Yining Wu
Martin Caminada
Dov M. Gabbay

Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming

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 [6] 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.

Keywords: abstract argumentation, complete extensions, logic programming, 3-valued stable models.

1. Introduction

Dung's theory of abstract argumentation has been shown to be suitable to express a whole range of logical formalisms for nonmonotonic reasoning, including logic programming with weak negation [6]. The main concept in Dung's theory is that of an argumentation framework, which is essentially a directed graph in which the nodes represent arguments and the arrows represent the defeat an attack relation.

Given such a graph, different sets of nodes can be accepted according to various argument based semantics such as grounded, preferred and stable semantics [6], semi-stable semantics [3] or ideal semantics [7]. Many of these semantics can be seen as restricted cases of complete semantics; an overview is provided at the left hand side of Figure 1. The facts that every stable extension is also a semi-stable extension and that every semi-stable extension is also a preferred extension has been proved in [3]. The facts that every preferred extension is also a complete extension and that the grounded extension is also a complete extension have been stated in [6].

At the right hand side of Figure 1 we find a number of logic programming semantics. The 3-valued stable semantics in logic programming was introduced in [14]. It has been used as the basis for describing other semantics in

Special Issue: New Ideas in Argumentation Theory
Edited by **Dov M. Gabbay and Leendert van der Torre**

Studia Logica (2009) 93: 383–403
DOI: 10.1007/s11225-009-9210-5

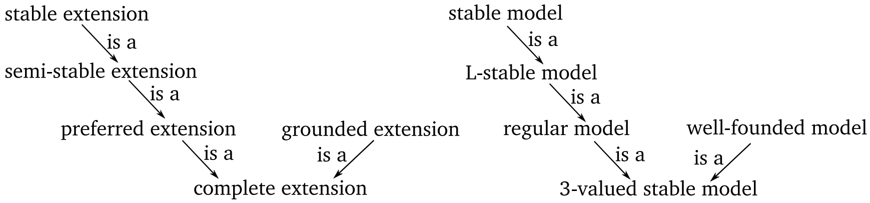

Figure 1. An overview of the different semantics

logic programming such as well-founded model, regular models, stable models and L-stable models. It has been proved that the well-founded model is also a 3-valued stable model [14]. In [8] it is shown that every L-stable model is also a regular model, that every regular model is also a 3-valued stable model and that every (2-valued) stable model is also an L-stable model.

Many of the existing semantics for logic programing can be understood in a uniform way using argumentation. The overlap between logic programming and abstract argumentation has been studied by Dung in [6] which shows 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 will examine an additional similarity between argumentation theory and logic programming, namely the correspondence between 3-valued stable models in logic programming and complete extensions in abstract argumentation. This correspondence can easily lead to other overlaps between abstract argumentation semantics and logic programming semantics since they are both used as basis of defining other semantics in abstract argumentation and logic programming.

The remaining part of this paper is organized in the following way. Section 2 and Section 3 state some preliminaries on argument semantics, argument labellings and logic programing. Section 4 demonstrates the equivalence between complete labellings (which coincide with complete extensions [1]) and 3-valued stable models. Finally in Section 5 we discuss the main results of the paper and identify some possibilities for further research.

2. Argument Semantics and Argument Labellings

In this section, we briefly restate some preliminaries regarding argument semantics and argument-labellings. For simplicity, we only consider finite argumentation frameworks.

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 attack 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*. The approach of (argument) labellings has been used by Pollock [11] and by Jakobovits and Vermeir [10], and has recently been extended by Caminada [1], Vreeswijk [16] and Verheij [15]. 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 (possibly 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 (possibly 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 $\mathcal{L} : Ar \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ a labelling. We define $\texttt{Ext2Lab}_{(Ar,att)}(\mathcal{A}rgs)$ as $\{(A, \texttt{in}) \mid A \in \mathcal{A}rgs\} \cup \{(A, \texttt{out}) \mid \exists A' \in \mathcal{A}rgs : A' att A\} \cup \{(A, \texttt{undec}) \mid A \notin \mathcal{A}rgs \wedge \neg\exists A' \in \mathcal{A}rgs : A' att A\}$. We define $\texttt{Lab2Ext}_{(Ar,att)}(\mathcal{L})$ as $\{A \mid (A, \texttt{in}) \in \mathcal{L})\}$.

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

It can be proved that the various types of labellings correspond to the various kinds of argument semantics [1, 4].

Theorem 8. [1] *Let (Ar, att) be an argumentation framework. If $\mathcal{L}$ is a complete labelling then* $\texttt{Lab2Ext}(\mathcal{L})$ *is a complete extension. If $\mathcal{A}rgs$ is a complete extension then* $\texttt{Ext2Lab}(\mathcal{A}rgs)$ *is a complete labelling.*

Proof. Please refer to [2]. ■

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 $\texttt{Lab2Ext}^r$ and $\texttt{Ext2Lab}^r$) are bijective and are each other's inverse.

THEOREM 9. [1] *Let* $\texttt{Lab2Ext}^r_{(Ar,att)} : \{\mathcal{L} \mid \mathcal{L}$ *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* $\texttt{Lab2Ext}^r_{(Ar,att)}(\mathcal{L}) = \texttt{Lab2Ext}_{(Ar,att)}(\mathcal{L})$.
Let $\texttt{Ext2Lab}^r_{(Ar,att)} : \{\mathcal{A}rgs \mid \mathcal{A}rgs$ *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* $\texttt{Ext2Lab}^r_{(Ar,att)}(\mathcal{A}rgs) = \texttt{Ext2Lab}_{(Ar,att)}(\mathcal{A}rgs)$.
The functions $\texttt{Lab2Ext}^r_{(Ar,att)}$ *and* $\texttt{Ext2Lab}^r_{(Ar,att)}$ *are bijective and are each other's inverse.*

PROOF. Please refer to [2]. ■

From Theorem 9 it follows that complete labellings and complete extensions stand in a one-to-one relationship to each other. In essence, complete labellings and complete extensions are different ways to describe the same concept.

3. 3-Valued Stable Models in Logic Programming

We will first summarize some basic concepts and terminologies in the field of logic programming.

A logic program is a finite set of rules of the form $A \leftarrow A_1, \ldots, A_n, \text{not } B_1, \ldots, \text{not } B_m$, where $n, m \geq 0$ and A, A_i, B_j $(1 \leq i \leq n, 1 \leq j \leq m)$ are atoms. A is called the *head* of the rule. $A_1, \ldots, A_n, \text{not } B_1, \ldots, \text{not } B_m$ is the *body* of the rule.

Given a logic program P, $\mathcal{A}_P$ is the set of all atoms occurring in P. An *interpretation* $I =< T; F >$ for a program P can be viewed as a mapping from $\mathcal{A}_P$ 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 $\overline{I} = \mathcal{A}_P - (T \cup F)$. t, f, u denote *true*, *false* and *undefined* respectively, ordered as $f < u < t$.

DEFINITION 10. Let P be a logic program and M be a 3-valued interpretation for P. Then M is a *3-valued model* for P if every rule r in 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 positive, so it has a least model J. We define $\Gamma^*(I) = J$.

Definition 11. [14] 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 3-Valued Stable Models

In this section we first transform argumentation frameworks into logic programs and prove that the complete labellings of an argumentation framework coincide with the 3-valued stable models of the associated logic program (Section 4.1). Then, in Section 4.2 we transform logic programs into argumentation frameworks and prove that the 3-valued stable models of a logic program coincide with the complete labellings of the associated argumentation framework.

4.1. Transforming Argumentation Frameworks into Logic Programs

We use the approach of [9] to transform argumentation frameworks into logic programs.
An 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 are in the body of the rule.

Definition 12. Let $AF = (Ar, att)$ be an argumentation framework. We define the associated logic program P_{AF} as follows,
$P_{AF} = \{A \leftarrow \text{not } B_1, \ldots, \text{not } B_n \mid A, B_1, \ldots, B_n \in Ar\ (n \geq 0) \text{ and } \{B_i \mid (B_i, A) \in att\} = \{B_1, \ldots, B_n\}\}$.

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 13. 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 $\overline{\texttt{Lab2Mod}(\mathcal{L})} = \texttt{undec}(\mathcal{L})$.

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 $I \in \mathcal{M}odels$ 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 15. *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 $A \leftarrow \text{not } B_1, \dots, \text{not } B_n$ be a rule of P_{AF} (corresponding with an argument A that has attackers $B_1, \dots, B_n$). We distinguish three cases.

1. A is labelled $\texttt{in}$ by $\mathcal{L}$.
   Then from the fact that $\mathcal{L}$ is a complete labelling it follows that $B_1, \dots, B_n$ are labelled $\texttt{out}$ by $\mathcal{L}$. The reduct of the rule is therefore $A \leftarrow t$, so in the smallest model of $\frac{P_{AF}}{\texttt{Lab2Mod}(\mathcal{L})}$, A will be *true* in $\Gamma^*(\texttt{Lab2Mod}(\mathcal{L}))$.
2. A is labelled $\texttt{out}$ by $\mathcal{L}$.
   Then, from the fact that $\mathcal{L}$ is a complete labelling it follows that there is a B_i $(1 \leq i \leq n)$ that is labelled $\texttt{in}$. The reduct of the rule is therefore $A \leftarrow v_1, \dots, f, \dots, v_n$ $(v_i \in \{t, f, u\})$ 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. A is labelled $\texttt{undec}$ by $\mathcal{L}$.
   Then from the fact that $\mathcal{L}$ is a complete labelling it follows that not each $B_1, \dots, B_n$ is labelled $\texttt{out}$ by $\mathcal{L}$ and there is no B_i $(i \leq i \leq n)$ that is labelled $\texttt{in}$ by $\mathcal{L}$. It also follows that there is at least one B_i labelled $\texttt{undec}$. The reduct of the rule is therefore $A \leftarrow, v_1, \dots, u, \dots, v_n$ $(v_i \in \{t, u\})$. 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}. ■

The next thing to be proved is that 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 16. *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. Let $\mathcal{M}$ be 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 17. *Let $AF = (Ar, att)$ be an argumentation framework.*
Let $\texttt{Lab2Mod}^r : \{\mathcal{L} \mid \mathcal{L}$ *is a complete labelling of* $AF\} \rightarrow \{\mathcal{M} \mid \mathcal{M}$ *is a 3-valued stable model of* $P_{AF}\}$ *be a function defined by* $\texttt{Lab2Mod}^r(\mathcal{L}) = \texttt{Lab2Mod}(\mathcal{L})$.
Let $\texttt{Mod2Lab}^r : \{\mathcal{M} \mid \mathcal{M}$ *is a 3-valued stable of model of* $P_{AF}\} \rightarrow \{\mathcal{L} \mid \mathcal{L}$ *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 17, it follows that complete labellings of an argumentation framework and 3-valued stable models of the associated logic program are one-to-one related.

4.2. Transforming Logic Programs to Argumentation Frameworks

We show that the complete labellings still coincide with 3-valued stable models if we transform logic programs into argumentation frameworks.

We follow the approach of structured arguments (like is taken in [13, 5, 12]) to do the transformation. The reasons for doing so are discussed in the epilogue.

Definition 18. Let P be a logic program.

- An argument A based on P is a finite tree of rules from P such that (1) each node (of the form $c \leftarrow a_1, \dots, a_n,$ not $b_1, \dots,$ not b_m with $n \geq 0$ and $m \geq 0$) has exactly n children, each having a different head $a_i \in \{a_1, \dots, a_n\}$ and (2) no rule occurs more than once in any root-originated branch of the tree. The conclusion of A ($\texttt{Conc}(A)$) is the head of its root.
- An argument a_1 attacks an argument a_2 iff a_1 has conclusion c and a_2 has a rule containing $\texttt{not}\ c$.

We say that argument A is a *subargument* of argument B iff A is a subtree of B.

In Definition 18 the reason for including the condition that each rule occurs no more than once in each root-originated branch is to make sure that a finite program will yield a finite number of arguments.[1]

Definition 19. Let P be a logic program. We define the associated argumentation framework $AF_P = (Ar, att)$ where Ar is the set of arguments that can be constructed using P, and att is the attack relation under P.

We define a strict order on the labels $\{\texttt{out}, \texttt{undec}, \texttt{in}\}$ such that $\texttt{out} < \texttt{undec} < \texttt{in}$.

In order to convert labellings to 3-valued stable interpretations, we define a function that assigns a label to each atom. The idea is that the label of an atom should be the label of the "best" argument that yields the atom as a conclusion (or be out if there is no argument at all that yields this atom as a conclusion).

Definition 20. Let P be a logic program and A_P be the set of all ground atoms that occur in P. Let $AF_P = (Ar, att)$ be the associated argumentation framework and $\mathcal{L}$ be a labelling of AF_P. We define a function $W(\mathcal{L}) : A_P \rightarrow \{\texttt{in}, \texttt{undec}, \texttt{out}\}$ such that $W(\mathcal{L})(c) = \max(\{\mathcal{L}(A) \mid A \in Ar \wedge \texttt{Conc}(A) = c\} \cup \{\texttt{out}\})$.

We now define two functions that, given a logic program P, allow a 3-valued interpretation to be converted to a labelling of AF_P and vice versa.

[1] Without this condition the logic program $\{a \leftarrow;\ a \leftarrow a\}$ would yield an infinite number of arguments.

Definition 21. Let $\mathcal{M}odels$ be all the 3-valued interpretations of P and $c \in A_P$. Let $\mathcal{L}abellings$ be the set of all labellings of AF_P and $\mathcal{L} \in \mathcal{L}abellings$. We introduce a function $\mathcal{L}ab2\mathcal{M}od : \mathcal{L}abellings \to \mathcal{M}odels$ such that $\mathcal{L}ab2\mathcal{M}od(\mathcal{L}) = \mathcal{M} =< T; F >$ where

- $T = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{in}\}$;
- $F = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{out}\}$;
- $\overline{\mathcal{M}} = \{c \mid c \in A_P \text{ and } W(\mathcal{L})(c) = \texttt{undec}\}$.

Definition 22. Let $\mathcal{M}odels$ be all the 3-valued interpretations of P and $\mathcal{L}abellings$ be the set of all labellings of AF_P. Let $M \in \mathcal{M}odels$ and $M =< T, F >$ and A be an argument in Ar. We define a function $\mathcal{M}od2\mathcal{L}ab : \mathcal{M}odels \to \mathcal{L}abellings$ such that

1. $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{in}$ if for each attacker B of A, $\texttt{Conc}(B) \in F$.
2. $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{out}$ if there is an attacker B of A such that $\texttt{Conc}(B) \in T$.
3. $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})(A) = \texttt{undec}$ if not each attacker of A has a conclusion that is in F and there is no attacker B of A such that $\texttt{Conc}(B) \in T$.

When a logic program is transformed into an argumentation framework, and $\mathcal{L}$ is a complete labelling of this argumentation framework, then $\mathcal{L}ab2\mathcal{M}od(\mathcal{L})$ is a 3-valued stable model of the logic program.

Theorem 23. *Let P be a logic program and $\mathcal{L}$ be a complete labelling of AF_P. Then $\mathcal{L}ab2\mathcal{M}od(\mathcal{L})$ is a 3-valued stable model of P.*

Proof. Let $\mathcal{M} = \mathcal{L}ab2\mathcal{M}od(\mathcal{L})$. In order to prove $\mathcal{M}$ is a 3-valued stable model of P we have to verify that $\mathcal{M}$ is a fixed point of Γ^*. We first examine $\frac{P}{\mathcal{M}}$ (the reduct of P under $\mathcal{M}$).

Let $A \in Ar$ and $c \leftarrow a_1, \ldots, a_n, \text{not } b_1, \ldots, \text{not } b_m$ $(n, m \geq 0)$ be the root of A. We distinguish three cases.

1. $c \in T$.
   This means that $W(\mathcal{L})(c) = \texttt{in}$. It follows that there exists an argument A such that A is labelled $\texttt{in}$ and $\texttt{Conc}(A) = c$. Then all attackers of A are labelled $\texttt{out}$. Let $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of A. It follows that $W(\mathcal{L})(b'_j) = \texttt{out}$ $(0 \leq j \leq l)$. Then b'_j is *false* in $\mathcal{M}$. We prove that all the conclusions of subarguments of A are *true* in $\Gamma^*(\mathcal{M})$ by induction.

- Basis. Let $c''' \leftarrow \text{not } b'''_1, \ldots, \text{not } b'''_{m'''}$ $(m''' \geq 0)$ be an arbitrary leaf in A such that the distance between the leaf and the root of A is the furthest. The distance between two nodes is the vertical distance between the two nodes in the direction from root down to leaves. Since $b'''_{j'''}$ $(0 \leq j''' \leq m''')$ is *false* in $\mathcal{M}$, then the reduct of the leaf is $c''' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *true* in $\Gamma^*(\mathcal{M})$.
- Step. Let $a''_1, \ldots, a''_{n''}$ $(n'' \geq 0)$ be heads of nodes such that the distance between them and the furthest leaf is n'. Assume that $a''_1, \ldots, a''_{n''}$ are *true* in $\Gamma^*(\mathcal{M})$.
  Let $c'' \leftarrow a''_{i''}, \ldots, a''_{j''}, \text{not } b''_1, \ldots, \text{not } b''_{m''}$ $(0 \leq i'', j'' \leq n'', m'' \geq 0)$ be a node that is $n' + 1$ distance from the furthest leaf in the tree of A. Since $b''_1, \ldots,$ $b''_{m''}$ are *false* in $\mathcal{M}$ and $a''_{i''}, \ldots, a''_{j''}$ are *true* in $\mathcal{M}$, the reduct of the node is $c'' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c'' will be *true* in $\Gamma^*(\mathcal{M})$.

So all the conclusions of subarguments of A are *true* in $\Gamma^*(\mathcal{M})$. Therefore in the least model of $\frac{P}{\mathcal{M}}$, c will be *true* in $\Gamma^*(\mathcal{M})$.

2. $c \in F$.
   This means that $W(\mathcal{L})(c) = \texttt{out}$. It follows that for all arguments A such that $\texttt{Conc}(A) = c$, A is labelled $\texttt{out}$. Then there exists an attacker of A that is labelled $\texttt{in}$. It follows that there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in A such that $W(\mathcal{L})(b'_j) = \texttt{in}$ $(1 \leq j \leq l)$. Then b'_j is *true* in $\mathcal{M}$. Then the reduct of the root of A is the $c \leftarrow f$. So in the least model of $\frac{P}{\mathcal{M}}$, c will be *false* in $\Gamma^*(\mathcal{M})$.
3. $c \in \overline{\mathcal{M}}$.
   This means that $W(\mathcal{L})(c) = \texttt{undec}$. It follows that there exists an argument A such that $\texttt{Conc}(A) = c$ and $\mathcal{L}(A) = \texttt{undec}$ and there is no argument A such that $\texttt{Conc}(A) = c$ and $\mathcal{L}(A) = \texttt{in}$. Then not each attacker of A is labelled $\texttt{out}$ and there is no attacker that is labelled $\texttt{in}$. It follows that there exists a rule $c'''' \leftarrow a''''_1, \ldots, a''''_{k'}, \text{not } b''''_1, \ldots, \text{not } b''''_{l'}$ $(k', l' \geq 0)$ in A such that $W(\mathcal{L})(b''''_j) = \texttt{undec}$ $(1 \leq j \leq l')$. Then b''''_j is *undefined* in $\mathcal{M}$. Let $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ be an arbitrary rule in A. Since there is no attacker that is labelled $\texttt{in}$, then for each b'_i $(0 \leq i \leq l)$, b'_i is not *true* in $\mathcal{M}$. We prove that the conclusions of subarguments of A are *undefined* in $\Gamma^*(\mathcal{M})$ if they have attackers that are labelled $\texttt{undec}$ by induction.

   - Basis. Let $c''' \leftarrow \text{not } b'''_1, \ldots, \text{not } b'''_{m'''}$ $(m''' \geq 0)$ be an arbitrary leaf in A such that the distance between the leaf and the root of A is the furthest. Since $b'''_{j'''}$ $(0 \leq j''' \leq m''')$ is either *false* or *undefined* in $\mathcal{M}$, then the reduct of the leaf is $c''' \leftarrow v_1, \ldots, v_{m'''}$ ($v_i \in \{t, u\}, 0 \leq i \leq m'''$). If the leaf has no attacker that is labelled `undec` the reduct of the leaf is $c''' \leftarrow t$. So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *true* in $\Gamma^*(\mathcal{M})$. If the leaf has an attacker that is labelled `undec` the reduct of the leaf is $c''' \leftarrow v_1, \ldots, u, \ldots, v_{m'''}$ ($v_i \in \{t, u\}, 0 \leq i \leq m'''$). So in the least model of $\frac{P}{\mathcal{M}}$, c''' will be *undefined* in $\Gamma^*(\mathcal{M})$.

- Step. Let $A''_1, \ldots, A''_{n''}$ ($n'' \geq 0$) be subarguments of A and their roots are n' distance from the furthest leaf. Let $a''_1, \ldots, a''_{n''}$ be conclusions of $A''_1, \ldots, A''_{n''}$ respectively. Assume that a''_i ($0 \leq i \leq n''$) is *true* in $\Gamma^*(\mathcal{M})$ if A''_i does not have an attacker that is labelled `undec` and a''_i is *undefined* in $\Gamma^*(\mathcal{M})$ if A''_i has an attacker that is labelled `undec`. Let $c'' \leftarrow a''_{i''}, \ldots, a''_{j''}, \text{not } b''_1, \ldots, \text{not } b''_{m''}$ ($0 \leq i'', j'' \leq n'', m'' \geq 0$) be a node that is $n' + 1$ distance from the furthest leaf in the tree of A. $b''_1, \ldots,$ $b''_{m''}$ are either *true* or *undefined* in $\mathcal{M}$ and $a''_{i''}, \ldots, a''_{j''}$ are either *true* or *undefined* in $\mathcal{M}$. Then the reduct of the node is $c'' \leftarrow v_1, \ldots, v_{j''-i''+1+m''}$ ($v_i \in \{t, u\}, 0 \leq i \leq j'' - i'' + 1 + m''$). If there is an attacker that is labelled `undec` the reduct of the node is $c'' \leftarrow v_1, \ldots, u, \ldots, v_{j''-i''+1+m''}$ ($v_i \in \{t, u\}, 0 \leq i \leq j''-i''+1+m''$). So in the least model of $\frac{P}{\mathcal{M}}$, c'' will be *undefined* in $\Gamma^*(\mathcal{M})$ if the subargument has an attacker that is labelled `undec`.

A has an attacker that is labelled `undec` and A is a subargument of A. Then in the least model of $\frac{P}{\mathcal{M}}$, c will be *undefined* in $\Gamma^*(\mathcal{M})$.

Since for any arbitrary atom c, it holds that $\mathcal{M}(c) = \Gamma^*(\mathcal{M})(c)$, it follows that $\mathcal{M} = \Gamma^*(\mathcal{M})$. Hence $\mathcal{M}$ is a fixed point of Γ^*, so $\mathcal{M}$ is a 3-valued stable model of P. ■

When $\mathcal{M}$ is a 3-valued stable model of a logic program, then $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$ is a complete labelling of the associated argumentation framework, as is stated by the following theorem.

Theorem 24. *Let P be a logic program and $\mathcal{M} =< T; F >$ be a 3-valued stable model of P. Let $\mathcal{L} = \mathcal{M}od2\mathcal{L}ab(\mathcal{M})$ and c be a ground atom. Then $\mathcal{L}$ is a complete labelling of AF_P such that $W(\mathcal{L})(c) =$* `in` *if $c \in T$, $W(\mathcal{L})(c) =$* `out` *if $c \in F$ and $W(\mathcal{L})(c) =$* `undec` *if $c \in \overline{\mathcal{M}}$.*

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

1. $\mathcal{L}(A) = \texttt{in}$.
   According to Definition 22, for each attacker B of A, $\texttt{Conc}(B) \in F$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ be the root of B. Then $c \in F$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that reduct of the rule is equivalent to $c \leftarrow f$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in B such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. It follows that B has an attacker whose conclusion is in T which implies B is labelled $\texttt{out}$. Since this holds for an arbitrary attacker B of A it follows that each attacker of A is labelled $\texttt{out}$. So A is legally $\texttt{in}$ in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

2. $\mathcal{L}(A) = \texttt{out}$.
   According to Definition 22, there exists an attacker B of A such that $\texttt{Conc}(B) \in T$. Then from the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that the reduct of the root of B is $\texttt{Conc}(B) \leftarrow t$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots,$ $\text{not } b_n$ be the root of B and $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of B. So $c \in T$ and $c \leftarrow t$. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. It follows that the conclusions of all attackers of B are in F which implies B is labelled $\texttt{in}$. So A is legally $\texttt{out}$ in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

3. $\mathcal{L}(A) = \texttt{undec}$.
   According to Definition 22, there is no attacker of A that has a conclusion that is in T and not all attackers of A have a conclusion that is in F.

   (a) Assume there is an attacker B of A that is labelled $\texttt{in}$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of B and $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of B. According to Definition 22, the conclusion of each attacker of B is *false* in $\mathcal{M}$. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. So the reduct of the root of B is equivalent to $c \leftarrow t$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that $c \in T$. Contradiction.

   (b) Assume all attackers of A are labelled $\texttt{out}$. Let B be an arbitrary attacker of A and $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ be the root of B. According to Definition 22, there exists an attacker of B whose conclusion is *true* in $\mathcal{M}$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in B such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. Then the reduct of the root of B is equivalent to $c \leftarrow f$. From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that $c \in F$. Then each attacker of A has a conclusion that is in F. Contradiction.

   Therefore there is no attacker of A that is labelled $\texttt{in}$ and not all attackers of A are labelled $\texttt{out}$. So A is legally $\texttt{undec}$ in $\mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

Since this holds for any arbitrary argument A, it follows that each argument that is $\texttt{in}$ is legally $\texttt{in}$, each argument that is $\texttt{out}$ is legally $\texttt{out}$, and each argument that is $\texttt{undec}$ is legally $\texttt{undec}$. Hence, $\mathcal{L}$ is a complete labelling of AF_P.

The next things to be proved is that (1) if $c \in T$ then $W(\mathcal{L})(c) = \texttt{in}$, (2) if $c \in F$ then $W(\mathcal{L})(c) = \texttt{out}$ and (3) if $c \in \overline{\mathcal{M}}$ then $W(\mathcal{L})(c) = \texttt{undec}$.

1. If $c \in T$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that there is a rule whose reduct is equivalent to $c \leftarrow t$. Let this rule be the root of an argument A which implies that $\texttt{Conc}(A) = c$. Let $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of A and $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k, l \geq 0)$ be an arbitrary rule of A. Then for all b'_j $(0 \leq j \leq l)$, $b'_j \in F$. It follows that the conclusions of all attackers of A are in F which implies that A is labelled $\texttt{in}$. So $\mathcal{L}(A) = \texttt{in} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{in}$.
2. If $c \in F$.
   From the fact that $\Gamma^*(\mathcal{M}) = \mathcal{M}$ it follows that each rule with c in the head has the reduct $c \leftarrow f$. Let $A \in Ar$ be an arbitrary argument such that $\texttt{Conc}(A) = c$ and $c \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n$ $(m, n \geq 0)$ be the root of A. Then the reduct of the root of A is equivalent to $c \leftarrow f$. Then there exists a rule $c' \leftarrow a'_1, \ldots, a'_k, \text{not } b'_1, \ldots, \text{not } b'_l$ $(k \geq 0, l \geq 1)$ in A such that there is a $b'_j \in T$ $(1 \leq j \leq l)$. It follows that A has an attacker whose conclusion is in T which implies A is labelled $\texttt{out}$. It follows that each argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{out}$. So $\mathcal{L}(A) = \texttt{out} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{out}$.
3. If $c \in \overline{\mathcal{M}}$ then there is no rule whose reduct is $c \leftarrow t$ and there is a rule whose reduct is $c \leftarrow u$. It follows that there is no argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{in}$ and there is an argument A such that $\texttt{Conc}(A) = c$ is labelled $\texttt{undec}$. So $\mathcal{L}(A) = \texttt{undec} = \max(\{\mathcal{L}(A') \mid A' \in Ar \wedge \texttt{Conc}(A') = c\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(c) = \texttt{undec}$. ■

When $\mathcal{L}ab2\mathcal{M}od$ and $\mathcal{M}od2\mathcal{L}ab$ 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 25. *Let P be a logic program and AF_P be the associated argumentation framework.*
Let $\mathcal{L}ab2\mathcal{M}od^r : \{\mathcal{L} \mid \mathcal{L}$ is a complete labelling of $AF_P\} \rightarrow \{\mathcal{M} \mid \mathcal{M}$ is

a 3-valued stable model of $P\}$ be a function defined by $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}) = \mathcal{L}ab2\mathcal{M}od(\mathcal{L})$. Let $\mathcal{M}od2\mathcal{L}ab^r : \{\mathcal{M} \mid \mathcal{M}$ is a 3-valued stable of model of$P\}$ $\rightarrow \{\mathcal{L} \mid \mathcal{L}$ is a complete labelling of $AF_P\}$ be a function defined by $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}) = \mathcal{M}od2\mathcal{L}ab(\mathcal{M})$.

$\mathcal{L}ab2\mathcal{M}od^r$ and $\mathcal{M}od2\mathcal{L}ab^r$ are bijective and are each other's inverses.

Proof. As every function that has an inverse is bijective, we only need to prove that $\mathcal{L}ab2\mathcal{M}od^r$ and $\mathcal{M}od2\mathcal{L}ab^r$ are each other's inverses. That is $(\mathcal{L}ab2\mathcal{M}od^r)^{-1} = \mathcal{M}od2\mathcal{L}ab^r$ and $(\mathcal{M}od2\mathcal{L}ab^r)^{-1} = \mathcal{L}ab2\mathcal{M}od^r$. Let $AF_P = (Ar, att)$ be an argumentation framework. We prove the following two things:

1. For every 3-valued stable model $\mathcal{M}$ of P it holds that $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M})) = \mathcal{M}$.
   Let $\mathcal{M}$ be a 3-valued stable model $\mathcal{M}$ of P.
   If $\mathcal{M}(c) = t$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{in}$ (Theorem 24), so c is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.
   If $\mathcal{M}(c) = f$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{out}$ (Theorem 24), so c is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.
   If $\mathcal{M}(c) = u$ then $W(\mathcal{M}od2\mathcal{L}ab^r)(c) = \texttt{undec}$ (Theorem 24), so c is *undefined* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{M}od2\mathcal{L}ab^r(\mathcal{M}))$.
2. For every complete labelling $\mathcal{L}$ of AF_P it holds that $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})) = \mathcal{L}$.
   Let $\mathcal{L}$ be a complete labelling of AF_P and let $A \in Ar$.
   If $\mathcal{L}(A) = \texttt{in}$ then each attacker of A is labelled $\texttt{out}$. Let B be an arbitrary attacker of A and $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{out} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(b) = \texttt{out}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = f$. Then for each attacker B of A, $\texttt{Conc}(B) = b$ is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. It follows from Definition 22 that A is labelled $\texttt{in}$ in $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))$. So $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{in}$.

   If $\mathcal{L}(A) = \texttt{out}$ then there is an attacker B of A that is labelled $\texttt{in}$. Let $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{in} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then $W(\mathcal{L})(b) = \texttt{in}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = t$. Then there is an attacker B of A such that $\texttt{Conc}(B) = b$ is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. It follows from Definition 22 that A is labelled $\texttt{out}$ in $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))$. So $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{out}$.

   If $\mathcal{L}(A) = \texttt{undec}$ then there is an attacker B of A that is labelled $\texttt{undec}$ and there is no attacker of A that is labelled $\texttt{in}$. Let $\texttt{Conc}(B) = b$, then $\mathcal{L}(B) = \texttt{undec} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b\} \cup \{\texttt{out}\})$. Then

$W(\mathcal{L})(b) = \texttt{undec}$. So $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})(b) = u$. Then there is an attacker B of A such that $\texttt{Conc}(B) = b$ is *undefined* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$.

Assume there is an attacker C of A such that $\texttt{Conc}(C)$ is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. Let $\texttt{Conc}(C) = b'$, then $W(\mathcal{L})(b') = \texttt{in}$. It follows that $\texttt{in} = \max(\{\mathcal{L}(B') \mid B' \in Ar \wedge \texttt{Conc}(B') = b'\} \cup \{\texttt{out}\})$. Then there is an attacker (C) of A such that $\texttt{Conc}(C) = b'$ and $\mathcal{L}(C) = \texttt{in}$. Then from the fact that $\mathcal{L}$ is a complete labelling it follows that $\mathcal{L}(A) = \texttt{out}$. Contradiction.

So there is no attacker of A has a conclusion that is *true* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$ and not each attacker of A has a conclusion that is *false* in $\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L})$. So $\mathcal{M}od2\mathcal{L}ab^r(\mathcal{L}ab2\mathcal{M}od^r(\mathcal{L}))(A) = \texttt{undec}$. ■

From Theorem 17 and Theorem 25, 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

The results presented in this paper show that the complete labellings are semantically equivalent to 3-valued stable models.

We transformed argumentation frameworks into logic programs and proved that the complete labellings of an argumentation framework coincide with 3-valued stable models of the associated logic programs. We can obtain the same correspondence between complete labellings and 3-valued stable models if we transform logic programs into argumentation frameworks. Since complete extensions and complete labellings are one-to-one related, complete extensions and 3-valued stable models stand in a one-to-one relationship to each other. Therefore, complete extensions and 3-valued stable models express the same concept in different ways.

Since complete extensions and 3-valued stable models are both used as bases for describing other semantics in abstract argumentation and logic programming, the currently proved equivalence between complete semantics and 3-valued stable model semantics could perhaps be used to prove other equivalences as well, between argumentation and logic programming semantics.

One particular topic for further study would for instance be the possible correspondence between the semi-stable extensions in abstract argumentation [3] and the L-stable models [8] in logic programming. Once established this equivalence 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.

Epilogue

In section 4.2, the transformation from logic programming to argumentation was done using arguments that have an internal structure. This internal structure was then used to determine the attack relationship. This approach is in line with Dung's argumentation theory, which is after all about *abstract* argumentation systems, meaning that it is a meta-theory of argumentation that abstracts from specific aspects of the underlying object level argumentation formalisms. In particular, it abstracts from the internal structure of the arguments and the nature of the attack relation. These need to be specified in order for the Dung-style argumentation theory to be "instantiated" into a full object-level argumentation formalism.

In this paper, we have chosen to do the translation from logic programming to argumentation using an instantiated object-level argumentation formalism. An interesting question is whether one could also perform the translation purely at the abstract level. In some cases, this would actually be possible, by applying the procedure of Section 4.1 in reverse order. Recall that the translation from argumentation to logic programming (Definition 12) produces a logic program where the arguments are represented by atoms such that each atom occurs in the head of exactly one rule, and the body of each rule consists of only weakly negated atoms. So if we have a logic program with these properties, we can directly transform it back into an argumentation framework. The problems begin when some atoms occur in the head of more than one rule (or in the head of no rule at all) or when the body of a rule contains non-negated atoms. One could, however, devise a program transformation that transforms a "general" logic program into the shape that is required for further transformation into an argumentation framework. Such a translation could be done by adding extra atoms to a logic program to deal with the non-negated atoms in the bodies of the rules and the occurrences of atoms in the heads of more than one rule. More specifically, this could be done in the following way. First, we would translate each rule containing non-negated literals in the body, like

$$c \leftarrow a_1, \ldots, a_n, \texttt{not}\ b_1, \ldots, \texttt{not}\ b_m$$

by replacing it by a rule

$$c \leftarrow \texttt{not}\ a_1^*, \ldots, \texttt{not}\ a_n^*, \texttt{not}\ b_1, \ldots, \texttt{not}\ b_m$$

and adding the additional rules

$$a_i^* \leftarrow \texttt{not}\ a_i\ (1 \leq i \leq n)$$

This yields a new program in which each rule contains only weakly negated literals in the body. The next step is to deal with literals that occur in the head of more than one rule. Suppose there are n rules ($n \geq 2$) with atom c in the head:

$$c \leftarrow \texttt{not}\ a_{i,1}, \ldots, \texttt{not}\ a_{i,m}\ (1 \leq i \leq n)$$

Then replace these rules by

$$c_i \leftarrow \texttt{not}\ a_{i,1}, \ldots, \texttt{not}\ a_{i,m}\ (1 \leq i \leq n)$$

and add the rule

$$c^{**} \leftarrow \texttt{not}\ c_1, \ldots, \texttt{not}\ c_n$$

as well as the rule

$$c \leftarrow \texttt{not}\ c^{**}$$

Similarly, for each atom c occurring in the program, but not in the head of any clause, add a clause

$$c^{**} \leftarrow$$

as well as a clause

$$c \leftarrow \texttt{not}\ c^{**}$$

This yields a program in which each literal is the head of exactly one rule, and in which the body of each rule consists only of weakly negated atoms. Such a program can be directly translated into an argumentation framework by the reverse process of Definition 12. This transformation takes place entirely on the abstract argumentation level, without the need for instantiated arguments.

The problem, however, is that the thus described transformation process, although intuitive, does not preserve the original meaning of the logic program, at least not from the perspective of the 3-valued stable model semantics. To see why things fail, consider the following logic program P.

$a \leftarrow b$
$b \leftarrow a$

In the 3-valued stable model semantics, this program has only one model: $< \emptyset; \{a, b\} >$. However, the translation process yields the following program.

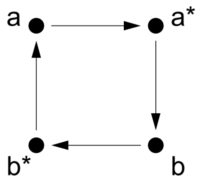

Figure 2. Argumentation framework associated with logic program P

$a \leftarrow \texttt{not}\ b^*$
$b^* \leftarrow \texttt{not}\ b$
$b \leftarrow \texttt{not}\ a^*$
$a^* \leftarrow \texttt{not}\ a$

This program can then be translated into the argumentation framework of Figure 2.

The program has three 3-valued stable models: $< \{a, b\}; \{a^*, b^*\} >$, $< \{a^*, b^*\}; \{a, b\} >$ and $< \emptyset, \emptyset >$. Only the second model corresponds with the meaning of the original program. Therefore the transformation process is not meaning-preserving, at least not from the perspective of the 3-valued stable model semantics.

Although one cannot rule out the existence of another transformation process that is meaning-preserving, such a process is likely to be more complex than the process described above. Our approach of instantiated arguments (Section 4.2) avoids these problems by using abstract argumentation the way it is intended: as a meta-level theory that is capable of describing instantiated argumentation formalisms by abstracting from some of their properties.

References

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

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

[3] Caminada, M.W.A., 'Semi-stable semantics', in P.E. Dunne and T.J.M. Bench-Capon, (eds.), *Computational Models of Argument; Proceedings of COMMA 2006*, IOS Press, 2006, pp. 121–130.

[4] Caminada, M.W.A., '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, Berlin, Springer Verlag, 2007, pp. 222–234.

[5] Caminada, M.W.A., and L. Amgoud, 'On the evaluation of argumentation formalisms', *Artificial Intelligence*, 171(5-6): 286–310, 2007.

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

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

[8] Eiter, T., N. Leone, and D. Saccà, 'On the partial semantics for disjunctive deductive databases', *Ann. Math. Artif. Intell.*, 19(1-2): 59–96, 1997.

[9] Gabbay, D. M., and A. Garcez, 'Logical modes of attack in argumentation networks', *Studia Logica*, 93(2-3): 199–230, 2009.

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

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

[12] Prakken, H., 'An abstract framework for argumentation with structured arguments', Technical Report UU-CS-2009-019, Department of Information and Computing Sciences, Utrecht University, 2009.

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

[14] Przymusinski, T. C., 'The well-founded semantics coincides with the three-valued stable semantics', *Fundamenta Informaticae*, 13(4): 445–463, 1990.

[15] Verheij, B., '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*, 2007, pp. 623–628.

[16] Vreeswijk, G.A.W., 'An algorithm to compute minimally grounded and admissible defence sets in argument systems', in P.E. Dunne and T.J.M. Bench-Capon, (eds.), *Computational Models of Argument; Proceedings of COMMA 2006*, IOS, 2006, pp. 109–120.

Yining Wu
Interdisciplinary Lab
for Intelligent and Adaptive Systems
University of Luxembourg
yining.wu@uni.lu

Martin Caminada
Interdisciplinary Lab
for Intelligent and Adaptive Systems
University of Luxembourg
martin.caminada@uni.lu

Dov M. Gabbay
King's College, London
Bar-Alan University, Israel
University of Luxembourg
dov.gabbay@kcl.ac.uk