, C., Hadad, A., & Williams, M. (2012). Efficient argumentation for medical decision-making. In Brewka, G., Eiter, T., & McIlraith, S. A. (Eds.), *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press.

Cyras, K., & Toni, F. (2016). ABA+: Assumption-Based Argumentation with Preferences. In *Proceedings of the15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 553–556.

Dung, P. M. (1991). Negations as hypotheses: An abductive foundation for logic programming. In Furukawa, K. (Ed.), *Proceedings of the 8th International Conference on Logic Programming (ICLP)*, pp. 3–17. MIT Press.

Dung, P. M., Kowalski, R. A., & Toni, F. (2009). Assumption-based argumentation. In Simari, G., & Rahwan, I. (Eds.), *Argumentation in Artificial Intelligence*, pp. 199–218. Springer US.

Dung, P. M., Mancarella, P., & Toni, F. (2007). Computing ideal sceptical argumentation. *Artificial Intelligence*, *171*(10-15), 642–674.

Dung, P. M., & Thang, P. M. (2009). Modular argumentation for modelling legal doctrines in common law of contract. *Artificial Intelligence and Law*, *17*(3), 167–182.

Dung, P. M. (1995a). An argumentation-theoretic foundation for logic programming. *The Journal of Logic Programming*, *22*(2), 151–177.

Dung, P. M. (1995b). On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, *77*(2), 321–357.

Dung, P. M., Kowalski, R. A., & Toni, F. (2006). Dialectic Proof Procedures for Assumption-Based, Admissible Argumentation. *Artificial Intelligence*, *170*(2), 114–159.

Dung, P. M., & Thang, P. M. (2009). Modular argumentation for modelling legal doctrines in common law of contract. *Artificial Intelligence and Law*, *17*(3), 167–182.

Dung, P. M., & Thang, P. M. (2014). Closure and consistency in logic-associated argumentation. *Journal of Artificial Intelligence Research*, *49*, 79–109.

Dung, P. M., Thang, P. M., & Hung, N. D. (2010). Modular Argumentation for Modelling Legal Doctrines of Performance Relief. *Argument & Computation*, *1*(1), 47–69.

Dung, P. M., Thang, P. M., & Toni, F. (2008). Towards Argumentation-Based Contract Negotiation. In *Proceedings of the 2nd International Conference on Computational Models of Argument (COMMA)*, pp. 134–146.

Dunne, P. E. (2009). The Computational Complexity of Ideal Semantics. *Artificial Intelligence*, *173*(18), 1559–1591.

Eiter, T., Leone, N., & Saccà, D. (1997). On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence*, *19*(1-2), 59–96.

Fan, X., & Toni, F. (2014). A general framework for sound assumption-based argumentation dialogues. *Artificial Intelligence*, *216*, 20–54.

Fan, X., & Toni, F. (2015). On computing explanations in argumentation. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1496–1502. AAAI Press.

Fan, X., Toni, F., & Hussain, A. (2010). Two-Agent Conflict Resolution with Assumption-Based Argumentation. In *Proceedings of the 3rd International Conference on Computational Models of Argument (COMMA)*, pp. 231–242.

Gaertner, D., & Toni, F. (2007). Computing arguments and attacks in assumption-based argumentation. *IEEE Intelligent Systems*, *22*(6), 24–33.

Gaertner, D., & Toni, F. (2008). Hybrid argumentation and its properties. In Besnard, P., Doutre, S., & Hunter, A. (Eds.), *Proceedings of Computational Models of Argument (COMMA)*, pp. 183–195. IOS Press.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., & Schneider, M. T. (2011). Potassco: The potsdam answer set solving collection. *AI Communications*, *24*(2), 107–124.

Gottlob, G. (1995). Translating Default Logic into Standard Autoepistemic Logic. *Journal of the ACM*, *42*(4), 711–740.

Greco, S., Trubitsyna, I., & Zumpano, E. (2007). On the semantics of logic programs with preferences. *Journal of Artificial Intelligence Research*, *30*, 501–523.

Grooters, D., & Prakken, H. (2016). Two aspects of relevance in structured argumentation: Minimality and paraconsistency. *Journal of Artificial Intelligence Research*, *56*, 197–245.

Heyninck, J., Pardo, P., & Straßer, C. (2017). Assumption-based approaches to reasoning with priorities. In *Proceedings of the 1st Workshop on Advances In Argumentation In Artificial Intelligence (AI^3)*.

Heyninck, J., & Straßer, C. (2016). Relations between Assumption-Based Approaches in Nonmonotonic Logic and Formal Argumentation. In *Proceedings of the 16th International Workshop on Non-Monotonic Reasoning (NMR)*.

Imielinski, T. (1987). Results on Translating Defaults to Circumscription. *Artificial Intelligence*, *32*(1), 131–146.

Janhunen, T. (1999). On the Intertranslatability of Non-monotonic Logics. *Annals of Mathematics and Artificial Intelligence*, *27*(1-4), 79–128.

Janhunen, T., Niemelä, I., Seipel, D., Simons, P., & You, J. (2006). Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic*, *7*(1), 1–37.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., & Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, *7*(3), 499–562.

Lin, F., & Zhao, Y. (2004). ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, *157*(1-2), 115–137.

Liu, G., Janhunen, T., & Niemelä, I. (2012). Answer set programming via mixed integer programming. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Matt, P.-A., Toni, F., Stournaras, T., & Dimitrelos, D. (2008). Argumentation-Based Agents for eProcurement. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 71–74.

Nieves, J. C., & Osorio, M. (2016). Ideal extensions as logical programming models. *Journal of Logic and Computation*, *26*(5), 1361—-1393.

Oren, N., Reed, C., & Luck, M. (2010). Moving Between Argumentation Frameworks. In *Proceedings of the 3rd International Conference on Computational Models of Argument (COMMA)*, pp. 379–390.

Osorio, M., Zepeda, C., Nieves, J. C., & Cortés, U. (2005). Inferring Acceptable Arguments with Answer Set Programming. In *Proceedings of the 6th Mexican International Conference on Computer Science (ENC'05)*, pp. 198–205.

Polberg, S., & Oren, N. (2014). Revisiting Support in Abstract Argumentation Systems. In *Proceedings of the 5th International Conference on Computational Models of Argument (COMMA)*, pp. 369–376.

Przymusinski, T. C. (1990). The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, *13*(4), 445–463.

Przymusinski, T. C. (1991). Semantics of disjunctive logic programs and deductive databases. In Delobel, C., Kifer, M., & Masunaga, Y. (Eds.), *Proceedings of the 2nd International Conference on Deductive and Object-Oriented Databases (DOOD)*, pp. 85–107. Springer Berlin Heidelberg.

Rahwan, I., & Simari, G. R. (2009). *Argumentation in Artificial Intelligence*. Springer US.

Ruiz, C., & Minker, J. (1994). Computing stable and partial stable models of extended disjunctive logic programs. In *Selected Papers of the Workshop on Non-Monotonic Extensions of Logic Programming (NMELP)*, pp. 205–229.

Schulz, C. (2015). Graphical representation of assumption-based argumentation. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4204–4205. AAAI Press.

Schulz, C., & Toni, F. (2014). Complete assumption labellings. In Parsons, S., Oren, N., Reed, C., & Cerutti, F. (Eds.), *Proceedings of Computational Models of Argument (COMMA)*, pp. 405–412. IOS Press.

Schulz, C., & Toni, F. (2015). Logic programming in assumption-based argumentation revisited - semantics and graphical representation. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1569–1575. AAAI Press.

Schulz, C., & Toni, F. (2016). Justifying answer sets using argumentation. *Theory and Practice of Logic Programming*, *16*(01), 59–110.

Schulz, C., & Toni, F. (2017). Labellings for assumption-based and abstract argumentation. *International Journal of Approximate Reasoning*, *84*, 110 – 149.

Strass, H. (2013). Approximating operators and semantics for abstract dialectical frameworks. *Artificial Intelligence*, *205*, 39–70.

Toni, F. (2007). Assumption-based argumentation for closed and consistent defeasible reasoning. In Satoh, K., Inokuchi, A., Nagao, K., & Kawamura, T. (Eds.), *Revised Selected Papers of the 21st Anual Conference of the Japanese Society for Artificial Intelligence (JSAI)*, pp. 390–402. Springer Berlin Heidelberg.

Toni, F. (2008). Assumption-based argumentation for epistemic and practical reasoning. In Casanovas, P., Sartor, G., Casellas, N., & Rubino, R. (Eds.), *Computable Models* *of the Law, Languages, Dialogues, Games, Ontologies*, pp. 185–202. Springer Berlin Heidelberg.

Toni, F. (2013). A generalised framework for dispute derivations in assumption-based argumentation. *Artificial Intelligence*, *195*, 1–43.

Toni, F. (2014). A tutorial on assumption-based argumentation. *Argument & Computation*, *5*, 89–117.

Toni, F. (2012). Reasoning on the Web with Assumption-Based Argumentation. In *Proceedings of the 8th International Summer School on Reasoning Web*, pp. 370–386.

Čyras, K., & Toni, F. (2015). Non-Monotonic Inference Properties for Assumption-Based Argumentation. In *Revised Selected Papers of the 3rd International Workshop on Theory and Applications of Formal Argumentation (TAFA)*, pp. 92–111.

Verheij, B. (1996). Two approaches to dialectical argumentation: Admissible sets and argumentation stages. In Meyer, J.-J. C., & van der Gaag, L. C. (Eds.), *Proceedings of the 8th Dutch Conference on Artificial Intelligence (NAIC)*, pp. 357–368.

Vesic, S. (2013). Identifying the class of maxi-consistent operators in argumentation. *Journal of Artificial Intelligence Research*, *47*, 71–93.

Wu, Y., Caminada, M. W. A., & Gabbay, D. M. (2009). Complete extensions in argumentation coincide with 3-valued stable models in logic programming. *Studia Logica*, *93*(2-3), 383–403.

You, J. H., & Yuan, L. Y. (1995). On the equivalence of semantics for normal logic programs. *The Journal of Logic Programming*, *22*(3), 211–222.

Young, A. P., Modgil, S., & Rodrigues, O. (2016). Prioritised Default Logic as Rational Argumentation. In *Proceedings of the 15th International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, pp. 626–634.