Logic Programming in Assumption-Based Argumentation Revisited – Semantics and Graphical Representation

Claudia Schulz and **Francesca Toni**
{claudia.schulz, ft}@imperial.ac.uk
Department of Computing
Imperial College London
London SW7 2AZ, UK

Abstract

Logic Programming and Argumentation Theory have been existing side by side as two separate, yet related, techniques in the field of Knowledge Representation and Reasoning for many years. When Assumption-Based Argumentation (ABA) was first introduced in the nineties, the authors showed how a logic program can be encoded in an ABA framework and proved that the stable semantics of a logic program corresponds to the stable extension semantics of the ABA framework encoding this logic program. We revisit this initial work by proving that the 3-valued stable semantics of a logic program coincides with the complete semantics of the encoding ABA framework, and that the L-stable semantics of this logic program coincides with the semi-stable semantics of the encoding ABA framework. Furthermore, we show how to graphically represent the structure of a logic program encoded in an ABA framework and that not only logic programming and ABA semantics but also Abstract Argumentation semantics can be easily applied to a logic program using these graphical representations.

1 Introduction

Logic Programming (LP) is a frequently used technique for the representation of reasoning problems in terms of a logic program consisting of rules which are made of atoms and default elements called *negation-as-failure* (NAF) literals, which are assumed to be true as long as their complementary atom cannot be proven to hold. A variety of semantics have been defined for logic programs yielding different solutions to a given problem, e.g. the stable (Gelfond and Lifschitz 1988), 3-valued stable (Przymusinski 1989), 3-valued L-stable (Eiter, Leone, and Saccà 1997), and well-founded (Van Gelder, Ross, and Schlipf 1991) model semantics. Most of them are defined as sets of atoms forming a fixpoint of a function which alters the original logic program (see (Baral and Gelfond 1994) for an overview). *Assumption-Based Argumentation* (ABA) (Bondarenko et al. 1997; Dung, Kowalski, and Toni 2009; Toni 2014) is another technique for representing reasoning problems, but in terms of an ABA framework consisting of rules which are made of atoms and default elements called *assumptions*, where for each assumption a contrary literal is defined. Even though the representation of knowledge in ABA resembles logic programs, the semantics of an ABA framework are determined in a completely different way, namely as sets of assumptions called *extensions*, which are able to "defend" themselves against all contrary evidence. Recently, a new method for determining the complete semantics of an ABA framework has been introduced, where all assumptions are assigned one of the labels IN, OUT, or UNDEC according to *labelling* rules, thus splitting the assumptions into acceptable (IN), non-acceptable (OUT), and neutral (UNDEC) ones (Schulz and Toni 2014a). This method was proven equivalent to the notion of complete extension semantics, where the assumptions labelled IN coincide with the extensions.

In early work on ABA (Bondarenko et al. 1997), the authors demonstrate how to encode a logic program in an ABA framework in such a way that, for example, the stable models of the logic program correspond to the stable extensions of the encoding ABA framework. This correspondence has recently proven useful to explain stable models in argumentative terms (Schulz and Toni 2013; 2014b). Inspired by the usefulness of these correspondence results, we show that the 3-valued stable models of a logic program correspond to the complete labellings and extensions of the encoding ABA framework. Furthermore, we investigate the relation of the 3-valued L-stable semantics of a logic program with ABA semantics. By defining a labelling version of the semi-stable extension semantics in ABA (Caminada et al. 2013), we prove that the 3-valued L-stable models of a logic program correspond to the semi-stable labellings and extensions of the encoding ABA framework. These results do not only improve the understanding of the relationship between LP and ABA, but can also be useful in applications such as explanations of LP semantics (Schulz and Toni 2014b).

Based on our correspondence results, we also demonstrate how the semantics of a logic program can be easily displayed in a graphical representation of the ABA labelling semantics. Consequently, our correspondence results do not only provide novices and experts in LP with a different way of thinking about the 3-valued (L-) stable model semantics (namely in argumentative terms), but also with a graphical representation thereof. In addition, we show how such an ABA graph can be used to apply Abstract Argumentation (AA) semantics (Dung 1995; Caminada and Gabbay 2009) to a logic program, which is

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

particularly interesting for the semi-stable AA semantics as it differs from the 3-valued L-stable model semantics for logic programs (Caminada, Sá, and Alcântara 2013), as opposed to the semi-stable ABA semantics which (we prove) corresponds to the 3-valued L-stable model semantics for logic programs.

2 Background

Logic Programming

A *logic program* $\mathcal{P}$ is a set of *clauses* of the form $a \leftarrow a_1, \ldots, a_m, \ not \ a_{m+1}, \ldots, not \ a_{m+n}$ $(m, n \geq 0)$, where a and all a_i are atoms and $not \ a_{m+1}, \ldots, not \ a_{m+n}$ are *negation-as-failure* (NAF) literals. A *literal* is an atom or a NAF literal. $\neg.l$ denotes the complement of a literal l, i.e. if l is an atom a then $\neg.l = not \ a$, and if l is a NAF literal $not \ a$ then $\neg.l = a$. For a set of literals S, $\neg.S = \{\neg.l \mid l \in S\}$. $\mathcal{HB}_\mathcal{P}$ denotes the *Herbrand Base* of $\mathcal{P}$ and $Lit_\mathcal{P} = \mathcal{HB}_\mathcal{P} \cup \neg.\mathcal{HB}_\mathcal{P}$ is the set of all literals. Clauses containing variables are used as shorthand notation for all their ground instances. All LP concepts reviewed in the following can be found in (Przymusinski 1991b; Eiter, Leone, and Saccà 1997).

A *3-valued interpretation* of a logic program $\mathcal{P}$ is a pair $\langle \mathcal{T}, \mathcal{F} \rangle$, where $\mathcal{T}, \mathcal{F} \subseteq \mathcal{HB}_\mathcal{P}$ and $\mathcal{T} \cap \mathcal{F} = \emptyset$.

Atoms in $\mathcal{T}$ are considered TRUE, atoms in $\mathcal{F}$ FALSE. All other atoms in $\mathcal{U} = \mathcal{HB}_\mathcal{P} \backslash (\mathcal{T} \cup \mathcal{F})$ are UNDEFINED. A NAF literal $not \ a$ is TRUE iff a is FALSE, $not \ a$ is FALSE iff a is TRUE, and $not \ a$ is UNDEFINED iff a is UNDEFINED.

The *truth value* of $l \in Lit_\mathcal{P}$ with respect to $\langle \mathcal{T}, \mathcal{F} \rangle$ is denoted $val(l)$, where:

- $val(l) = T$, if l is TRUE;
- $val(l) = F$, if l is FALSE;
- $val(l) = U$, if l is UNDEFINED.

The truth values are ordered by $T > U > F$. The *partial reduct* $\frac{\mathcal{P}}{\langle \mathcal{T}, \mathcal{F} \rangle}$ of $\mathcal{P}$ with respect to a 3-valued interpretation $\langle \mathcal{T}, \mathcal{F} \rangle$ is obtained by replacing each NAF literal in every clause of $\mathcal{P}$ by its truth value. Based on this, the semantics of a logic program is defined as follows:

- A 3-valued interpretation $\langle \mathcal{T}, \mathcal{F} \rangle$ *satisfies* a clause $a \leftarrow a_1, \ldots, a_m, not \ a_{m+1}, \ldots, not \ a_{m+n}$ if $val(a) \geq min\{val(a_1), \ldots, val(not \ a_{m+n})\}$.
  $\langle \mathcal{T}, \mathcal{F} \rangle$ *satisfies* $a \leftarrow$ if $val(a) = T$.
- A 3-valued interpretation $\langle \mathcal{T}, \mathcal{F} \rangle$ of $\mathcal{P}$ is a *3-valued model* of $\mathcal{P}$ if $\langle \mathcal{T}, \mathcal{F} \rangle$ satisfies every clause in $\mathcal{P}$.
- A 3-valued model $\langle \mathcal{T}, \mathcal{F} \rangle$ of $\mathcal{P}$ is a *3-valued stable model* of $\mathcal{P}$ if it is the minimal 3-valued model of $\frac{\mathcal{P}}{\langle \mathcal{T}, \mathcal{F} \rangle}$.
- A 3-valued stable model $\langle \mathcal{T}, \mathcal{F} \rangle$ of $\mathcal{P}$ is a *3-valued L-stable model* (Least-undefined) of $\mathcal{P}$ if $\mathcal{U}$ is minimal (w.r.t. set inclusion) among all 3-valued stable models of $\mathcal{P}$.

Example 1. Let $\mathcal{P}_1$ be the following logic program:
$k \leftarrow not \ p$; $p \leftarrow not \ k$; $r \leftarrow not \ r$; $r \leftarrow not \ r, not \ k$.
$\mathcal{P}_1$ has three 3-valued stable models: $\langle \mathcal{T}_1 = \{p\}, \mathcal{F}_1 = \{k\}\rangle$, $\langle \mathcal{T}_2 = \{k\}, \mathcal{F}_2 = \{p\}\rangle$, $\langle \mathcal{T}_3 = \emptyset, \mathcal{F}_3 = \emptyset\rangle$. The first two are the 3-valued L-stable models of $\mathcal{P}_1$.

Assumption-Based Argumentation

An *Assumption-Based Argumentation (ABA) framework* (Bondarenko et al. 1997; Dung, Kowalski, and Toni 2009) is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$, where:

- $(\mathcal{L}, \mathcal{R})$ is a deductive system, with $\mathcal{L}$ a language and $\mathcal{R}$ a set of inference rules[1] of the form $s_0 \leftarrow s_1, \ldots, s_n$ $(n \geq 0)$ with $s_0, \ldots, s_n \in \mathcal{L}$;
- $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set of *assumptions*;
- ${}^-$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$ defining the *contrary* of assumptions, where $\overline{\alpha}$ denotes the contrary of $\alpha \in \mathcal{A}$.

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

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

We call a set of assumptions $AP \subseteq \mathcal{A}$ an *argument-supporting* set iff there exists an argument $AP \vdash s$ for some $s \in \mathcal{L}$. Let $Asms, Asms_1 \subseteq \mathcal{A}$ and $\alpha \in \mathcal{A}$.

- $Asms$ *attacks* α iff there exists an argument $AP \vdash \overline{\alpha}$ such that $AP \subseteq Asms$. $Asms$ attacks $Asms_1$ iff $Asms$ attacks some $\alpha \in Asms_1$.
- $Asms^+ = \{\alpha \in \mathcal{A} \mid Asms \text{ attacks } \alpha\}$ consists of all assumptions that $Asms$ attacks.
- $Asms$ *defends* α iff $Asms$ attacks all sets of assumptions attacking α.
- $Asms$ is a *complete assumption extension* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff $Asms$ consists of all assumptions it defends and $Asms$ does not attack itself.
- $Asms$ is a *semi-stable assumption extension* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff it is a complete assumption extension of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ and $Asms \cup Asms^+$ is maximal (w.r.t. set inclusion) among all complete assumption extensions of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ (Caminada et al. 2013).

Example 2. Consider the ABA framework ABA_1 with:

- $\mathcal{L} = \{k, p, r, \kappa, \pi, \rho\}$
- $\mathcal{R} = \{k \leftarrow \pi;\ p \leftarrow \kappa;\ r \leftarrow \rho;\ r \leftarrow \rho, \kappa\}$
- $\mathcal{A} = \{\kappa, \pi, \rho\};\ \ \overline{\kappa} = k;\ \overline{\pi} = p;\ \overline{\rho} = r$

In this framework $\{\pi\}$ (and any superset thereof) attacks κ, $\{\kappa\}$ (and any superset thereof) attacks π, and $\{\rho\}$ (and any superset) attacks ρ. ABA_1 has three complete assumption extensions: $Asms_1 = \{\kappa\}$, $Asms_2 = \{\pi\}$, $Asms_3 = \emptyset$. Then,

[1] Using the same notation $\leftarrow$ for clauses in a logic program and rules in ABA will facilitate the presentation of our results.

- $Asms_1^+ = \{\pi\},\ Asms_1 \cup Asms_1^+ = \{\kappa, \pi\}$;
- $Asms_2^+ = \{\kappa\},\ Asms_2 \cup Asms_2^+ = \{\kappa, \pi\}$;
- $Asms_3^+ = \emptyset,\ Asms_3 \cup Asms_3^+ = \emptyset$.

Thus, both $Asms_1$ and $Asms_2$ are semi-stable assumption extensions of ABA_1.

Inspired by the labelling approach for AA semantics (Dung 1995; Caminada and Gabbay 2009), a labelling version of the complete semantics was recently introduced for ABA and was proven equivalent to the complete assumption extension semantics (Schulz and Toni 2014a), where the assumptions labelled IN coincide with the extensions.

An *assumption labelling* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ is a total function $LabAsm : \mathcal{A} \to \{\text{IN}, \text{OUT}, \text{UNDEC}\}$.

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

$LabAsm$ is a *complete assumption labelling* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff for each assumption $\alpha \in \mathcal{A}$ it holds that:

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

Example 3. ABA_1 has three complete assumption labellings, coinciding with the three assumption extensions given in Example 2:

- $\text{IN}(LabAsm_1) = \{\kappa\},\ \text{OUT}(LabAsm_1) = \{\pi\},\ \text{UNDEC}(LabAsm_1) = \{\rho\}$;
- $\text{IN}(LabAsm_2) = \{\pi\},\ \text{OUT}(LabAsm_2) = \{\kappa\},\ \text{UNDEC}(LabAsm_2) = \{\rho\}$;
- $\text{IN}(LabAsm_3) = \emptyset,\ \text{OUT}(LabAsm_3) = \emptyset,\ \text{UNDEC}(LabAsm_3) = \{\kappa, \pi, \rho\}$.

3 Logic Programming encoded in ABA

Even though the semantics of a logic program and of an ABA framework are determined in completely different ways, they share structural features. Both represent knowledge in terms of if-then statements comprising defeasible elements, i.e. elements which are true by default as long as no contrary information can be proven to hold: NAF literals in logic programs and assumptions in ABA frameworks. Every assumption α has a contrary $\overline{\alpha} = x$, where x could also be the contrary of other assumptions. A NAF literal $not\ a$ has a complement a, but in contrast to contraries in ABA, a is the complement of only one NAF literal (namely of $not\ a$). Therefore, a logic program can be seen as a special instance of an ABA framework which means that every logic program can be encoded in an ABA framework.

Definition 1 ((Bondarenko et al. 1997))**.** Let $\mathcal{P}$ be a logic program with Herbrand Base $\mathcal{HB}_\mathcal{P}$. The *corresponding ABA framework* of $\mathcal{P}$ is $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ where

- $\mathcal{L} = Lit_\mathcal{P}$;
- $\mathcal{R} = \mathcal{P}$;
- $\mathcal{A} = \neg.\mathcal{HB}_\mathcal{P}$;
- for all $not\ a \in \mathcal{A}$: $\overline{not\ a} = a$.

Example 4. The corresponding ABA framework of $\mathcal{P}_1$ (Example 1), called $ABA_{\mathcal{P}_1}$, is equivalent to ABA_1 (Examples 2,3), where $not\ k$ substitutes κ, $not\ p$ substitutes π, and $not\ r$ substitutes ρ. Thus, the semantics of $ABA_{\mathcal{P}_1}$ and ABA_1 correspond, for example the complete assumption labellings of $ABA_{\mathcal{P}_1}$ are $\text{IN}(LabAsm_1) = \{not\ k\}$, $\text{OUT}(LabAsm_1) = \{not\ p\}, \text{UNDEC}(LabAsm_1) = \{not\ r\}$ etc.

3-Valued Stable Models vs. Complete Labellings

We now show that the 3-valued stable models of a logic program correspond to the complete assumption labellings of the corresponding ABA framework.

Theorem 1. *Let $\mathcal{P}$ be a logic program, $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ the corresponding ABA framework of $\mathcal{P}$, and $LabAsm$ an assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$. If $LabAsm$ is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$, then $\langle \mathcal{T}, \mathcal{F} \rangle$ with $\mathcal{T} = \neg.\text{OUT}(LabAsm)$ and $\mathcal{F} = \neg.\text{IN}(LabAsm)$ is a 3-valued stable model of $\mathcal{P}$, where $\mathcal{U} = \neg.\text{UNDEC}(LabAsm)$.*

Proof. Let $\vdash_{MP}$ denote modus ponens, where NAF literals are treated like atoms.

- By Theorem 2 in (Schulz and Toni 2014a): $\text{IN}(LabAsm)$ is a complete assumption extension.
- By Theorem 5.9 in (Bondarenko et al. 1997): $\mathcal{P} \cup \text{IN}(LabAsm)$ is a complete scenario of $\mathcal{P}$ as defined by (Dung 1991).
- By Corollary 4.16(i) in (Brogi et al. 1992): $E = \mathcal{P} \cup \text{IN}(LabAsm) \cup \{\neg not\ a \mid a \in \mathcal{HB}_\mathcal{P}, P \cup \text{IN}(LabAsm) \vdash_{MP} a\}$ is a stationary expansion of $\mathcal{P}$ as defined by (Przymusinski 1991a).
- By Theorem 3.1 in (Przymusinski 1991a): $M = \{a \mid E \vdash_{MP} a\} \cup \{not\ a \mid E \vdash_{MP} not\ a\}$ is a partial stable model of $\mathcal{P}$ as defined in (Przymusinski 1991b).
- $\{a | E \vdash_{MP} a\}$ is equivalent to $\{a | \mathcal{P} \cup \text{IN}(LabAsm) \vdash_{MP} a\}$ and $\{not\ a \mid E \vdash_{MP} not\ a\}$ to $\{not\ a \mid \mathcal{P} \cup \text{IN}(LabAsm) \vdash_{MP} not\ a\}$. Thus, $M = \{a \mid \mathcal{P} \cup \text{IN}(LabAsm) \vdash_{MP} a\} \cup \text{IN}(LabAsm)$.
- By Proposition 3.2 in (Przymusinski 1991b): $M = \langle \mathcal{T}, \mathcal{F} \rangle$ with $\mathcal{T} = \{a \mid \mathcal{P} \cup \text{IN}(LabAsm) \vdash_{MP} a\}$ and $\mathcal{F} = \neg.\text{IN}(LabAsm)$ is a 3-valued stable model of $\mathcal{P}$.
- By definition of ABA arguments and complete assumption labellings: $\mathcal{T} = \{a \mid AP \vdash a, AP \subseteq \text{IN}(LabAsm)\} = \neg.\{not\ a \mid AP \vdash a, AP \subseteq \text{IN}(LabAsm)\} = \neg.\text{OUT}(LabAsm)$
- By definition of 3-valued model: $\mathcal{U} = \mathcal{HB}_\mathcal{P} \setminus (\mathcal{T} \cup \mathcal{F}) = \mathcal{HB}_\mathcal{P} \setminus (\neg.\text{OUT}(LabAsm) \cup \neg.\text{IN}(LabAsm)) = \neg.\mathcal{A} \setminus \neg.(\text{OUT}(LabAsm) \cup \text{IN}(LabAsm))$ (by Def. 1) $= \neg.\text{UNDEC}(LabAsm)$ □

Theorem 2. *Let $\mathcal{P}$ be a logic program, $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ the corresponding ABA framework of $\mathcal{P}$, and $\langle \mathcal{T}, \mathcal{F} \rangle$ a 3-valued interpretation of $\mathcal{P}$. If $\langle \mathcal{T}, \mathcal{F} \rangle$ is a 3-valued stable model of $\mathcal{P}$ then $LabAsm$ with $\text{IN}(LabAsm) = \neg.\mathcal{F}$, $\text{OUT}(LabAsm) = \neg.\mathcal{T}$, and $\text{UNDEC}(LabAsm) = \neg.\mathcal{U}$ is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$.*

Proof. Let $\vdash_{MP}$ denote modus ponens, where NAF literals are treated like atoms.

- By definition of 3-valued stable model: $M = \mathcal{T} \cup \neg.\mathcal{F}$ is a partial stable model of $\mathcal{P}$ as defined in (Przymusinski 1991b).
- By Theorem 3.1 in (Przymusinski 1991a) $E = \mathcal{P} \cup \{not\ a \mid not\ a \in M\} \cup \{\neg not\ a \mid a \in M\}$ is a stationary expansion of $\mathcal{P}$.
- By Corollary 4.16(ii) in (Brogi et al. 1992): $\mathcal{P} \cup (E \cap \neg.\mathcal{HB}_\mathcal{P})$ is a complete scenario of $\mathcal{P}$.
- By Theorem 5.9 in (Bondarenko et al. 1997): $E \cap \neg.\mathcal{HB}_\mathcal{P}$ is a complete assumption extension.
- This can be simplified to $\{not a | not a \in M\}$ is a complete assumption extension, and further to $\neg.\mathcal{F}$ is a complete assumption extension.
- By Theorem 2 in (Schulz and Toni 2014a): $\text{IN}(LabAsm) = \neg.\mathcal{F}$
- By Theorem 2 in (Schulz and Toni 2014a): $\text{OUT}(LabAsm) = \{not\ a \mid AP \vdash a, AP \subseteq \neg.\mathcal{F}\} = \{not\ a \,|\, \mathcal{P} \cup \neg.\mathcal{F} \vdash_{MP} a\} = \{not\ a \,|\, \mathcal{P} \cup \{not\ b \,|\, not\ b \in M\} \vdash_{MP} a\} = \{not\ a \mid a \in \mathcal{T}\} = \neg.\mathcal{T}$
- By Theorem 2 in (Schulz and Toni 2014a): $\text{UNDEC}(LabAsm) = \mathcal{A} \setminus (\text{IN}(LabAsm) \cup \text{OUT}(LabAsm)) = \neg.\mathcal{HB}_\mathcal{P} \setminus (\neg.\mathcal{F} \cup \neg.\mathcal{T}) = \neg.\mathcal{U}$ □

From Theorems 1 and 2 and Theorem 2 in (Schulz and Toni 2014a) it follows directly that the 3-valued stable models of a logic program and the complete assumption extensions of the corresponding ABA framework coincide, too.

Example 5. There is a direct correspondence between $\langle \mathcal{T}_1, \mathcal{F}_1 \rangle$ and $LabAsm_1$, $\langle \mathcal{T}_2, \mathcal{F}_2 \rangle$ and $LabAsm_2$, and $\langle \mathcal{T}_3, \mathcal{F}_3 \rangle$ and $LabAsm_3$ of $\mathcal{P}_1$ and $ABA_{\mathcal{P}_1}$ (Examples 1,3).

This correspondence between LP and ABA semantics has significant impact. Firstly, it can be useful for approaches like (Schulz and Toni 2013; 2014b) where ABA is used to explain answer sets of an extended logic program (logic programs with both NAF and strong negation) and which rely on the correspondence of the answer set semantics in LP and the stable semantics in ABA. Secondly, novices in LP have the possibility to view LP from a different perspective, namely an argumentative one, and thereby gain a better understanding of it. Even for LP experts, thinking about the semantics of a logic program in terms of ABA semantics can provide useful insight, especially because assumption labellings can be easily represented in a graph (Section 4), making the LP semantics graphically accessible through the corresponding ABA framework.

L-Stable Models vs. Semi-Stable Labellings

Inspired by the definition of semi-stable argument labelling in AA which is a complete argument labelling (Caminada and Gabbay 2009) with a minimal set of undecided arguments, we now define a labelling version of the semi-stable semantics in ABA as complete assumption labelling with a minimal set of undecided assumptions. We prove correspondence of this new semi-stable assumption labelling with the semi-stable assumption extension semantics in ABA as well as with the 3-valued L-stable model semantics in LP. These results extend the correspondence of LP and ABA semantics, and its impact explained in the previous paragraph.

Definition 2. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework and let $LabAsm$ be an assumption labelling. $LabAsm$ is a *semi-stable assumption labelling* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff it is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ where $\text{UNDEC}(LabAsm)$ is minimal (w.r.t. set inclusion) among all complete assumption labellings of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$.

The following theorems prove the correspondence between semi-stable assumption labellings and semi-stable assumption extensions of an ABA framework.

Theorem 3. *Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework, $LabAsm$ an assumption labelling, and $Asms \subseteq \mathcal{A}$ a set of assumptions. $LabAsm$ is a semi-stable assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff $Asms = \text{IN}(LabAsm)$ is a semi-stable assumption extension of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ with $Asms^+ = \text{OUT}(LabAsm)$ and $\mathcal{A} \setminus (Asms \cup Asms^+) = \text{UNDEC}(LabAsm)$.*

Proof. By Theorem 2 in (Schulz and Toni 2014a) and since maximising $Asms \cup Asms^+$ is equivalent to minimising $\mathcal{A} \setminus Asms \cup Asms^+$. □

Example 6. $ABA_{\mathcal{P}_1}$ has two semi-stable assumption labellings, $LabAsm_1$ and $LabAsm_2$, which correspond to the two semi-stable assumption extensions $Asms_1$ and $Asms_2$ (Examples 2,3).

We now show that the 3-valued L-stable models of a logic program correspond to the semi-stable assumption labellings of the corresponding ABA framework.

Theorem 4. *Let $\mathcal{P}$ be a logic program, $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ the corresponding ABA framework of $\mathcal{P}$, and $LabAsm$ an assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$. If $LabAsm$ is a semi-stable assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ then $\langle \mathcal{T}, \mathcal{F} \rangle$ with $\mathcal{T} = \neg.\text{OUT}(LabAsm)$ and $\mathcal{F} = \neg.\text{IN}(LabAsm)$ is a 3-valued L-stable model of $\mathcal{P}$, where $\mathcal{U} = \neg.\text{UNDEC}(LabAsm)$.*

Proof. By Theorem 1 and since minimising $\text{UNDEC}(LabAsm)$ is equivalent to minimising $\neg.\text{UNDEC}(LabAsm)$. □

Theorem 5. *Let $\mathcal{P}$ be a logic program, $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ the corresponding ABA framework of $\mathcal{P}$, and $\langle \mathcal{T}, \mathcal{F} \rangle$ a 3-valued interpretation of $\mathcal{P}$. If $\langle \mathcal{T}, \mathcal{F} \rangle$ is a 3-valued L-stable model of $\mathcal{P}$ then $LabAsm$ with $\text{IN}(LabAsm) = \neg.\mathcal{F}$, $\text{OUT}(LabAsm) = \neg.\mathcal{T}$, and $\text{UNDEC}(LabAsm) = \neg.\mathcal{U}$ is a semi-stable assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$.*

Proof. By Theorem 2 and since minimising $\mathcal{U}$ is equivalent to minimising $\neg.\mathcal{U}$. □

From Theorems 3 to 5 it follows directly that the 3-valued L-stable models of a logic program also correspond to the semi-stable assumption extensions of the corresponding ABA framework.

Example 7. The two 3-valued L-stable models of $\mathcal{P}_1$, $\langle \mathcal{T}_1, \mathcal{F}_1 \rangle$ and $\langle \mathcal{T}_2, \mathcal{F}_2 \rangle$, correspond to the two semi-stable assumption labellings and extensions of $ABA_{\mathcal{P}_1}$ (Examples 1,6).

4 Graphical representation of assumption labellings

In this section, we show how to represent an ABA framework graphically, which has not been done in the ABA literature before, and how to display its complete labellings. This is particularly interesting when using ABA to encode a logic program, as the correspondence of LP and ABA semantics proven in previous sections implies that the semantics of a logic program can be represented graphically in argumentative terms. Since the semantics of an ABA framework are based on attacking sets of assumption, an ABA framework can be represented as a graph with sets of assumptions as nodes, and edges between them indicating attacks. Figure 1 illustrates all sets of assumptions and attacks between them of $ABA_{\mathcal{P}_1}$ and indicates the labels of assumptions according to the three complete assumption labellings of $ABA_{\mathcal{P}_1}$. However, the large number of sets of assumptions and attacks makes this graph rather complicated and unclear.

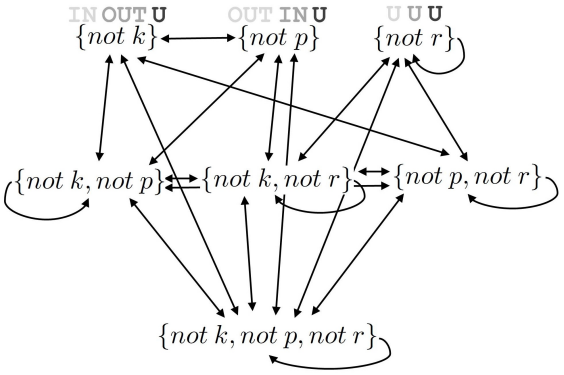

Figure 1: Attacks between all sets of assumption in $ABA_{\mathcal{P}_1}$ along with the three complete assumption labellings (see Example 3), indicated by the three differently coloured letters above the singleton sets, where "U" is shorthand for UNDEC.

For this reason, we prove that considering all sets of assumptions is equivalent to examining only argument-supporting sets of assumptions of an ABA framework when determining complete assumption labellings. This means that it is also sufficient to display only argument-supporting sets of assumptions in a graph, resulting in a clearer graphical representation.

Definition 3. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ be an ABA framework and let $Asms \subseteq \mathcal{A}$. $Asms_{sets} = \{AP \mid AP \subseteq Asms, \exists s \in \mathcal{L} \text{ s.t. } AP \vdash s\}$ is the set of all argument-supporting subsets of $Asms$.

Note that all assumptions in $Asms$ occur as singleton sets in $Asms_{sets}$, i.e. for every $\alpha \in Asms$: $\{\alpha\} \in Asms_{sets}$ since for any assumption α, $\{\alpha\} \vdash \alpha$ is an argument.

Example 8. Given $\mathcal{A} = \{not\ k, not\ p, not\ r\}$ in $ABA_{\mathcal{P}_1}$, $\mathcal{A}_{sets} = \{\{not\ k\}, \{not\ p\}, \{not\ r\}, \{not\ k, not\ r\}\}$ is the set of all argument-supporting subsets of $\mathcal{A}$.

Definition 4. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ be an ABA framework and let $LabAsm$ be an assumption labelling. $LabAsm$ is a *complete assumption labelling w.r.t. argument-supporting sets* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ iff for each assumption $\alpha \in \mathcal{A}$ it holds that:

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

Lemma 6. *Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ be an ABA framework and let $LabAsm$ be an assumption labelling. $LabAsm$ is a complete assumption labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ iff $LabAsm$ is a complete assumption labelling w.r.t. argument-supporting sets of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$.*

Proof (Sketch). Based on the ideas that (1) if an argument-supporting set of assumptions attacks an assumption then all its supersets also attack this assumption and (2) if a set of assumptions attacks an assumption then there exists an argument-supporting set of assumptions attacking this assumption (see Appendix for the full proof). □

It follows from Lemma 6 that an ABA framework can be represented equivalently as a graph of all sets of assumptions or as a graph of only argument-supporting sets of assumptions. Figure 2 illustrates the graph of only argument-supporting sets of assumptions and their attacks in $ABA_{\mathcal{P}_1}$ and indicates the three complete assumption labellings (equivalently the three complete assumption labellings w.r.t. argument-supporting sets of $ABA_{\mathcal{P}_1}$). This graph is considerably clearer than the graph containing all sets of assumptions in Figure 1. Since $ABA_{\mathcal{P}_1}$ encodes $\mathcal{P}_1$, the graph in Figure 2 in fact represents the logic program $ABA_{\mathcal{P}_1}$. Furthermore, as complete assumption labellings correspond to 3-valued stable models, the labellings shown in Figure 2 represent the 3-valued stable models of $\mathcal{P}_1$ in terms of the truth values of the NAF literals in the program. This graphical representation of a logic program can be useful for both novices and experts to better understand a program's structure, in particular with respect to the dependencies between literals.

Sets of Assumptions

Sets of assumptions play an important role in the semantics of an ABA framework. Thus, another labelling idea for ABA is to assign labels to whole sets of assumptions rather than to single assumptions. Due to the results from Lemma 6, we only consider argument-supporting sets of assumptions for this new labelling approach.

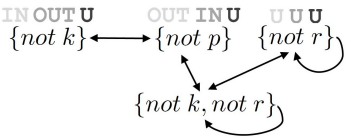

Figure 2: Attacks between argument-supporting sets of assumption in $ABA_{\mathcal{P}_1}$ along with the three complete assumption labellings (see Example 3).

Definition 5. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework. An *assumption-set labelling* is a total function $LabAsmSet : \mathcal{A}_{sets} \rightarrow \{\texttt{IN}, \texttt{OUT}, \texttt{UNDEC}\}$.

$\texttt{IN}(LabAsmSet)$ consists of all sets of assumptions labelled IN, $\texttt{OUT}(LabAsmSet)$ of all sets labelled OUT, and $\texttt{UNDEC}(LabAsmSet)$ of all sets labelled UNDEC.

We define the label of a set of assumptions in a complete assumption-set labelling based on the labels with respect to a complete assumption labelling of the assumptions contained in this set. Let the labels of an assumption labelling be ordered by IN $>$ UNDEC $>$ OUT.

Definition 6. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework and let $LabAsmSet$ be an assumption-set labelling. $LabAsmSet$ is a complete assumption-set labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ if there exists a complete assumption labelling $LabAsm$ of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ such that $\forall AP \in \mathcal{A}_{sets}$: $LabAsmSet(AP) = min\{LabAsm(\alpha) \mid \alpha \in AP\}$

Following the labelling approach of single assumptions, we also define a semi-stable version of the assumption-set labelling.

Definition 7. Let $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ be an ABA framework and let $LabAsmSet$ be an assumption-set labelling. $LabAsmSet$ is a *semi-stable assumption-set labelling* of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ iff it is a complete assumption-set labelling of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$ where $\texttt{UNDEC}(LabAsmSet)$ is minimal (w.r.t. set inclusion) among all complete assumption-set labellings of $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, {}^- \rangle$.

Example 9. $ABA_{\mathcal{P}_1}$ has three complete assumption-set labellings:

- $\texttt{IN}(LabAsmSet_1) = \{\{not\ k\}\}$,
  $\texttt{OUT}(LabAsmSet_1) = \{\{not\ p\}\}$,
  $\texttt{UNDEC}(LabAsmSet_1) = \{\{not\ r\}, \{not\ k, not\ r\}\}$;
- $\texttt{IN}(LabAsmSet_2) = \{\{not\ p\}\}$,
  $\texttt{OUT}(LabAsmSet_2) = \{\{not\ k\}, \{not\ k, not\ r\}\}$,
  $\texttt{UNDEC}(LabAsmSet_2) = \{\{not\ r\}\}$;
- $\texttt{IN}(LabAsmSet_3) = \emptyset$, $\texttt{OUT}(LabAsmSet_3) = \emptyset$,
  $\texttt{UNDEC}(LabAsmSet_3) = \{\{not\ k\}, \{not\ p\}, \{not\ r\}, \{not\ k, not\ r\}\}$.

Since $\texttt{UNDEC}(LabAsmSet_2)$ is a subset of both $\texttt{UNDEC}(LabAsmSet_1)$ and $\texttt{UNDEC}(LabAsmSet_3)$, only $LabAsmSet_2$ is a semi-stable assumption-set labelling of $ABA_{\mathcal{P}_1}$.

Since assumption-set labellings are defined on sets of argument-supporting assumptions, they can be represented on the same ABA graph as assumption labellings. Figure 3 illustrates the three complete assumption-set labellings of $ABA_{\mathcal{P}_1}$. Comparing this to Figure 2 illustrates why $ABA_{\mathcal{P}_1}$ has only one semi-stable assumption-set labelling, but two semi-stable assumption labellings: The set $\{not\ k, not\ r\}$ is taken into account for the assumption-set labelling, but not for the assumption labelling.

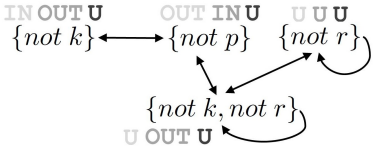

Figure 3: The three complete assumption-set labellings of $ABA_{\mathcal{P}_1}$ (see Example 9).

Since arguments are deductions from argument-supporting sets of assumptions, every argument-supporting set represents at least one argument. Thus, the new assumption-set labelling basically assigns labels to arguments. In fact, complete (semi-stable) assumption-set labellings correspond to complete (respectively semi-stable) argument labellings in AA (Dung 1995; Caminada and Gabbay 2009). Detailed results are omitted for space reasons. However, this implies that a graph of argument-supporting sets of assumptions can illustrate both ABA semantics and AA semantics. Since an ABA framework can encode a logic program this further implies that the graphical representation of an ABA framework can be used to display the structure of a logic program along with its LP semantics as well as with the outcome of applying AA semantics to this logic program. This is particularly interesting in the case of the semi-stable semantics in AA, which does not generally correspond to the 3-valued L-stable semantics in LP (Caminada, Sá, and Alcântara 2013).

5 Conclusions

We extended existing work on the correspondence of logic programming and Assumption-Based Argumentation (ABA) semantics (Bondarenko et al. 1997) by proving that the 3-valued stable models of a logic program correspond to the complete assumption labellings (and thus also to the complete assumption extensions) of an ABA framework encoding this logic program. Furthermore, we defined semi-stable assumption labellings inspired by the notion of semi-stable labellings in Abstract Argumentation (AA) (Caminada and Gabbay 2009) and proved that the 3-valued L-stable models of a logic program correspond to the semi-stable assumption labellings of the encoding ABA framework. These results are useful for both novices and experts in logic programming as they provide a different view on these logic programming semantics in argumentative terms.

Furthermore, we show how the complete and semi-stable assumption labellings of an ABA framework can be displayed on a graph of attacking sets of argument-supporting assumptions. Thus, the 3-valued (L-)stable models of a logic program can be easily illustrated in terms of the labelling

graph of the ABA framework encoding this logic program. These graphs also allow to show the results of applying AA semantics to the logic program, which is particularly interesting in the case of the semi-stable semantics as it does not generally correspond to the 3-valued L-stable model semantics of a logic program.

The correspondence results are not only of theoretical interest. Recently, ABA has been used to explain answer sets (Gelfond and Lifschitz 1991) of a logic program, based on the correspondence between the stable semantics of a logic program and the stable semantics of an ABA framework encoding this logic program (Schulz and Toni 2013; 2014b). The results presented here can therefore inspire similar explanation methods for different logic programming semantics.

6 Appendix

Proof (of Lemma 6). We prove both directions.

- From left to right: Let $LabAsm$ be a complete assumption labelling.
  - If $LabAsm(\alpha) = \text{IN}$ then each set of assumptions AP attacking α contains some β such that $LabAsm(\beta) = \text{OUT}$, in particular every $AP \in \mathcal{A}_{sets}$.
  - If $LabAsm(\alpha) = \text{OUT}$ then there exists a set of assumptions AP attacking α such that $AP \subseteq \text{IN}(LabAsm)$. Then there exists an argument $AP' \vdash \overline{\alpha}$ attacking α such that $AP' \subseteq AP$. Therefore, $AP' \in \mathcal{A}_{sets}$ and $AP' \subseteq \text{IN}(LabAsm)$.
  - If $LabAsm(\alpha) = \text{UNDEC}$ then each set of assumptions AP attacking α contains some β such that $LabAsm(\beta) \neq \text{IN}$, in particular every $AP \in \mathcal{A}_{sets}$. Furthermore there exists a set of assumptions AP_1 attacking α such that $AP_1 \cap \text{OUT}(LabAsm) = \emptyset$. Then there exists an argument $AP_1' \vdash \overline{\alpha}$ attacking α such that $AP_1' \subseteq AP_1$. Therefore, $AP_1' \in \mathcal{A}_{sets}$ and $AP_1' \cap \text{OUT}(LabAsm) = \emptyset$.
- From right to left: Let $LabAsm$ be a complete assumption labelling w.r.t. argument-supporting sets.
  - If $LabAsm(\alpha) = \text{IN}$ then each set of assumptions $AP \in \mathcal{A}_{sets}$ attacking α contains some β such that $LabAsm(\beta) = \text{OUT}$. Since every AP' attacking α is a superset of some $AP \in \mathcal{A}_{sets}$ attacking α, every such AP' contains some β such that $LabAsm(\beta) = \text{OUT}$.
  - If $LabAsm(\alpha) = \text{OUT}$ then there exists a set of assumptions $AP \in \mathcal{A}_{sets}$ attacking α such that $AP \subseteq \text{IN}(LabAsm)$, satisfying the OUT-condition of a complete assumption labelling.
  - If $LabAsm(\alpha) = \text{UNDEC}$ then each set of assumptions in $\mathcal{A}_{sets}$ attacking α contains some β such that $LabAsm(\beta) \neq \text{IN}$. Since every AP' attacking α is a superset of some $AP \in \mathcal{A}_{sets}$ attacking α, every such AP' contains some β such that $LabAsm(\beta) \neq \text{IN}$. Furthermore there exists a set of assumptions $AP_1 \in \mathcal{A}_{sets}$ attacking α such that $AP_1 \cap \text{OUT}(LabAsm) = \emptyset$, satisfying the UNDEC-condition of a complete assumption labelling. □

References

Baral, C., and Gelfond, M. 1994. Logic programming and knowledge representation. *The Journal of Logic Programming* 1920, Supplement 1(0):73 – 148.

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

Brogi, A.; Lamma, E.; Mancarella, P.; and Mello, P. 1992. Normal logic programs as open positive programs. In *JICSLP'92*.

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

Caminada, M.; Sá, S.; Alcântara, J.; and Dvořák, W. 2013. On the difference between assumption-based argumentation and abstract argumentation. Proceedings of BNAIC'13.

Caminada, M.; Sá, S.; and Alcântara, J. 2013. On the equivalence between logic programming semantics and argumentation semantics. In *ECSQARU'13*.

Dung, P.; Kowalski, R.; and Toni, F. 2009. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*. Springer US.

Dung, P. 1991. Negations as hypotheses: An abductive foundation for logic programming. In *ICLP'91*.

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

Eiter, T.; Leone, N.; and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence*.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *ICLP/SLP'88*, 1070–1080.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–386.

Przymusinski, T. 1989. Every logic program has a natural stratification and an iterated least fixed point model. In *PODS'89*.

Przymusinski, T. 1991a. Semantics of disjunctive logic programs and deductive databases. In *Deductive and Object-Oriented Databases*. Springer Berlin Heidelberg.

Przymusinski, T. 1991b. Stable semantics for disjunctive programs. *New Generation Computing*.

Schulz, C., and Toni, F. 2013. Aba-based answer set justification. *TPLP, On-line Supplement* 13(4-5).

Schulz, C., and Toni, F. 2014a. Complete assumption labellings. In *COMMA'14*.

Schulz, C., and Toni, F. 2014b. Justifying answer sets using argumentation. *TPLP*. To appear.

Toni, F. 2014. A tutorial on assumption-based argumentation. *Argument & Computation* 5(1):89–117.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38(3):619–649.