Complete Extensions in Argumentation Coincide with Three-Valued Stable Models in Logic Programming

Martin Caminada [a] Yining Wu [a]

[a] *University of Luxembourg*

Abstract

In this paper, we prove the correspondence between complete extensions in abstract argumentation and 3-valued stable models in logic programming. This result is in line with earlier work of [8] that identified the correspondence between the grounded extension in abstract argumentation and the well-founded model in logic programming, as well as between the stable extensions in abstract argumentation and the stable models in logic programming. We believe the results of this paper are not only relevant by themselves, but can also potentially be used for future work on the correspondence between argumentation and logic programming semantics.

1 Introduction

Formal argumentation has become a popular approach for purposes varying from nonmonotonic reasoning [2, 3], multi-agent communication [1] and reasoning in the semantic web [19]. Although some research on formal argumentation can be traced back to the early 1990s (like for instance the work of Vreeswijk [23] and of Simari and Loui [20]) the topic really started to take off with Dung's theory of abstract argumentation [8]. Here, arguments are seen as abstract entities (although they can be instantiated using approaches like [3] and [16]) among which an attack relationship is defined. The thus formed *argumentation framework* can be represented as a directed graph in which the arguments serve as nodes and the attack relation as the arrows.

Given such a graph, an interesting question is which sets of nodes can reasonably be accepted. Several criteria of acceptance have been stated, including grounded, complete, preferred and stable semantics [8], as well as more recent approaches like semi-stable semantics [6] and ideal semantics [9].

The diversity of abstract argumentation is to some extent matched by the field of logic programming, where a wide variety of approaches has been formulated to describe the meaning of a logic program. Examples of these are the well-founded [21], regular [25] and stable model semantics [12, 13], of which the last one has currently gained the most popularity.

There exists an interesting overlap between abstract argumentation and logic programming, which is also reflected in the similarity between argumentation and logic programming semantics. For instance, in [8] it is observed that the grounded extension in abstract argumentation corresponds to the well-founded model in logic programming, and that the stable extensions in abstract argumentation correspond to the stable models in logic programming.

In the current paper, we examine another overlap between abstract argumentation semantics and logic programming semantics. We show that the complete extensions of abstract argumentation [8] coincide with the 3-valued stable models of logic programming [17]. This overlap is relevant because both complete extensions and 3-valued stable models have been used as the basis for describing other semantics for abstract argumentation and logic programming. Ideally, the correspondence between complete extensions and 3-valued stable models could therefore serve as a basis for identifying additional correspondences between abstract argumentation semantics and logic programming semantics, although these are not yet explicitly identified in the current paper.

The remaining part of this paper is structured as follows. Section 2 and Section 3 state some preliminaries on argument semantics, argument labellings and logic program. Section 4 demonstrates the equivalence between complete labellings and 3-valued stable models. Finally in Section 5 we address some future work and conclude the paper with a discussion.

2 Argument Semantics and Argument Labellings

In this section, we briefly restate some preliminaries regarding argument semantics and argument-labellings.

Definition 1. *An* argumentation framework *is a pair* (Ar, att) *where* Ar *is a finite set of arguments and* $att \subseteq Ar \times Ar$.

We say that argument A *attacks* argument B iff $(A, B) \in att$. An argumentation framework can be represented as a directed graph in which the arguments are represented as nodes and the attacks relation is represented as arrows.

Definition 2 (defense / conflict-free)**.** *Let* (Ar, att) *be an argumentation framework,* $A \in Ar$ *and* $\mathcal{A}rgs \subseteq Ar$. $\mathcal{A}rgs$ *is* conflict-free *iff* $\neg\exists A, B \in \mathcal{A}rgs : A$ *attacks* B. $\mathcal{A}rgs$ defends *argument* A *iff* $\forall B \in Ar : (B$ *attacks* $A \supset \exists C \in \mathcal{A}rgs : C$ *attacks* $B)$. *Let* $F(\mathcal{A}rgs) = \{A \mid A$ *is defended by* $\mathcal{A}rgs\}$.

Definition 3 (acceptability semantics)**.** *Let* (Ar, att) *be an argumentation framework. A conflict-free set* $\mathcal{A}rgs \subseteq Ar$ *is called a* complete *extension iff* $\mathcal{A}rgs = F(\mathcal{A}rgs)$.

The concept of complete semantics was originally stated in terms of sets of arguments. It is equally well possible, however, to express this concept in terms of *argument labellings*. This approach has been proposed by Pollock [15] and Jakobovits and Vermeir [14], and has recently been extended by Caminada [4], Vreeswijk [24] and Verheij [22]. The idea of a labelling is to associate with each argument exactly one label, which can either be `in`, `out` or `undec`. The label `in` indicates that the argument is explicitly accepted, the label `out` indicates that the argument is explicitly rejected, and the label `undec` indicates that the status of the argument is undecided, meaning that one abstains from an explicit judgment whether the argument is `in` or `out`.

Definition 4. *A* labelling *is a function* $\mathcal{L} : Ar \longrightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$.

We write $\texttt{in}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{in}\}$, $\texttt{out}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{out}\}$ and $\texttt{undec}(\mathcal{L})$ for $\{A \mid \mathcal{L}(A) = \texttt{undec}\}$. We say that an argument A is *legally* `in` iff $\mathcal{L}(A) = \texttt{in}$ and all the attackers of A are labelled `out`. We say that an argument A is *legally* `out` iff $\mathcal{L}(A) = \texttt{out}$ and there exists an attacker of A which is labelled `in`. We say that an argument A is *legally* `undec` iff $\mathcal{L}(A) = \texttt{undec}$ and there is no attacker of A that is labelled `in` and not all the attackers of A are labelled `out`.

Definition 5. *Let* $\mathcal{L}$ *be a labelling of argumentation framework* (Ar, att) *and* $A \in Ar$. *We say that:*

1. A *is* legally `in` *iff* $\mathcal{L}(A) = \texttt{in}$ *and* $\forall B \in Ar : (B\ att\ A \supset \mathcal{L}(B) = \texttt{out})$
2. A *is* legally `out` *iff* $\mathcal{L}(A) = \texttt{out}$ *and* $\exists B \in Ar : (B\ att\ A \wedge \mathcal{L}(B) = \texttt{in})$.
3. A *is* legally `undec` *iff* $\mathcal{L}(A) = \texttt{undec}$
   and $\neg\forall B \in Ar : (B\ att\ A \supset \mathcal{L}(B) = \texttt{out})$
   and $\neg\exists B \in Ar : (B\ att\ A \wedge \mathcal{L}(B) = \texttt{in})$.

We say that an argument A is *illegally* `in` iff $\mathcal{L}(A) = \texttt{in}$ but A is not legally `in`. We say that an argument A is *illegally* `out` iff $\mathcal{L}(A) = \texttt{out}$ but A is not legally `out`. We say that an argument A is *illegally* `undec` iff $\mathcal{L}(A) = \texttt{out}$ but A is not legally `undec`.

Definition 6. *An* admissible labelling $\mathcal{L}$ *is a labelling where each argument that is labelled* `in` *is legally* `in` *and each argument that is labelled* `out` *is legally* `out`.
A complete labelling *is an admissible labelling where each argument that is labelled* `undec` *is legally* `undec`.

We now define two functions that, given an argumentation framework, allow a set of arguments to be converted to a labelling and vice versa. The function $\texttt{Ext2Lab}_{(Ar,att)}$ takes a set of arguments (sometimes an extension) and converts it to a labelling. The function $\texttt{Lab2Ext}_{(Ar,att)}$ takes an labelling and converts it to a set of arguments (sometimes an extension). Since a labelling is a function, it is possible to represent the labelling as a set of pairs.

Definition 7. *Let* (Ar, att) *be an argumentation framework,* $\mathcal{A}rgs \subseteq Ar$ *such that* $\mathcal{A}rgs$ *is conflict-free, and* $\mathcal{L} : Ar \rightarrow \{\mathtt{in}, \mathtt{out}, \mathtt{undec}\}$ *a labelling. We define* $\mathtt{Ext2Lab}_{(Ar,att)}(\mathcal{A}rgs)$ *as* $\{(A, \mathtt{in}) \mid A \in \mathcal{A}rgs\} \cup \{(A, \mathtt{out}) \mid \exists A' \in \mathcal{A}rgs : A' att A\} \cup \{(A, \mathtt{undec}) \mid A \notin \mathcal{A}rgs \wedge \neg\exists A' \in \mathcal{A}rgs : A' att A\}$. *We define* $\mathtt{Lab2Ext}_{(Ar,att)}(\mathcal{L})$ *as* $\{A \mid (A, \mathtt{in}) \in \mathcal{L})\}$.

When the associated argumentation framework is clear, we sometimes simply write `Ext2Lab` and `Lab2Ext` instead of $\mathtt{Ext2Lab}_{(Ar,att)}$ and $\mathtt{Lab2Ext}_{(Ar,att)}$.

It can be proved that the various types of labellings correspond to the various kinds of argument semantics [4, 7].

Theorem 8. *[4] Let* (Ar, att) *be an argumentation framework. If* $\mathcal{L}$ *is a complete labelling then* $\mathtt{Lab2Ext}(\mathcal{L})$ *is a complete extension. If* $\mathcal{A}rgs$ *is a complete extension then* $\mathtt{Ext2Lab}(\mathcal{A}rgs)$ *is a complete labelling.*

Proof. Please refer to [5]. □

When the domain and the range of `Lab2Ext` are restricted to complete labellings and complete extensions, and the domain and the range of `Ext2Lab` are restricted to complete extensions and complete labellings, then the resulting functions (call them $\mathtt{Lab2Ext}^r$ and $\mathtt{Ext2Lab}^r$) are bijective and are each other's inverse.

Theorem 9. *[4] Let* $\mathtt{Lab2Ext}^r_{(Ar,att)} : \{\mathcal{L} \mid \mathcal{L}$ *is a complete labelling of* $(Ar, att)\} \rightarrow \{\mathcal{A}rgs \mid \mathcal{A}rgs$ *is a complete extension of* $(Ar, att)\}$ *be a function defined by* $\mathtt{Lab2Ext}^r_{(Ar,att)}(\mathcal{L}) = \mathtt{Lab2Ext}_{(Ar,att)}(\mathcal{L})$. *Let* $\mathtt{Ext2Lab}^r_{(Ar,att)} : \{\mathcal{A}rgs \mid \mathcal{A}rgs$ *is a complete extension of* $(Ar, att)\} \rightarrow \{\mathcal{L} \mid \mathcal{L}$ *is a complete labeling of* $(Ar, att)\}$ *be a function defined by* $\mathtt{Ext2Lab}^r_{(Ar,att)}(\mathcal{A}rgs) = \mathtt{Ext2Lab}_{(Ar,att)}(\mathcal{A}rgs)$. *The functions* $\mathtt{Lab2Ext}^r_{(Ar,att)}$ *and* $\mathtt{Ext2Lab}^r_{(Ar,att)}$ *are bijective and are each other's inverse.*

Proof. Please refer to [5]. □

From Theorem 9 it follows that complete labellings and complete extensions stand in a one-to-one relationship to each other.

3 3-Valued Stable Models in Logic Programming

We will first summarize basic concepts and terminologies of standard logic programming.

Definition 10. *A* (normal) logic program *is a finite set of universally quantified rules of the form,*

$$\forall x_1, \ldots, x_n (A \leftarrow A_1, \ldots, A_m, \textit{not } B_1, \ldots, \textit{not } B_k),$$

commonly written $A \leftarrow A_1, \ldots, A_m, \textit{not } B_1, \ldots, \textit{not } B_k$, *where* $n, m, k \geq 0$ *and* A, A_i, B_j *are atoms.* A *is called the* head *of the rule, denoted by* $H(r)$. $A_1, \ldots, A_m, \textit{not } B_1, \ldots, \textit{not } B_k$ *is the* body *of the rule, denoted by* $B(r)$. *A rule (program) is* definite *or (*positive*) if it does not contain* `not`.

Given a logic program P, the *Herbrand Base* $\mathcal{B}_P$ of P is the set of all ground atoms that are the instances of the atoms occurring in P. The set of all ground instances of P w.r.t. $\mathcal{B}_P$ is denoted by $ground(P)$. A *(Herbrand) interpretation* $I =< T; F >$ for a program P can be viewed as a mapping from $\mathcal{B}_P$ to the set of truth values $\{t, f, u\}$, denoted by:

$$I(A) = \begin{cases} t & \text{if } A \in T, \\ u & \text{if } A \in \overline{I} \\ f & \text{if } A \in F \end{cases}$$

where $\overline{I} = \mathcal{B}_P - (T \cup F)$. t, f, u denote true, false and undefined respectively, ordered as $f < u < t$.

Definition 11. *[18] Let* P *be a logic program and* 3-valued model M *be an interpretation for* P. *Then* M *is a* 3-valued model *for* P *if every rule* r *in* $ground(P)$ *is satisfied by* M.

Let P be a logic program and I be any 3-valued interpretation . The *GL-transformation* $\frac{P}{I}$ of P w.r.t. I is obtained by replacing in the body of every rule of P all negative literals which are true (respectively undefined, false). by t (respectively u, f). The resulting program $\frac{P}{I}$ is definite, so it has a least model J. We define $\Gamma^*(I) = J$.

Definition 12. *[17] A 3-valued interpretation* M *of a logic program* P *is a* 3-valued stable model *of* P *if* $\Gamma^*(M) = M$.

4 Complete Labellings Coincide with Three-Valued Stable Models

We use the approach in [11] to transform argumentation frameworks into logic programs.
Each argumentation framework can be transformed into a logic program by generating a rule for each argument in the argumentation framework such that the argument itself is in the head of the rule and the negations of all its attackers be in the body of the rule.

Definition 13. *Let $AF = (Ar, att)$ be an argumentation framework. We define the associated logic program P_{AF} as follows,*
$P_{AF} = \{A \leftarrow$ *not* $B_1, \ldots,$ *not* $B_n \mid A, B_1, \ldots, B_n \in Ar\ (n \geq 0)$ *and* $\{B_i \mid (B_i, A) \in att\} = \{B_1, \ldots, B_n\}\}$.

We now define two functions that, given an argumentation framework AF, allow a labelling to be converted to a 3-valued interpretation of P_{AF} and vice versa.

Definition 14. *Let $\mathcal{L}abellings$ be the set of all labellings of AF and $\mathcal{M}odels$ be all the 3-valued interpretations of P_{AF}. Let $\mathcal{L} \in \mathcal{L}abellings$. We introduce a function* $\texttt{Lab2Mod} : \mathcal{L}abellings \rightarrow \mathcal{M}odels$ *such that* $\texttt{Lab2Mod}(\mathcal{L}) =< \texttt{in}(\mathcal{L}); \texttt{out}(\mathcal{L}) >$ *and* $\texttt{Lab2Mod}(\mathcal{L}) = \texttt{undec}(\mathcal{L})$.

Definition 15. *Let $\mathcal{L}abellings$ be the set of all labellings of AF and $\mathcal{M}odels$ be all the 3-valued interpretations of P_{AF}. Let $I \in \mathcal{M}odels$ and $I =< T, F >$. We define a function* $\texttt{Mod2Lab} : \mathcal{M}odels \rightarrow \mathcal{L}abellings$ *such that* $\texttt{in}(\texttt{Mod2Lab}(I)) = T$ *and* $\texttt{out}(\texttt{Mod2Lab}(I)) = F$ *and* $\texttt{undec}(\texttt{Mod2Lab}(I)) = \overline{I}$.

When $\mathcal{L}$ is a complete labelling of an argumentation framework, then $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of the associated logic program, as is stated by the following theorem.

Theorem 16. *Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{L}$ be a complete labelling of AF. Then* $\texttt{Lab2Mod}(\mathcal{L})$ *is a 3-valued stable model of P_{AF}.*

Proof. In order to prove $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of P_{AF} we have to verify that $\texttt{Lab2Mod}(\mathcal{L})$ is a fixed point of Γ^*. We first examine $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$ (the reduct of P_{AF} under $\texttt{Lab2Mod}(\mathcal{L})$).

Let $A \leftarrow$ not $B_1, \ldots,$ not B_n be a rule of P_{AF} (corresponding with an argument A that has attackers $B_1, \ldots, B_n$). We distinguish three cases.

1. All $B_1, \ldots, B_n$ are labelled `out` by $\mathcal{L}$. Then A is labelled `in` by $\mathcal{L}$. The reduct of the rule is the $A \leftarrow t$, so in the smallest model of $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$, A will be *true* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

2. There is a B_i $(1 \leq i \leq n)$ that is labelled `in`. Then A is labelled `out` by $\mathcal{L}$. The reduct of the rule is that $A \leftarrow v_1, \ldots, f, \ldots, v_n$ $(v_i \in \{t, f, u\})$ which is equivalent to $A \leftarrow f$. Since there is no other rule with A in the head, this means that in the smallest model of $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$, A will be *false* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

3. Not each $B_1, \ldots, B_n$ is labelled `out` by $\mathcal{L}$ and there is no B_i $(i \leq i \leq n)$ that is labelled `in` by $\mathcal{L}$. Then A is labelled `undec` by $\mathcal{L}$. It also implies that there is at least one B_i labelled `undec`. Thus the reduct of the rule is $A \leftarrow, v_1, \ldots, u, \ldots, v_n$ $(v_i \in \{t, u\})$. Since this is the only rule that has A in the head, A will be *undefined* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.

Since for any arbitrary argument A, it holds that $\texttt{Lab2Mod}(\mathcal{L})(A) = \Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))(A)$, it follows that $\texttt{Lab2Mod}(\mathcal{L}) = \Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$. Hence $\texttt{Lab2Mod}(\mathcal{L})$ is a fixed point of Γ^*, so $\texttt{Lab2Mod}(\mathcal{L})$ is a 3-valued stable model of P_{AF}. □

When an argumentation framework is transformed into a logic program, and $\mathcal{M}$ is a 3-valued stable model of this logic program, then $\texttt{Mod2Lab}(\mathcal{M})$ is a complete labelling of the original argumentation framework.

Theorem 17. *Let $AF = (Ar, att)$ be an argumentation framework and $\mathcal{M}$ be a 3-valued stable model of P_{AF}. Then* $\texttt{Mod2Lab}(\mathcal{M})$ *is a complete labelling of AF.*

Proof. $\mathcal{M}$ is a 3-valued stable model of P_{AF}. Then $\mathcal{M}$ is a fixed point of Γ^*, that is $\Gamma^*(\mathcal{M}) = \mathcal{M}$. We now prove that $\texttt{Mod2Lab}(\mathcal{M})$ is a complete labelling of AF.
Let A be an arbitrary argument in Ar. We distinguish three cases.

1. $\mathcal{M}(A) = t$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow \text{not } B_1, \ldots, \text{not } B_n$ is equivalent to $A \leftarrow t$. This means that each B_i $(1 \leq i \leq n)$ is labelled `out` in $\texttt{Mod2Lab}(\mathcal{M})$. So A is legally `in` in $\texttt{Mod2Lab}(\mathcal{M})$.

2. $\mathcal{M}(A) = f$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow \text{not } B_1, \ldots, \text{not } B_n$ is equivalent to $A \leftarrow f$. This implies that there exists a B_i $(1 \leq i \leq n)$ that is labelled `in` in $\texttt{Mod2Lab}(\mathcal{M})$. So A is legally `out` in $\texttt{Mod2Lab}(\mathcal{M})$.

3. $\mathcal{M}(A) = u$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the rule $A \leftarrow \text{not } B_1, \ldots, \text{not } B_n$ is equivalent to $A \leftarrow u$. This implies that there exists a B_i $(1 \leq i \leq n)$ that is *undefined* in M and that each of the B_j $(1 \leq j \leq n, j \neq i)$ is either *false* or *undefined* in M. Hence, A has no attackers that is labelled `in` by $\texttt{Mod2Lab}(\mathcal{M})$ and not all its attackers are labelled `out` by $\texttt{Mod2Lab}(\mathcal{M})$. Thus A is legally `undec` in $\texttt{Mod2Lab}(\mathcal{M})$.

Since this holds for any arbitrary argument A, it follows that each argument that is `in` is legally `in`, each argument that is `out` is legally `out`, and each argument that is `undec` is legally `undec`. Hence, $\texttt{Mod2Lab}(\mathcal{M})$ is a complete labelling of AF. □

When `Lab2Mod` and `Mod2Lab` are restricted to work only on complete labellings and 3-valued stable models, they turn out to be bijective and each other's inverse.

Theorem 18. *Let $AF = (Ar, att)$ be an argumentation framework.*
Let $\texttt{Lab2Mod}^r : \{\mathcal{L} \mid \mathcal{L} \text{ is a complete labelling of } AF\} \rightarrow \{\mathcal{M} \mid \mathcal{M} \text{ is a 3-valued stable model of } P_{AF}\}$ be a function defined by $\texttt{Lab2Mod}^r(\mathcal{L}) = \texttt{Lab2Mod}(\mathcal{L})$.
Let $\texttt{Mod2Lab}^r : \{\mathcal{M} \mid \mathcal{M} \text{ is a 3-valued stable of model of } P_{AF}\} \rightarrow \{\mathcal{L} \mid \mathcal{L} \text{ is a complete labelling of } AF\}$ be a function defined by $\texttt{Mod2Lab}^r(\mathcal{M}) = \texttt{Mod2Lab}(\mathcal{M})$.
$\texttt{Lab2Mod}^r$ and $\texttt{Mod2Lab}^r$ are bijective and are each other's inverse.

Proof. As every function that has an inverse is bijective, we only need to prove that $\texttt{Lab2Mod}^r$ and $\texttt{Mod2Lab}^r$ are each other's inverse, meaning that $(\texttt{Lab2Mod}^r)^{-1} = \texttt{Mod2Lab}^r$ and $(\texttt{Mod2Lab}^r)^{-1} = \texttt{Lab2Mod}^r$. Let $AF = (Ar, att)$ be an argumentation framework, we prove the following two things:

1. For every complete labelling $\mathcal{L}$ of AF it holds that $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L})) = \mathcal{L}$.
   Let $\mathcal{L}$ be a complete labelling of AF and $A \in Ar$.
   If $\mathcal{L}(A) = \texttt{in}$ then A is t in $\texttt{Lab2Mod}^r(\mathcal{L})$, so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{in}$.
   If $\mathcal{L}(A) = \texttt{out}$ then A is f in $\texttt{Lab2Mod}^r(\mathcal{L})$, so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{out}$.
   If $\mathcal{L}(A) = \texttt{undec}$ then A is u in $\texttt{Lab2Mod}^r(\mathcal{L})$, so $\texttt{Mod2Lab}^r(\texttt{Lab2Mod}^r(\mathcal{L}))(A) = \texttt{undec}$.

2. For every 3-valued stable model $\mathcal{M}$ of P_{AF} it holds that $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M})) = \mathcal{M}$.
   Let $\mathcal{M}$ be a 3-valued stable model $\mathcal{M}$ of P_{AF}.
   If $\mathcal{M}(A) = t$ then $\texttt{Mod2Lab}^r(A) = \texttt{in}$, so A is t in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.
   If $\mathcal{M}(A) = f$ then $\texttt{Mod2Lab}^r(A) = \texttt{out}$, so A is f in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.
   If $\mathcal{M}(A) = u$ then $\texttt{Mod2Lab}^r(A) = \texttt{undec}$, so A is u in $\texttt{Lab2Mod}^r(\texttt{Mod2Lab}^r(\mathcal{M}))$.

□

From Theorem 18, it follows that complete labellings and 3-valued stable models are one-to-one related. Since Theorem 9 states that complete extensions and complete labellings are one-to-one related, it follows that complete extensions, complete labellings and 3-valued stable models are different ways of describing essentially the same concept.

5 Discussion

In this paper we have shown that complete labellings have a one-to-one correspondence with 3-valued stable models (Theorem 18). As it was shown earlier that complete extensions have a one-to-one correspondence with complete labellings (Theorem 9), it directly follows that complete extensions have a one-to-one relationship with 3-valued stable models. The existence of the associated translation functions (`Lab2Ext`,

Ext2Lab, Lab2Mod and Mod2Lab) implies that complete extensions and 3-valued stable models are different ways of expressing essentially the same concept.

Both complete extensions and 3-valued stable models have been used as a basis for describing various other semantics in abstract argumentation and logic programming. As a result of this, we expect to obtain a number of established results almost immediately using the results of this paper, like the correspondence between the grounded extension in abstract argumentation (which is the smallest complete extension) and the well-founded model in logic programming (which is the smallest 3-valued stable model), as well as the correspondence between the stable extensions in abstract argumentation and the stable models in logic programming.

However, the results in this paper also allow for new correspondences to be identified. A topic for further study would for instance be the possible correspondence between the semi-stable extensions in abstract argumentation [6] and the L-stable models [10] in logic programming, which once established would allow for algorithms and complexity results that were found for argumentation under semi-stable semantics to be applied to logic programming under the L-stable model approach.

References

[1] L. Amgoud, N. Maudet, and S. Parsons. Modelling dialogues using argumentation. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-00)*, pages 31–38, Boston, MA, 2000.

[2] ASPIC-consortium. Deliverable D2.5: Draft formal semantics for ASPIC system, June 2005.

[3] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5-6):286–310, 2007.

[4] M.W.A. Caminada. On the issue of reinstatement in argumentation. In M. Fischer, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Logics in Artificial Intelligence; 10th European Conference, JELIA 2006*, pages 111–123. Springer, 2006. LNAI 4160.

[5] M.W.A. Caminada. On the issue of reinstatement in argumentation. Technical Report UU-CS-2006-023, Institute of Information and Computing Sciences, Utrecht University, 2006.

[6] M.W.A. Caminada. Semi-stable semantics. In P.E. Dunne and TJ.M. Bench-Capon, editors, *Computational Models of Argument; Proceedings of COMMA 2006*, pages 121–130. IOS Press, 2006.

[7] M.W.A. Caminada. An algorithm for computing semi-stable semantics. In *Proceedings of the 9th European Conference on Symbolic and Quantitalive Approaches to Reasoning with Uncertainty (ECSQARU 2007)*, number 4724 in Springer Lecture Notes in AI, pages 222–234, Berlin, 2007. Springer Verlag.

[8] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

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

[10] Thomas Eiter, Nicola Leone, and Domenico Saccà. On the partial semantics for disjunctive deductive databases. *Ann. Math. Artif. Intell.*, 19(1-2):59–96, 1997.

[11] Dov M Gabbay and A Garcez. Logical modes of attack in argumentation networks. To appear in Studia Logica, 2009.

[12] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference/Symposium on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[13] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–385, 1991.

[14] H. Jakobovits and D. Vermeir. Robust semantics for argumentation frameworks. *Journal of logic and computation*, 9(2):215–261, 1999.

[15] J. L. Pollock. *Cognitive Carpentry. A Blueprint for How to Build a Person*. MIT Press, Cambridge, MA, 1995.

[16] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.

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

[18] Teodor C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Comput.*, 9(3/4):401–424, 1991.

[19] I. Rahwan, F. Zablith, and C. Reed. Laying the foundations for a world wide argument web. *Artificial Intelligence*, 171(10-15):897–921, 2007.

[20] G.R. Simari and R.P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53:125–157, 1992.

[21] Allen van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *J. ACM*, 38(3):620–650, 1991.

[22] Bart Verheij. A labeling approach to the computation of credulous acceptance in argumentation. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pages 623–628, 2007.

[23] G. A. W. Vreeswijk. Studies in defeasible argumentation. *PhD thesis at Free University of Amsterdam*, 1993.

[24] G.A.W. Vreeswijk. An algorithm to compute minimally grounded and admissible defence sets in argument systems. In P.E. Dunne and TJ.M. Bench-Capon, editors, *Computational Models of Argument; Proceedings of COMMA 2006*, pages 109–120. IOS, 2006.

[25] Jia-Huai You and Li Yan Yuan. Three-valued formalization of logic programming: is it needed? In *PODS '90: Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 172–182, New York, NY, USA, 1990. ACM.