# On the Equivalence between Assumption-Based Argumentation and Logic Programming (Extended Abstract)*

**Martin Caminada**[1], **Claudia Schulz**[2]
[1] School of Computer Science and Informatics, Cardiff University, UK
[2] Ubiquitous Knowledge Processing Lab (UKP-TUDA), Department of Computer Science, Technische Universität Darmstadt, Germany
CaminadaM@cardiff.ac.uk, schulz@ukp.informatik.tu-darmstadt.de

## Abstract

In this work, we explain how Assumption-Based Argumentation (ABA) is subsumed by Logic Programming (LP). The translation from ABA to LP (with a few restrictions on the ABA framework) results in a normal logic program whose semantics coincide with the semantics of the underlying ABA framework. Although the precise technicalities are beyond the current extended abstract (these can be found in the associated full paper) we provide a number of examples to illustrate the general idea.

## 1 Introduction

Assumption-Based Argumentation (ABA) [Bondarenko *et al.*, 1997; Dung *et al.*, 2009; Toni, 2014] has become one of the leading approaches for formal argumentation. 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 can be given in terms of assumption labellings [Schulz and Toni, 2014; 2017], which assign to each assumption a label IN, OUT or UNDEC.

ABA has a well-studied relationship to abstract argumentation (AA) [Dung, 1995], where arguments and attacks between them are given rather than constructed from knowledge and semantics can be defined in terms of sets of argument labellings [Caminada and Gabbay, 2009], in that both flat ABA is an instance of AA [Dung *et al.*, 2007; Toni, 2014; Caminada *et al.*, 2015] and AA is an instance of flat ABA [Toni, 2012] under many well-studied semantics.

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 and 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.

In our work, 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. We here give the main idea of this work, using a number of running examples for illustration, and refer to the full paper [Caminada and Schulz, 2017] for details.

## 2 Assumption-Based Argumentation

An *ABA framework* $\mathcal{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, ^- \rangle$ [Toni, 2014; Bondarenko *et al.*, 1997] consists of a set of inference rules $\mathcal{R}$, based on a formal language $\mathcal{L}$. An important part of this formal language is the set of *assumptions* $\mathcal{A} \subseteq \mathcal{L}$. An assumption represents a defeasible piece of information that is assumed to be true by default, unless contrary information is available. Thus, each assumption is associated (through the function $^-$) with an element of the formal language $\mathcal{L}$ that is called its *contrary*.

As an example, consider the ABA framework $\mathcal{F}_{ex}$ with the set of assumptions $\mathcal{A} = \{\beta, \gamma, \delta, \epsilon, \varphi\}$, inference rules

$$a \leftarrow \qquad b \leftarrow \gamma \qquad c \leftarrow \beta$$
$$d \leftarrow \gamma, \varphi \qquad e \leftarrow a, \delta \qquad f \leftarrow a, \epsilon$$

and contraries

$$\overline{\beta} = b \qquad \overline{\gamma} = c \qquad \overline{\delta} = d$$
$$\overline{\epsilon} = e \qquad \overline{\varphi} = f$$

Based on an ABA framework, one can start to construct *arguments*. These are basically derivations that use the rules of the ABA framework together with the set of assumptions to infer a conclusion. An argument is denoted $Asm \vdash x$, where $Asm$ is the set of all assumptions used in the derivation and $x$ is the conclusion, i.e. the head of the sentence obtained in the last derivation step. For instance, in the above example $\mathcal{F}_{ex}$, one can construct the following arguments: $\emptyset \vdash a$ (applying rule $a \leftarrow$), $\{\gamma\} \vdash b$ (applying rule $b \leftarrow \gamma$), $\{\beta\} \vdash c$ (applying rule $c \leftarrow \beta$), $\{\gamma, \varphi\} \vdash d$ (applying rule $d \leftarrow \gamma, \varphi$), $\{\delta\} \vdash e$ (applying rules $a \leftarrow$ and $e \leftarrow a, \delta$) and $\{\epsilon\} \vdash f$ (applying

*This paper is an extended abstract of an article in the Journal of Artificial Intelligence Research [Caminada and Schulz, 2017].

rules $a \leftarrow$ and $f \leftarrow a, \epsilon$). Apart from the arguments that are constructed using the inference rules, there are also "trivial" arguments that consist of a single assumption. In the example above, these are $\{\beta\} \vdash \beta$, $\{\gamma\} \vdash \gamma$, $\{\delta\} \vdash \delta$, $\{\epsilon\} \vdash \epsilon$ and $\{\varphi\} \vdash \varphi$.

The semantics of an ABA framework can be provided using *assumption labellings* [Schulz and Toni, 2014; 2017]. In essence, the idea is to label each assumption in $\mathcal{A}$ with IN, OUT or UNDEC. An assumption labelling is called a *complete* assumption labelling iff for each assumption $\chi \in \mathcal{A}$ it holds that:

- if $\chi$ is labelled IN then each ABA argument for conclusion $\bar{\chi}$ has at least one assumption that is labelled OUT
- if $\chi$ is labelled OUT then there exists an ABA argument for conclusion $\bar{\chi}$ that has all its assumptions labelled IN
- if $\chi$ is labelled UNDEC then there exists an ABA argument for conclusion $\bar{\chi}$ without any assumption labelled OUT and there does not exist an ABA argument for conclusion $\bar{\chi}$ that has all its assumptions labelled IN

Given a (complete) assumption labelling $\mathcal{L}ab$, we write $\text{IN}(\mathcal{L}ab)$ for the set of assumptions labelled IN, $\text{OUT}(\mathcal{L}ab)$ for the set of assumptions labelled OUT, and $\text{UNDEC}(\mathcal{L}ab)$ for the set of assumptions labelled UNDEC. We will sometimes write a (complete) assumption labelling $\mathcal{L}ab$ as a triplet $(\text{IN}(\mathcal{L}ab), \text{OUT}(\mathcal{L}ab), \text{UNDEC}(\mathcal{L}ab))$.

In the above example of ABA framework $\mathcal{F}_{ex}$, there are three complete assumption labellings: $\mathcal{L}ab_1 = (\{\gamma\}, \{\beta\}, \{\delta, \epsilon, \varphi\})$, $\mathcal{L}ab_2 = (\{\beta, \delta, \varphi\}, \{\gamma, \epsilon\}, \emptyset\})$ and $\mathcal{L}ab_3 = (\emptyset, \emptyset, \{\beta, \gamma, \delta, \epsilon, \varphi\})$.

A complete assumption labelling $\mathcal{L}ab$ of an ABA framework $\mathcal{F}$ is called:

- a *preferred* assumption labelling if $\text{IN}(\mathcal{L}ab)$ is maximal (w.r.t. $\subseteq$) among all complete assumption labellings of $\mathcal{F}$
- a *grounded* assumption labelling if $\text{IN}(\mathcal{L}ab)$ is minimal (w.r.t. $\subseteq$) among all complete assumption labellings of $\mathcal{F}$
- a *semi-stable* assumption labelling if $\text{UNDEC}(\mathcal{L}ab)$ is minimal (w.r.t. $\subseteq$) among all complete assumption labellings of $\mathcal{F}$
- a *stable* assumption labelling if $\text{UNDEC}(\mathcal{L}ab) = \emptyset$
- an *ideal* assumption labelling if $\text{IN}(\mathcal{L}ab)$ is maximal (w.r.t. $\subseteq$) among all complete assumption labellings of $\mathcal{F}$ satisfying that for each preferred assumption labelling $\mathcal{L}ab_{pref}$ of $\mathcal{F}$, $\text{IN}(\mathcal{L}ab) \subseteq \text{IN}(\mathcal{L}ab_{pref})$

In the example of $\mathcal{F}_{ex}$, $\mathcal{L}ab_1$ is a preferred assumption labelling, $\mathcal{L}ab_2$ is a preferred, semi-stable and stable assumption labelling, and $\mathcal{L}ab_3$ is a grounded and ideal assumption labelling.

## 3 From ABA to LP

Given an ABA framework $\mathcal{F}$, it is possible to define an associated (normal) logic program $P_{\mathcal{F}}$ by replacing each assumption $\chi$ in a rule by its negated (using negation-as-failure) contrary, i.e. by `not` $\bar{\chi}$. As an example, when doing this for the ABA framework $\mathcal{F}_{ex}$, the result is the following logic program $P_{\mathcal{F}_{ex}}$.

$$
\begin{array}{lcl}
a & \leftarrow & \\
b & \leftarrow & \texttt{not}\ c \\
c & \leftarrow & \texttt{not}\ b \\
d & \leftarrow & \texttt{not}\ c, \texttt{not}\ f \\
e & \leftarrow & a, \texttt{not}\ d \\
f & \leftarrow & a, \texttt{not}\ e
\end{array}
$$

Given a logic program $P$, different semantics can be defined. For current purposes, we apply the concept of a 3-valued interpretation $\langle T, F \rangle$ where $T$ and $F$ are subsets of the atoms occurring in the logic program, with $T \cap F = \emptyset$. Given a logic program $P$ and a 3-valued interpretation $\langle T, F \rangle$, one can define the reduced logic program $P^{\langle T,F \rangle}$ by replacing each occurrence of `not` $x$ by `TRUE` if $x \in F$, by `FALSE` if $x \in T$ and by `UNDEFINED` otherwise. The resulting logic program $P^{\langle T,F \rangle}$ does not contain any negation-as-failure, so it has a unique minimal 3-valued model [Przymusinski, 1990]. If this unique minimal 3-valued model is equal to $\langle T, F \rangle$, then $\langle T, F \rangle$ is said to be a *3-valued stable model* of $P$ [Przymusinski, 1990].

As an example, the logic program $P_{\mathcal{F}_{ex}}$ has three 3-valued stable models: $\mathcal{M}od_1 = \langle \{a, b\}, \{c\} \rangle$, $\mathcal{M}od_2 = \langle \{a, c, e\}, \{b, f, d\} \rangle$ and $\mathcal{M}od_3 = \langle \{a\}, \emptyset \rangle$. To verify that for instance $\mathcal{M}od_1$ is a 3-valued stable model of $P_{\mathcal{F}_{ex}}$, we observe that $P_{\mathcal{F}_{ex}}^{\langle \{a,b\},\{c\} \rangle}$ is as follows.

$$
\begin{array}{lcl}
a & \leftarrow & \\
b & \leftarrow & \texttt{TRUE} \\
c & \leftarrow & \texttt{FALSE} \\
d & \leftarrow & \texttt{TRUE}, \texttt{UNDEFINED} \\
e & \leftarrow & a, \texttt{UNDEFINED} \\
f & \leftarrow & a, \texttt{UNDEFINED}
\end{array}
$$

This logic program has the unique minimal 3-valued model $\langle \{a, b\}, \{c\} \rangle$ which is equal to $\mathcal{M}od_1$, hence $\mathcal{M}od_1$ is a 3-valued stable model of $P_{\mathcal{F}_{ex}}$. In a similar way, it can be verified that also $\mathcal{M}od_2$ and $\mathcal{M}od_3$ are 3-valued stable models of $P_{\mathcal{F}_{ex}}$.

A 3-valued stable model $\mathcal{M}od = \langle T, F \rangle$ of logic program $P$ is called:

- a *regular* model if $T$ is maximal (w.r.t. $\subseteq$) among all 3-valued stable models of $P$
- a *well-founded* model if $T$ is minimal (w.r.t. $\subseteq$) among all 3-valued stable models of $P$
- an *L-stable* model if $T \cup F$ is maximal (w.r.t. $\subseteq$) among all 3-valued stable models of $P$
- a *(2-valued) stable* model if $T \cup F$ consists of all the atoms in $P$
- an *ideal* model if $T$ is maximal (w.r.t. $\subseteq$) among all 3-valued stable models of $P$ satisfying that for each regular model $\langle T_{reg}, F_{reg} \rangle$ of $P$, $T \subseteq T_{reg}$

In the example of $P_{ex}$, $\mathcal{M}od_1$ is a regular model, $\mathcal{M}od_2$ is a regular, L-stable and (2-valued) stable model, and $\mathcal{M}od_3$ is a well-founded and ideal model.

One of our main findings is that it is possible convert an assumption labelling $\mathcal{L}ab$ of an ABA framework $\mathcal{F}$ to a 3-valued stable model $\mathcal{M}od$ of the associated logic program $P_{\mathcal{F}}$, and vice versa.

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 (with the additional condition that each of these contraries actually occurs in $P_{\mathcal{F}}$). 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 reduct. That is, we define $\texttt{Lab2Mod}(\mathcal{L}ab)$ (the function that converts a labelling $\mathcal{L}ab$ of $\mathcal{F}$ into a 3-valued stable model $\mathcal{M}od$ of $P_{\mathcal{F}}$) as the unique minimal model of $P_{\mathcal{F}}^{\langle T', F' \rangle}$.

As an example, consider again $\mathcal{F}_{ex}$, which (as we have seen before) has the complete assumption labellings:
$\mathcal{L}ab_1 = (\{\gamma\}, \{\beta\}, \{\delta, \epsilon, \varphi\})$,
$\mathcal{L}ab_2 = (\{\beta, \delta, \varphi\}, \{\gamma, \epsilon\}, \emptyset\})$,
$\mathcal{L}ab_3 = (\emptyset, \emptyset, \{\beta, \gamma, \delta, \epsilon, \varphi\})$
It holds that

- $\texttt{Lab2Mod}(\mathcal{L}ab_1) = \langle \{a, b\}, \{c\} \rangle = \mathcal{M}od_1$ (with $\langle T', F' \rangle = \langle \{b\}, \{c\} \rangle$),
- $\texttt{Lab2Mod}(\mathcal{L}ab_2) = \langle \{a, c, e\}, \{b, f, d\} \rangle = \mathcal{M}od_2$ (with $\langle T', F' \rangle = \langle \{c, e\}, \{b, f, d\} \rangle$), and
- $\texttt{Lab2Mod}(\mathcal{L}ab_3) = \langle \{a\}, \emptyset \rangle = \mathcal{M}od_3$ (with $\langle T', F' \rangle = \langle \emptyset, \emptyset \rangle$).

In this example, the application of the reduct and minimal model thereof adds atom $a$ as a true atom to each interpretation $\langle T', F' \rangle$.

It is also possible to go the other way around. That is, given a 3-valued stable model $\mathcal{M}od$ of $P_{\mathcal{F}}$, one can define the associated complete assumption labelling $\mathcal{L}ab$ of $\mathcal{F}$, using a function called Mod2Lab. The idea is that:

- the assumptions whose contrary is in $F$ are labelled IN
- the assumptions whose contrary is in $T$ are labelled OUT
- the assumptions whose contrary does occur in $P_{\mathcal{F}}$ but neither in $T$ or $F$ are labelled UNDEC
- the assumptions whose contrary does not even occur in $P_{\mathcal{F}}$ are labelled IN as well

As an example, for $\mathcal{F}$ and $P_{\mathcal{F}}$ it can be observed that $\texttt{Mod2Lab}(\mathcal{M}od_1) = \mathcal{L}ab_1$, $\texttt{Mod2Lab}(\mathcal{M}od_2) = \mathcal{L}ab_2$ and $\texttt{Mod2Lab}(\mathcal{M}od_3) = (\mathcal{L}ab_3)$.

We have proven that when the domain of Lab2Mod consists of the complete assumption labellings of $\mathcal{F}$, and the domain of Mod2Lab consists of the 3-valued stable models of $P_{\mathcal{F}}$, Lab2Mod and Mod2Lab are bijective functions that are each other's inverse. Moreover,

1. when $\mathcal{L}ab$ is a preferred assumption labelling of $\mathcal{F}$, $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a regular model of $P_{\mathcal{F}}$, and when $\mathcal{M}od$ is a regular model of $P_{\mathcal{F}}$, $\texttt{Mod2Lab}(\mathcal{M}od)$ is a preferred assumption labelling of $\mathcal{F}$;
2. when $\mathcal{L}ab$ is a stable assumption labelling of $\mathcal{F}$, $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a (2-valued) stable model of $P_{\mathcal{F}}$, and when $\mathcal{M}od$ is a (2-valued) stable model of $P_{\mathcal{F}}$, $\texttt{Mod2Lab}(\mathcal{M}od)$ is a stable assumption labelling of $\mathcal{F}$;
3. when $\mathcal{L}ab$ is a grounded assumption labelling of $\mathcal{F}$, $\texttt{Lab2Mod}(\mathcal{L}ab)$ is a well-founded model of $P_{\mathcal{F}}$, and when $\mathcal{M}od$ is a well-founded model of $P_{\mathcal{F}}$, $\texttt{Mod2Lab}(\mathcal{M}od)$ is a grounded assumption labelling of $\mathcal{F}$;
4. when $\mathcal{L}ab$ is an ideal assumption labelling of $\mathcal{F}$, $\texttt{Lab2Mod}(\mathcal{L}ab)$ is an ideal model of $P_{\mathcal{F}}$, and when $\mathcal{M}od$ is an ideal model of $P_{\mathcal{F}}$, $\texttt{Mod2Lab}(\mathcal{M}od)$ is an ideal assumption labelling of $\mathcal{F}$.

| assumption labelling of $\mathcal{F}$ | logic programming model of $P_{\mathcal{F}}$ |
|---|---|
| complete | 3-valued stable |
| preferred | regular |
| stable | (2-valued) stable |
| grounded | well-founded |
| ideal | ideal |

Table 1: Semantic equivalences when translating ABA to LP

Overall, the results regarding formal correspondences between assumption labellings and logic programming models are summed up in Table 1.

## 4 Limitations

For our translation from ABA to LP to work, there need to be a few restrictions on the ABA framework. The first restriction is that the ABA framework should be *flat* [Toni, 2013]. This means that no assumption appears in the head of an ABA rule. This is important, for if the head of an ABA rule was an assumption, the resulting LP rule would have negation-as-failure in its head, so the result would not be a normal logic program.

The second restriction is that each assumption should have just a single contrary, rather than a set of contraries. This restriction is shared by most of the ABA literature (like [Bondarenko *et al.*, 1997; Dung *et al.*, 2009; 2007; Toni, 2014; Schulz and Toni, 2014]), although some works on ABA (like [Gaertner and Toni, 2007; 2008; Fan and Toni, 2014; 2015]) allow an assumption to have more than one contrary. Our current translation from ABA to LP would not be well-defined in the latter case.

The third restriction is that the contrary of an assumption should not be an assumption itself. Although this restriction is not required for the well-definedness of the resulting logic program, examples exist where violating this restriction results in breaking the link between the assumption labellings of the ABA framework and the logic programming models of the associated logic program.

For the second and third restriction, workarounds exist. For instance, it is possible to convert an ABA framework where assumptions have multiple contraries to an equivalent ABA framework[1] where each assumption has a single contrary [Gaertner and Toni, 2008]. Furthermore, it is possible to

[1]With an equivalent ABA framework we mean an ABA framework that has the same complete (resp. preferred, semi-stable, stable, grounded and ideal) assumption labellings.

convert an ABA framework where the contrary of an assumption may be an assumption to an equivalent ABA framework where this is not the case. Details of the respective translations can be found in the full paper. For an ABA framework that does not comply with the second or third restriction, the workaround idea is to first translate it to an equivalent ABA framework that does comply with the restriction, and then translate this to the associated logic program. The logic programming models (2-valued) stable, well-founded and ideal semantics, respectively) of this logic program will coincide with the assumption labellings of the original ABA framework as previously described.

Another type of restriction is regarding the semantics applied to the ABA framework. Although our translation works for complete, preferred, stable, grounded and ideal assumption labellings (which coincide with the 3-valued stable, regular, (2-valued) stable, well-founded and ideal models, respectively, of the associated logic program through the functions `Lab2Mod` and `Mod2Lab`) it is *not* the case that the semi-stable assumption labellings of an ABA framework coincide with the L-stable models of the associated logic program. However, it can be observed that semi-stable assumption labellings *do* coincide with L-stable models when the ABA framework is *assumption-spanning*, that is, when each non-assumption in the language is the contrary of an assumption.

## 5 From LP to ABA

Apart from the translation from ABA to LP, it is also possible to translate from LP to ABA, as investigated in previous work [Bondarenko *et al.*, 1997; Schulz and Toni, 2015; 2016]. The idea is that, given a logic program, one can translate it to an ABA framework by replacing each occurrence of `not` $x$ by an assumption `not_x`.[2] For instance, for the logic program $P_{ex}$ mentioned above, the corresponding ABA framework is as follows,

$a \leftarrow$
$b \leftarrow$ `not_c` (with $\overline{\texttt{not\_c}} = c$)
$c \leftarrow$ `not_b` (with $\overline{\texttt{not\_b}} = b$)
$d \leftarrow$ `not_c`, `not_f` (with $\overline{\texttt{not\_f}} = f$)
$e \leftarrow a$, `not_d` (with $\overline{\texttt{not\_d}} = d$)
$f \leftarrow a$, `not_e` (with $\overline{\texttt{not\_e}} = e$)

with the set of assumptions in the underlying language being $\{$`not_a`, `not_b`, `not_c`, `not_d`, `not_e`, `not_f`$\}$. Note that the assumption `not_a` does not occur in any of the ABA rules.

We observe that when translating a logic program to an ABA framework and then back to a logic program, the final logic program is precisely the same as the one at the start. This makes it possible to reuse part of our existing technical results from the "ABA to LP" direction to apply to the "LP to ABA" direction (see the full paper for details). In particular, it turns out that the various types of models of a logic program $P$ coincide (through the functions `Mod2Lab` and `Lab2Mod`) with the various types of assumption labellings of the associated ABA framework $\mathcal{F}_P$, thus generalising existing work [Bondarenko *et al.*, 1997; Schulz and Toni, 2015; 2016]. An overview of the semantic correspondence is provided in Table 2.

| logic programming model of $P$ | assumption labelling of $\mathcal{F}_P$ |
|---|---|
| 3-valued stable | complete |
| regular | preferred |
| L-stable | semi-stable |
| (2-valued) stable | stable |
| well-founded | well-founded |
| ideal | ideal |

Table 2: Semantic equivalences when translating from LP to ABA

As indicated in Table 2, for the "LP to ABA" direction, we do obtain equivalence between L-stable models and semi-stable assumption labellings (which is not the case for the "ABA to LP" direction, see Table 1). This is because translating a logic program results in an ABA framework that is assumption-spanning.

## 6 Discussion

In this work 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 and Toni, 2015], but is in fact also itself subsumed by normal logic programming 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 essentially the same as the outcome yielded by its (syntactically nearly identical) associated normal logic program under common LP semantics, and vice versa. The difference is that in ABA the outcome is defined in terms of assumptions (which correspond to negation-as-failure atoms in the associated logic program) whereas in LP the outcome is defined in terms of atoms in the logic program. However, since in LP the status of the atoms is determined solely by the status of the negation-as-failure atoms, flat ABA and normal LP are semantically equivalent. As a by-product of our work, we prove that the ideal semantics for ABA is not just inspired by, but *coincides* with, the ideal scenario semantics for LP [Alferes *et al.*, 1993].

ABA methods have already been successfully applied to LP, e.g. for explaining [Schulz and Toni, 2016] as well as visualizing [Schulz, 2015] logic programs under certain semantics. Our results pave the way for the application of LP methods to (flat) ABA frameworks, since our translation prevents an exponential blow-up while preserving the semantics. Efficient computation methods for LP semantics [Janhunen *et al.*, 2006; Gebser *et al.*, 2011] could for instance be used to determine the semantics of ABA frameworks, which is a promising direction for future work.

[2] Apart from that, the resulting ABA framework has an assumption `not_x` $\in \mathcal{A}$ (with $\overline{\texttt{not\_x}} = x$) for each atom $x$ occurring in the logic program, even in cases when `not_x` itself does not occur in any of the ABA inference rules in $\mathcal{R}$. This makes sure that the resulting ABA framework is assumption-spanning

## References


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

[Bondarenko *et al.*, 1997] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[Caminada and Gabbay, 2009] M.W.A. Caminada and D.M. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2-3):109–145, 2009. Special issue: new ideas in argumentation theory.

[Caminada and Schulz, 2017] M.W.A. Caminada and C. Schulz. On the equivalence between assumption-based argumentation and logic programming. *Journal of Artificial Intelligence Research*, 60:779–825, 2017.

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

[Dung *et al.*, 2007] P.M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[Dung *et al.*, 2009] P.M. Dung, R.A. Kowalski, and F. Toni. Assumption-based argumentation. In G. Simari and I. Rahwan, editors, *Argumentation in Artificial Intelligence*, pages 199–218. Springer US, 2009.

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

[Fan and Toni, 2014] X. Fan and F. Toni. A general framework for sound assumption-based argumentation dialogues. *Artificial Intelligence*, 216:20–54, 2014.

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

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

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

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

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

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

[Schulz and Toni, 2014] C. Schulz and F. Toni. Complete assumption labellings. In S. Parsons, N. Oren, C. Reed, and F. Cerutti, editors, *Proceedings of Computational Models of Argument (COMMA)*, pages 405–412. IOS Press, 2014.

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

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

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

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

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

[Toni, 2008] F. Toni. Assumption-based argumentation for epistemic and practical reasoning. In Pompeu Casanovas, Giovanni Sartor, Núria Casellas, and Rossella Rubino, editors, *Computable Models of the Law, Languages, Dialogues, Games, Ontologies*, pages 185–202. Springer Berlin Heidelberg, 2008.

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

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

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