On the Equivalence Between Abstract Dialectical Frameworks and Logic Programs

JOÃO ALCÂNTARA and SAMY SÁ  
Federal University of Ceará, Brazil  
(e-mails: jnando@lia.ufc.br, samy@ufc.br)

JUAN ACOSTA-GUADARRAMA  
Autonomous University of Juarez, Mexico  
(e-mail: juan.acosta@uacj.mx)

submitted 30 July 2019; accepted 31 July 2019

Abstract

Abstract Dialectical Frameworks (ADFs) are argumentation frameworks where each node is associated with an acceptance condition. This allows us to model different types of dependencies as supports and attacks. Previous studies provided a translation from Normal Logic Programs (NLPs) to ADFs and proved the stable models semantics for a normal logic program has an equivalent semantics to that of the corresponding ADF. However, these studies failed in identifying a semantics for ADF’s equivalent to a three-valued semantics (as partial stable models and well-founded models) for NLPs. In this work, we focus on a fragment of ADFs, called Attacking Dialectical Frameworks (ADF⁺s), and provide a translation from NLPs to ADF⁺s robust enough to guarantee the equivalence between partial stable models, well-founded models, regular models, stable models semantics for NLPs and respectively complete models, grounded models, preferred models, stable models for ADFs. In addition, we define a new semantics for ADF⁺s, called \( L \)-stable, and show it is equivalent to the \( L \)-stable semantics for NLPs.

KEYWORDS: Abstract Dialectical Frameworks, Normal Logic Programs, Argumentation, Instantiations of Argumentation Frameworks

1 Introduction

Logic Programming and Formal Argumentation Theory are two different formalisms widely used for the representation of knowledge and reasoning. The connection between them is especially clear when comparing the semantics proposed to each formalism. The first questions were raised and answered in (Dung 1995), the work that originally introduced Abstract Argumentation Frameworks (AAF): it was shown how to translate a Normal Logic Program (NLP) to an AAF and proved the stable models (resp. the well-founded model) of an NLP correspond to the stable extensions (resp. the grounded extension) of its corresponding AAF. Other advances were made when (Wu et al. 2009) pointed the equivalence between the complete semantics for AAF and the partial stable semantics for NLPs. Those semantics generalize many others, yielding a plethora of results gathered in (Caminada et al. 2015a). One equivalence formerly expected to hold, however, could not be achieved: the correspondence between the semi-stable semantics for AAFs (Caminada 2006) and the \( L \)-stable semantics for NLPs (Eiter et al. 1997).
Despite their success, AAFs are not immune to criticisms. A contentious issue refers to their alleged limited expressivity as they lack features which are common in almost every form of argumentation found in practice (Brewka and Woltran 2010). Indeed, in AAFs, the only interaction between atomic arguments is given by the attack relation.

With such a motivation, in (Brewka and Woltran 2010; Brewka et al. 2013) they defined Abstract Dialectical Frameworks (ADF$s), a generalization of AAFs, to express arbitrary relationships among arguments. In an ADF, besides the attack relation, arguments may support each other, or a group of arguments may jointly attack another while each argument in the group is not strong enough to do so. Such additional expressiveness arises by associating to each node (argument) its two-valued acceptance conditions which can get expressed as arbitrary propositional formulas. The intuition is that an argument is accepted if its associated acceptance condition is true.

A translation from NLPs to ADF$s is given in (Brewka and Woltran 2010), where they showed Stable Models Semantics for NLPs has an equivalent semantics for ADF$s. However, they did not identify a semantics for ADF$s equivalent to a 3-valued semantics (such as Partial Stable Models) for NLPs (Brewka and Woltran 2010; Strass 2013).

In this work, we will not only identify such semantics, but we will also ascertain only a fragment of ADF$s, called Attacking Abstract Dialectical Frameworks (ADF$^+$s), is needed. In fact, we will adapt the translation from NLPs to Abstract Argumentation proposed in (Wu et al. 2009; Caminada et al. 2015a) to provide a translation from NLPs to ADF$^+$s to account for various equivalences between their semantics. That includes to prove the equivalence between partial stable models, well-founded models, regular models, stable models semantics for NLPs and respectively complete models, grounded models, preferred models, stable models for ADF$s. Also, we define a new semantics for ADF$^+$s, called L-stable (for least-stable), and show it is equivalent to the L-stable Semantics for NLPs (Eiter et al. 1997). Hence, our results allow us to apply proof procedures and implementations for ADF$s to NLPs and vice-versa.

The paper proceeds as follows. Firstly we recall the basic definition of ADF$s and NLPs as well as some of their well-established semantics. Next, we consider the Attacking Abstract Dialectical Frameworks (ADF$^+$s), a fragment of ADF$s in which the unique relation involving arguments is the attack relation. In Section 4, we show a translation from NLPs to ADF$^+$s and prove the equivalence between partial stable models (NLPs) and complete models (ADF$^+$s), well-founded models (NLPs) and grounded models (ADF$^+$s), regular models (NLPs) and preferred models (ADF$^+$s), stable models (NLPs) and stable models ADF$^+$s, L-stable models (NLPs) and L-stable models (ADF$^+$s). In Section 5, we compare our results with previous attempts to translate NLPs into ADF$s and ADF$s into NLPs and we present a brief account on the main connections between NLPs and Abstract Argumentation Frameworks (Dung 1995)/Assumption-Based Argumentation (Dung et al. 2009) as well as a comparison between ADF$^+$ and SETAF (Nielsen and Parsons 2006), an extension of AAFs to allow joint attacks on arguments. Finally, we round off with a discussion of the obtained results and pointer for future works.

2 Background

2.1 Abstract Dialectical Frameworks

Abstract Dialectical Frameworks (ADF$s) have been designed in (Brewka and Woltran 2010; Brewka et al. 2013) to treat arguments (called statements there) as abstract and
atomic entities. One can see it as a directed graph whose nodes represent statements, which can get accepted or not. Besides, the links between nodes represent dependencies: the status (accepted/not accepted) of a node $s$ only depends on the status of its parents ($\text{par}(s)$), i.e., the nodes with a direct link to $s$. We will restrict ourselves to finite ADFs:

**Definition 1 (Abstract Dialectical Frameworks (Brewka and Woltran 2010))**

An abstract dialectical framework is a tuple $D = (S, L, C)$ where

- $S$ is a finite set of statements (positions, nodes);
- $L \subseteq S \times S$ is a set of links, and $\forall s \in S$, $\text{par}(s) = \{ t \in S \mid (t, s) \in L \}$;
- $C = \{ C_s \mid s \in S \}$ is a set of total functions $C_s : 2^{\text{par}(s)} \to \{ t, f \}$, one for each statement $s$. $C_s$ is called the acceptance condition of $s$.

The function $C_s$ is intended to determine the acceptance status of a statement $s$, which only depends on the status of its parent nodes $\text{par}(s)$. Intuitively, $s$ will be accepted if there exists $R \subseteq \text{par}(s)$ such that $C_s(R) = t$, which means every statement in $R$ is accepted while each statement in $\text{par}(s) - R$ is not accepted. The acceptance conditions in $C$ of an ADF $D = (S, L, C)$ can as well be represented in two alternative ways:

- Any function $C_s \in C$ can be represented by the set of subsets of $\text{par}(s)$ leading to acceptance, i.e., $C^t_s = \{ C^t_s \mid s \in S \}$, where $C^t_s = \{ R \subseteq \text{par}(s) \mid C_s(R) = t \}$. We will indicate this alternative by denoting an ADF as $(S, L, C^t)$.
- Any function $C_s \in C$ can also be represented as a classical two-valued propositional formula $\varphi_s$ over the vocabulary $\text{par}(s)$ as follows:

$$\varphi_s \equiv \bigvee_{R \subseteq \text{par}(s)} \left( \bigwedge_{a \in R} a \land \bigwedge_{b \in \text{par}(s) - R} \neg b \right). \quad (1)$$

If $C_s(\emptyset) = t$ and $\text{par}(s) = \emptyset$, we obtain $\varphi_s \equiv t$. If there is no $R \subseteq \text{par}(s)$ such that $C_s(R) = t$, then $\varphi_s \equiv f$. By $C^\varphi$ we mean the set $\{ \varphi_s \mid s \in S \}$. We will indicate this alternative by denoting an ADF as $(S, L, C^\varphi)$. We also emphasize any propositional formula $\varphi_s$ equivalent (in the classical two-valued sense) to the formula in Equation (1) can be employed to represent $C_s$.

When referring to an ADF as $(S, L, C^\varphi)$, we will assume the acceptance formulas implicitly specify the parents a node depends on. Then, the set $L$ of links between statements can be ignored, and the ADF can be represented as $(S, C^\varphi)$, where $L$ gets recovered by $(t, s) \in L$ iff $t$ appears in $\varphi_s$. In order to define the different semantics for ADFs over the set of statements $S$, we will resort to the notion of (3-valued) interpretations:

**Definition 2 (Interpretations and Models (Brewka and Woltran 2010))**

Let $D = (S, C^\varphi)$ be an ADF. A 3-valued interpretation (or simply interpretation) over $S$ is a mapping $v : S \to \{ t, f, u \}$ that assigns one of the truth values true (t), false (f) or unknown (u), to each statement. Interpretations will be extended to assign values to formulas over statements according to Kleene’s strong 3-valued logic (Kleene et al. 1952): negation switches $t$ and $f$, and leaves $u$ unchanged; a conjunction is $t$ if both conjuncts are $t$, it is $f$ if some conjunct is $f$ and it is $u$ otherwise; disjunction is dual. A 3-valued interpretation $v$ is a model of $D$ if for all $s \in S$ we have $v(s) \neq u$ implies $v(s) = v(\varphi_s)$.

Sometimes we will refer to an interpretation $v$ over $S$ as a set $V = \{ s \mid s \in S \text{ and } v(s) = t \} \cup \{ \neg s \mid s \in S \text{ and } v(s) = f \}$. Obviously, if neither $s \in V$ nor $\neg s \in V$, then $v(s) = u$. 

Downloaded from https://www.cambridge.org/core. IP address: 54.70.40.11, on 06 Oct 2019 at 08:25:57, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068419000280
Furthermore, the three truth values are partially ordered by \( \leq_i \) according to their information content: \( u <_i t \) and \( u <_i f \) and no other pair is in \( <_i \). The pair \( \{t, f, u\}, \leq_i \) forms a complete meet-semilattice with respect to \( \cap \). This meet can be read as consensus and assigns \( t \cap t = t, f \cap f = f \), and returns \( u \) otherwise.

The information ordering \( \leq_i \) extends as usual to interpretations \( v_1, v_2 \) over \( S \) such that \( v_1 \leq_i v_2 \) if \( v_1(s) \leq_i v_2(s) \) for all \( s \in S \). The set of all 3-valued interpretations over \( S \) forms a complete meet-semilattice with respect to \( \leq_i \). The consensus meet operation \( \cap \) of this semilattice is given by \( (v_1 \cap v_2)(s) = v_1(s) \cap v_2(s) \) for all \( s \in S \). The least element of this semilattice is the interpretation \( v \) such that \( v(s) = u \) for each \( s \in S \).

In (Brewka et al. 2013), the semantics for ADFs were defined via an operator \( \Gamma_D \):

**Definition 3 (\( \Gamma_D \) Operator (Brewka et al. 2013))**  
Let \( D = (S, L, C^v) \) be an ADF and \( v \) be a 3-valued interpretation over \( S \). We have  
\[
\Gamma_D(v)(s) = \bigcap \{ w(\varphi_s) \mid w \in [v_2] \},
\]
in which \([v_2] = \{w \mid v \leq_i w \text{ and for each } s \in S, w(s) \in \{t, f\}\} \).

Each element in \([v_2] \) is a 2-valued interpretation extending \( v \). The elements of \([v_2] \) form an \( \leq_i \)-antichain with greatest lower bound \( v = \bigcap [v_2] \). For each \( s \in S \), \( \Gamma_D \) returns the consensus truth value for \( \varphi_s \), where the consensus takes into account all possible 2-valued interpretations \( w \) extending \( v \). If \( v \) is 2-valued, we get \([v_2] = \{v\}\). In this case, \( \Gamma_D(v)(s) = v(\varphi_s) \) and \( v \) is a 2-valued model for \( D \) iff \( \Gamma_D(v) = v \). As \([v_2] \) has only 2-valued interpretations, if \( \varphi_s^1 \) is equivalent to \( \varphi_s^2 \) in the classical two-valued sense, it is clear  
\[
\bigcap \{ w(\varphi_s^1) \mid w \in [v_2] \} = \bigcap \{ w(\varphi_s^2) \mid w \in [v_2] \}.
\]

That means when defining \( \Gamma_D \) operator, it does not matter the acceptance formula we choose as far as it is equivalent in the classical 2-valued sense. In addition, \( \Gamma_D \) operator can be employed to characterize also complete interpretations:

**Definition 4 (Complete Interpretations (Brewka et al. 2013))**  
Let \( D = (S, L, C^v) \) be an ADF and \( v \) be a 3-valued interpretation over \( S \). We state \( v \) is a complete interpretation of \( D \) iff \( v = \Gamma_D(v) \).

As shown in (Brewka and Woltran 2010), \( \Gamma_D \) operator is \( \leq_i \)-monotonic. Then a \( \leq_i \)-least fixpoint of \( \Gamma_D \) is always guaranteed to exists for every ADF \( D \). Note complete interpretations of \( D \) are also models of \( D \). For this reason, they are also called complete models. The notion of reduct borrowed from logic programming (Gelfond and Lifschitz 1988) is reformulated to deal with ADFs:

**Definition 5 (Reduct (Brewka et al. 2013))**  
Let \( D = (S, L, C^v) \) be an ADF and \( v \) be a 2-valued model of \( D \). The reduct of \( D \) with \( v \) is given by the ADF, \( D^v = (E_v, L^v, C^v) \), in which \( E_v = \{s \in S \mid v(s) = t\} \), \( L^v = L \cap (E_v \times E_v) \), and \( C^v = \{\varphi_s^v \mid s \in E_v \text{ and } \varphi_s^v = \varphi_s[b/f : v(b) = f]\} \); i.e., in each acceptance formula, \( \varphi_s^v \), we replace in \( \varphi_s \) every statement \( b \in S \) by \( f \) if \( v(b) = f \).

We can now define some of the main semantics for an ADF as follows:

---

1 A complete meet-semilattice is such that every non-empty finite subset has a greatest lower bound, the meet; and every nonempty directed subset has a least upper bound. A subset is directed if any two of its elements have an upper bound in the set.
Definition 6 (Semantics (Brewka et al. 2013))

Let \( D = (S, L, C^e) \) be an ADF, and \( v \) a model of \( D \). We state that

- \( v \) is a grounded model of \( D \) iff \( v \) is the \( \leq_i \)-least complete model of \( D \).
- \( v \) is a preferred model of \( D \) iff \( v \) is a \( \leq_i \)-maximal complete model of \( D \).
- \( v \) is a stable model of \( D \) iff \( v \) is a 2-valued model of \( D \) such that \( v \) is the grounded model of \( D^v = (E_v, L^v, C^v) \).

We proceed by displaying an example to illustrate these semantics:

Example 7

Consider the ADF, \( D = (S, C^e) \), given by \( a \neg b \mid b \neg a \mid c \neg b \land c \mid d \neg c \mid e \neg d \), where \( S = \{a, b, c, d, e\} \) and the acceptance formula of each \( s \in S \) is written in square brackets on the right of \( s \). As for the semantics for \( D \), we have a) \( \{a, \neg b\}, \{b, d, \neg a, \neg c, \neg e\} \) and \( \emptyset \) are its complete models; b) \( \emptyset \) is its grounded model; c) \( \{a, \neg b\}, \{b, d, \neg a, \neg c, \neg e\} \) are its preferred models; d) \( \{b, d, \neg a, \neg c, \neg e\} \) is its unique stable model.

Notice some ADFs have no stable models. For instance, in an ADF whose unique statement is \( a \neg a \), there is no stable model. Furthermore, an ADF can have more than one stable model as in the ADF represented by \( a \neg b \) and \( b \neg a \), which has \( \{a, \neg b\} \) and \( \{b, \neg a\} \) for its stable models. In contrast, the grounded model is unique for each ADF (see (Brewka and Woltran 2010; Brewka et al. 2013)).

2.2 Normal Logic Programs

Now we will focus on propositional normal logic programs. We assume the reader is familiar with the Stable Model Semantics (Gelfond and Lifschitz 1988).

Definition 8

A Normal Logic Program (NLP), \( P \), is a set of rules of the form \( a \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n \) \( (m, n \in \mathbb{N}) \), where \( a, a_i \ (1 \leq i \leq m) \) and \( b_j \ (1 \leq j \leq n) \) are atoms; \text{not} represents default negation, and \text{not} \( b_j \) is a default literal. We say \( a \) is the head of the rule, and \( a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n \) is its body. The Herbrand Base of \( P \) is the set \( HB_P \) of all atoms occurring in \( P \).

A wide range of logic programming semantics can be defined based on the 3-valued interpretations (for short, interpretations) of programs:

Definition 9 (Interpretation and Models (Przymusinski 1990))

A 3-valued interpretation, \( I \), of an NLP, \( P \), is a total function \( I : HB_P \rightarrow \{t, f, u\} \). We say \( I \) is a model of \( P \) iff for each rule \( a \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n \in P, min \{I(a_1), \ldots, I(a_m), \neg I(b_1), \ldots, \neg I(b_n)\} \leq I(a) \), where \( \neg t = f, \neg f = t \) and \( \neg u = u \).

When convenient, we will refer to an interpretation \( I \) of \( P \) as a set \( \mathcal{I} = \{a \mid HB_P \text{ and } I(a) = t\} \cup \{a \mid a \in HB_P \text{ and } I(a) = f\} \). If neither \( a \in \mathcal{I} \) nor \( \neg a \in \mathcal{I} \), then \( I(a) = u \).

Besides the information ordering \( \leq_i \), it is worth mentioning here the truth ordering \( \leq_t \) given by \( f <_t u <_t t \). The truth ordering \( \leq_t \) extends as usual to interpretations \( I_1, I_2 \) over \( HB_P \) such that \( I_1 \leq_i I_2 \) iff \( I_1(a) \leq_t I_2(a) \) for all \( a \in HB_P \). We also emphasize the notions of model of a logic program and model of an ADF follow distinct motivations: the models of a logic program are settled on \( \leq_t \) whereas the models of an ADF are settled on \( \leq_i \). In order to avoid confusion, we will let it explicit when referring to one of them.
Now we will consider the main semantics for NLPs. Let $P$ be a 3-valued interpretation of a program $P$; take $P/I$ to be the program built by the execution of the following steps:

1. Remove any $a \leftarrow a_1, \ldots, a_m$, $\$not\ b_1, \ldots, \$not\ b_n \in P$ such that $I(b_i) = t$ for some $i$ $(1 \leq i \leq n)$;
2. Afterwards, remove any occurrence of $\$not\ b_i$ from $P$ such that $I(b_i) = f$.
3. Then, replace any occurrence of $\$not\ b_i$ left by a special atom $u$ ($u \notin HB_P$).

Note $u$ is assumed to be unknown in each interpretation of $P$. As shown in (Przymusinski 1990), $P/I$ has a unique $\leq_i$-least 3-valued model, obtained by the $\Psi$ operator:

**Definition 10 ($\Psi^P_T$ Operator (Przymusinski 1990))**

Let $P$ be an NLP, $I$ and $J$ be interpretations of $P$ and $a \in HB_P$ an atom in $P$. Define $\Psi^P_T(J)$ to be the interpretation given by

- $\Psi^P_T(J)(a) = t$ if $a \leftarrow a_1, \ldots, a_m \in P/I$ and for all $i, 1 \leq i \leq m$, $J(a_i) = t$;
- $\Psi^P_T(J)(a) = f$ if for every $a \leftarrow a_1, \ldots, a_m \in P/I$, there exists $i, 1 \leq i \leq m$, such that $J(a_i) = f$;
- $\Psi^P_T(J)(a) = u$ otherwise.

Indeed, the $\leq_i$-least model of $P_T$, denoted by $\Omega_P(I)$, is given by the least fixed point of $\Psi^P_T$ iteratively obtained as follows for finite logic programs:

$$
\Psi^P_T(J)(a) = t \iff a \leftarrow a_1, \ldots, a_m \in P/I \text{ and for all } i, 1 \leq i \leq m, J(a_i) = t;
$$

$$
\Psi^P_T(J)(a) = f \iff \forall a \leftarrow a_1, \ldots, a_m \in P/I, \exists i, 1 \leq i \leq m, J(a_i) = f;
$$

$$
\Psi^P_T(J)(a) = u \iff a \notin HB_P.
$$

in which $\bot$ is an interpretation such that for each $a \in HB_P$, $(a) = f$. According to (Przymusinski 1990), there exists $n \in N$ such that $\Omega_P(I) = \Psi^P_T J = \Psi^P_T J = \Psi^P_T J$. We now specify the logic programming semantics to be examined in this paper.

**Definition 11**

Let $P$ be an NLP and $I$ be an interpretation:

- $I$ is a partial stable model (PSM) of $P$ iff $I = \Omega_P(I)$ (Przymusinski 1990).
- $I$ is a well-founded model of $P$ iff $I$ is the $\leq_i$-least PSM of $P$ (Przymusinski 1990).
- $I$ is a regular model of $P$ iff $I$ is a $\leq_i$-maximal PSM of $P$ (Eiter et al. 1997).
- $I$ is a stable model of $P$ iff $I$ is a PSM of $P$ where for each $a \in HB_P$, $I(a) \in \{t, f\}$ (Przymusinski 1990).
- $I$ is an $L$-stable model of $P$ iff $I$ is a PSM of $P$ with minimal $\{a \in HB_P \mid I(a) = u\}$ (w.r.t. set inclusion) among all partial stable models of $P$ (Eiter et al. 1997).

**Example 12**

Consider the NLP $P$:

$$
\begin{align*}
  b &\leftarrow c, \$not\ a \quad a \leftarrow \$not\ b \quad c \leftarrow d \quad p \leftarrow c, d, \$not\ p \quad p \leftarrow \$not\ a \quad d \leftarrow
\end{align*}
$$

Concerning the semantics of $P$, we have a) Partial stable models: $\{c, d\}, \{b, c, d, p, \$not\ a\}$ and $\{a, c, d, \$not\ b\}$; b) Well-founded model: $\{c, d\}$; c) Regular models: $\{b, c, d, p, \$not\ a\}$ and $\{a, c, d, \$not\ b\}$; d) Stable model and $L$-Stable model: $\{b, c, d, p, \$not\ a\}$.
In the next section, we will focus on a fragment of ADFs, dubbed Attacking Abstract Dialectical Frameworks (\(ADF^+\)s), and in the sequel we will show that \(ADF^+\)s are enough to capture any semantics based on partial stable models as those above mentioned.

### 3 Attacking Abstract Dialectical Frameworks

Now we consider the Attacking Abstract Dialectical Frameworks (\(ADF^+\)s), a fragment of ADFs in which the unique relation involving statements is the attack relation. We may note parenthetically some definitions related to ADFs become simpler when restricted to \(ADF^+\)s. We proceed by recalling the notions of supporting and attacking links:

**Definition 13 (Supporting and Attacking Links (Brewka and Woltran 2010))**

Let \(D = (S, L, C)\) be an ADF. A link \((r, s) \in L\) is

- **supporting in** \(D\) iff for no \(R \subseteq \text{par}(s)\) we have \(C_s(R) = t\) and \(C_s(R \cup \{r\}) = f\).
- **attacking in** \(D\) iff for no \(R \subseteq \text{par}(s)\) we have \(C_s(R) = f\) and \(C_s(R \cup \{r\}) = t\).

Formally, a link \((r, s)\) is **redundant** if it is both attacking and supporting. Redundant links can be deleted from an ADF as they mean no real dependencies (Brewka and Woltran 2010). Again in (Brewka and Woltran 2010), the authors introduced the Bipolar Abstract Dialectical Frameworks (\(BADF\)), a subclass of ADFs in which every link is either supporting or attacking. Now we regard a subclass of \(BADF\)s in which only attacking links are admitted:

**Definition 14 (ADF+)**

An Attacking Abstract Dialectical Framework, denoted by \(ADF^+\), is an ADF \((S, L, C)\) such that every \((r, s) \in L\) is an attacking link. This means that for every \(s \in S\), if \(C_s(M) = t\), then for every \(M' \subseteq M\), we have \(C_s(M') = t\).

In an \(ADF^+\) \((S, L, C)\), for each \(s \in S\), its acceptance formula \(\varphi_s\) can be simplified:

**Theorem 15**

Let \(D = (S, L, C^t)\) be an \(ADF^+\) and, for every \(s \in S\), we define \(C^\text{max}_s = \{ R \in C^t_s \mid \text{there is no } R' \in C^t_s \text{ such that } R \subset R' \}\). Then, for every \(s \in S\),

\[
\varphi_s \equiv \bigvee_{R \in C^\text{max}_s} \bigwedge_{b \in \text{par}(s) - R} \neg b.
\]

Hence, in \(ADF^+\)s, every acceptance formula corresponds to a propositional formula in the disjunctive normal form, where each disjunct is a conjunction of negative atoms. Notice replacing an acceptance formula by a two-valued equivalent one does not change complete semantics, and we are not interested in the three-valued models of the \(ADF^+\). The importance of these formulas will be evident below. Before, however, note \(ADF^+\) does not prohibit redundant links. For instance, consider the \(ADF\) \(D = (S, L, C)\), in which \(S = \{a, b, c\}\), \(L = \{(b, a), (c, a)\}\) and \(C^t_a = \{\{b\}\}\) and \(C^t_b = C^t_c = \{\emptyset\}\). We know \(D\) is an \(ADF^+\) as both \((b, a)\) and \((c, a)\) are attacking links. In addition, \((b, a)\) is a redundant link as it is also supporting. Redundant links can be easily identified in \(ADF^+\)s:

---

2 All proofs of the results presented in this paper can be found on [https://arxiv.org/pdf/1907.09548](https://arxiv.org/pdf/1907.09548)
Let \( D = (S, L, C^t) \) be an \( ADF^+ \). A link \((r, s) \in L\) is redundant iff \( r \in R \) for every \( R \in C^t_s \).

A straightforward consequence from Theorem 16 is that in \( ADF^+\)s, every acceptance formula \( \varphi_s \) in the disjunctive normal form as in Theorem 15, where each disjunct is a conjunction of negative atoms, disregards redundant links:

**Corollary 17**

Let \( D = (S, L, C^t) \) be an \( ADF^+ \). For each \( s \in S \), if \( \varphi_s \) is \( \bigvee_{R \in C^t_s} \bigwedge_{b \in \text{par}(s)-R} \neg b \) and \( L' = \{(r, s) \mid \neg r \text{ appears in } \varphi_s\} \), then \( L' \) has no redundant link.

**Example 18**

Let us recall the \( ADF^+ \) \( D = (S, L, C) \) above in which \( S = \{a, b, c\} \), \( L = \{(b, a), (c, a)\} \) and \( C^t_a = \{\{b\}, \emptyset\} \) and \( C^t_b = C^t_c = \{\emptyset\} \). With the general representation for \( \varphi_s \) in \( ADFs \), in which for every \( s \in S \), \( \varphi_s = \bigvee_{R \in C^t_s} \left( \bigwedge_{a \in R} a \land \bigwedge_{b \in \text{par}(s)-R} \neg b \right) \), we get \( a[(b \land \neg c) \lor (-b \land \neg c)] \) \( b[t] \) \( c[t] \). With the simpler representation for acceptance formulas given by Theorem 15, in which \( \varphi_s = \bigvee_{R \in C^t_s} \bigwedge_{b \in \text{par}(s)-R} \neg b \), we get \( a[\neg c] \) \( b[t] \) \( c[t] \). As expected, the redundant link \((b, a)\) is not taken into account to define \( \varphi_a \) as \( \neg c \).

Alternatively, redundant links in \( ADF^+\)s have the following property:

**Theorem 19**

Let \( D = (S, L, C^t) \) be an \( ADF^+ \), \( s \in S \); \( r \in \text{par}(s) \) and \( C^t_s(r) = \{ R \in C^t_s \mid r \in R \} \). A link \((r, s) \in L\) is redundant iff \( |C^t_s(r)| = \frac{|C^t_s|}{2} \).

Thus, identifying redundant links in an \( ADF^+ \) has a sub-quadratic time complexity on \( |C^t_s|\):

**Corollary 20**

Let \( D = (S, L, C^t) \) be an \( ADF^+ \). Deciding if a link \((r, s) \in L\) is redundant can be solved in sub-quadratic time on \( |C^t_s|\).

In contrast, identifying redundant links in \( ADFs \) is coNP-hard (Ellmauthaler 2012).

In Subsection 2.1, \( \Gamma_D \) operator is employed to define the semantics for \( ADF \). When restricted to \( ADF^+\)s, it assumes a simpler version:

**Theorem 21**

Let \( D = (S, L, C^\varphi) \) be an \( ADF^+ \), \( v \) be a 3-valued interpretation over \( S \), and for each \( s \in S \), \( \varphi_s \) is the formula \( \bigvee_{R \in C^\varphi_s} \bigwedge_{b \in \text{par}(s)-R} \neg b \) depicted in Theorem 15. It holds for every \( s \in S \), \( \Gamma_D(v)(s) = v(\varphi_s) \).

Besides being noticeably simpler when restricted to \( ADF^+ \), this new characterization of \( \Gamma_D \) might mean lower complexity of reasoning. In (Brewka et al. 2013), the problem of verifying whether a given interpretation is complete is proved to be DP-complete. In our case, owing to our definition of \( \Gamma_D \), this problem can get solved by assigning values to formulas over statements according to Kleene’s strong 3-valued logic. This evaluation
procedure is similar to (and has the same complexity as) that for Boolean formulas, which takes polynomial time (Buss 1987). We run this procedure for each statement in a given \( ADF \). Then, the overall algorithm runs in polynomial time. It is a promising result as the complexity of many reasoning tasks on \( ADF^+ \)’s may likely have the same complexity as standard Dung’s AAFs (Dung 1995). A consequence from Theorem 21 is the stable models of an \( ADF^+ \) \( D \) can get characterized as the two-valued complete models of \( D \):

**Theorem 22**

Let \( D = (S, L, C^\sigma) \) be an \( ADF^+ \). Then \( v \) is a stable model of \( D \) iff \( v \) is a 2-valued complete model of \( D \).

The main objective of this work is to show each semantics for NLPs presented in Subsection 2.2 has an equivalent one for \( ADF^+ \). Then we need to define a new semantics for \( ADF^+ \), which will be proved in the next section to be equivalent to the \( L \)-stable models semantics for NLPs:

**Definition 23 (L-stable)**

Let \( D = (S, L, C^\sigma) \) be an \( ADF^+ \), and \( v \) be a 3-valued interpretation of \( D \). We say \( v \) is an \( L \)-stable model of \( D \) iff \( v \) is a complete model with minimal \( \text{unk}(v) = \{s \in S \mid v(s) = u\} \) (w.r.t. set inclusion) among all complete models of \( D \).

Note \( L \)-stable Models semantics is defined for every \( ADF^+ \) and the \( L \)-stable models of an \( ADF^+ \) \( D \) will coincide with its stable models whenever \( D \) has at least one stable model. Indeed we can see a stable model \( v \) as an \( L \)-stable model in which \( \text{unk}(v) = \emptyset \).

**Example 24**

Consider the \( ADF^+ \) \( D = (S, C^\sigma) \) given by

\[
\begin{align*}
  a[-b] & \quad b[-a] & \quad c[\neg c \wedge \neg a] & \quad (\neg c \wedge \neg d) & \quad d[-d] & \quad e[\neg e \wedge \neg b],
\end{align*}
\]

where \( S = \{a, b, c, d, e\} \), and the acceptance formula of each statement \( s \in S \) is written in square brackets on the right of \( s \). As for the semantics of \( D \), a) \( \{a, \neg b\} \), \( \{b, \neg a, \neg e\} \) and \( \emptyset \) are its complete models; b) \( \emptyset \) is its grounded model; c) \( \{a, \neg b\} \) and \( \{b, \neg a, \neg e\} \) are its preferred models; d) \( D \) has no stable model; e) \( \{b, \neg a, \neg e\} \) is its unique \( L \)-stable model.

Thus none of these semantics for \( ADF^+ \) are equivalent to each other. However, in the sequel, we will show some equivalences between NLPs semantics and \( ADF^+ \) semantics.

### 4 Equivalence Between ADF and Logic Programs

We will show one particular translation from NLP to \( ADF^+ \) is able to account for a whole range of equivalences between their semantics. This includes to prove the equivalence between NLP partial stable models and \( ADF^+ \) complete models, NLP well-founded models and \( ADF^+ \) grounded models, NLP regular models and \( ADF^+ \) preferred models, NLP stable models and \( ADF^+ \) stable models, NLP \( L \)-stable models and \( ADF^+ \) \( L \)-stable models. Our treatment is based on a translation from NLP to Abstract Argumentation proposed in (Wu et al. 2009; Caminada et al. 2015a), where each NLP rule is directly translated into an argument. In contradistinction, we will adapt it to deal with \( ADF \) by translating each rule into a substatement, and then, substatements corresponding to rules with the same head are gathered to constitute a unique statement. Taking a particular NLP \( P \), one can start to construct substatements recursively as follows:
**Definition 25 (Substatement)**

Let $P$ be an NLP.

- If $a$ is a rule (fact) in $P$, then it is also a substatement (say $r$) in $P$ with $\text{Conc}_P(r) = a$, $\text{Rules}_P(r) = \{a\}$ and $\text{Sup}_P(r) = \{\}$.
- If $a ← \not b_1, \ldots, \not b_n$ is a rule in $P$, then it is also a substatement (say $r$) in $P$ with $\text{Conc}_P(r) = a$, $\text{ Rules}_P(r) = \{a ← \not b_1, \ldots, \not b_n\}$ and $\text{Sup}_P(r) = \{-b_1, \ldots, -b_n\}$.
- If $a ← a_1, \ldots, a_m, \not b_1, \ldots, \not b_n$ is a rule in $P$ and for each $a_i$ ($1 ≤ i ≤ m$) there exists a substatement $r_i$ in $P$ with $\text{Conc}_P(r_i) = a_i$ and $a ← a_1, \ldots, a_m, \not b_1, \ldots, \not b_n$ is not contained in $\text{Rules}_P(r_i)$, then $a ← r_1, \ldots, r_m, \not b_1, \ldots, \not b_n$ is a substatement (say $r$) in $P$ with $\text{Conc}_P(r) = a$, $\text{Rules}_P(r) = \text{Rules}_P(r_1) \cup \cdots \cup \text{Rules}_P(r_m) \cup \{a ← a_1, \ldots, a_m, \not b_1, \ldots, \not b_n\}$ and $\text{Sup}_P(r) = \text{Sup}_P(r_1) \cup \cdots \cup \text{Sup}_P(r_m) \cup \{-b_1, \ldots, -b_n\}$.
- Nothing more is a substatement in $P$.

For a substatement $r$ in $P$, $\text{Sup}_P(r)$ is referred to as the support of $r$ in $P$. Besides, for each substatement $r$ in $P$, we can also define $\text{Sup}_P(r)$ iteratively as follows:

$$\text{Sup}_P^0(r) = \emptyset$$

$$\text{Sup}_P^{i+1}(r) = \{-b_1, \ldots, -b_n\} \cup \text{Sup}_P^i(r_1) \cup \cdots \cup \text{Sup}_P^i(r_m)$$

such that $r$ is a substatement $a ← r_1, \ldots, r_m, \not b_1, \ldots, \not b_n$ in $P$, $a ← a_1, \ldots, a_m$, $\not b_1, \ldots, \not b_n ∈ \text{Rules}_P(r)$ and $∀a_i$ ($1 ≤ i ≤ m$) there exists a substatement $r_i$ in $P$ with $\text{Conc}_P(r_i) = a_i$. Note for each substatement $r$ in $P$, $∃k ∈ \mathbb{N}$ such that $\text{Sup}_P(r) = \text{Sup}_P^k(r)$. This notion of support is generalized to obtain the support of an atom in $P$:

**Definition 26 (Support)**

Let $P$ be an NLP over a set $A$ of atoms. For each $a ∈ A$, we define the support of $a$ in $P$ as $\text{Sup}_P(a) = \{\text{Sup}_P(r) \mid r$ is a substatement in $P$ such that $\text{Conc}_P(r) = a\}$.

**Example 27**

Consider the normal logic program $P$ from Example 12:

$$b ← c, \not a \quad a ← \not b \quad c ← d \quad p ← c, d, \not p \quad p ← \not a \quad d ←$$

We can obtain the following substatements:

- $r_1: \quad d ←$  
- $r_2: \quad c ← r_1$  
- $r_3: \quad p ← r_2, r_1, \not p$  
- $r_4: \quad a ← \not b$  
- $r_5: \quad p ← \not a$  
- $r_6: \quad b ← r_2, \not a$.

Thus:

- $\text{Sup}_P(r_1) = \{\}$  
- $\text{Sup}_P(r_3) = \{-p\}$  
- $\text{Sup}_P(r_5) = \{-a\}$  
- $\text{Sup}_P(r_2) = \{\}$  
- $\text{Sup}_P(r_4) = \{-b\}$  
- $\text{Sup}_P(r_6) = \{-a\}$.

And:

- $\text{Sup}_P(a) = \{\{-b\}\}$  
- $\text{Sup}_P(b) = \{\{-a\}\}$  
- $\text{Sup}_P(p) = \{\{-p\}, \{-a\}\}$  
- $\text{Sup}_P(c) = \{\emptyset\}$  
- $\text{Sup}_P(d) = \{\emptyset\}$.

After that, we can construct the corresponding ADF as follows:
On the Equivalence Between ADFs and NLPs

**Definition 28**
Let $P$ be an NLP over a set $A$ of atoms. Define an ADF $\Xi(P) = (A, L, C^t)$, in which

- $L = \{(b, a) \mid B \in \text{Sup}_P(a) \text{ and } \neg b \in B\}$;
- For each $a \in A$, $C^t_a = \left\{B' \subseteq \{b \in \text{par}(a) \mid \neg b \notin B\} \mid B \in \text{Sup}_P(a)\right\}$.

We can prove the resulting ADF $\Xi(P)$ is indeed an $ADF^+$:

**Proposition 29**
Let $P$ be an NLP. The corresponding $\Xi(P)$ is an $ADF^+$.

Hence, the acceptance condition for each statement in $\Xi(P)$ can be retrieved as follows:

**Proposition 30**
Let $P$ be an NLP and $\Xi(P) = (A, L, C^t)$ the corresponding $ADF^+$. The acceptance condition $\varphi_a$ for each $a \in A$ is given by

$$\varphi_a \equiv \bigvee_{B \in \text{Sup}_P(a)} \left(\bigwedge_{\neg b \in B} \neg b\right).$$

In particular, if $\text{Sup}_P(a) = \{\emptyset\}$, then $\varphi_a \equiv t$ and if $\text{Sup}_P(a) = \emptyset$, then $\varphi_a \equiv f$.

**Example 31**
Recalling the NLP $P$ in Example 27, we obtain $ADF^+ \Xi(P) = (A, L, C^t)$, in which $A = \{a, b, c, d, p\}$; $L = \{(b, a), (a, b), (p, p), (a, p)\}$; $C^t_a = C^t_b = C^t_c = C^t_d = \{\emptyset\}$ and $C^t_p = \{\{a\}, \{p\}, \emptyset\}$. The acceptance condition for each statement in $\Xi(P)$ is given below:

$$a[\neg b] \quad b[\neg a] \quad c[t] \quad d[t] \quad p[\neg p \vee \neg a].$$

Concerning the semantics of $\Xi(P)$, we have

- Complete models: $\{c, d\}$, $\{b, c, d, p, \neg a\}$ and $\{a, c, d, \neg b\}$;
- Grounded model: $\{c, d\}$;
- Preferred models: $\{b, c, d, p, \neg a\}$ and $\{a, c, d, \neg b\}$;
- Stable model and $L$-stable model: $\{b, c, d, p, \neg a\}$.

Now we can prove one of the main results of this paper: Partial Stable Models are equivalent to Complete Models.

**Theorem 32**
Let $P$ be an NLP and $\Xi(P)$ be the corresponding $ADF^+$. $v$ is a partial stable model of $P$ iff $v$ is a complete model of $\Xi(P)$.

With this equivalence showed in Theorem 32, the following results are immediate:

**Theorem 33**
Let $P$ be an NLP and $\Xi(P) = (A, L, C^t)$ the corresponding $ADF^+$. We have

- $v$ is a well-founded model of $P$ iff $v$ is a grounded model of $\Xi(P)$.
- $v$ is a regular model of $P$ iff $v$ is a preferred model of $\Xi(P)$.
- $v$ is a stable model of $P$ iff $v$ is a stable model of $\Xi(P)$.
- $v$ is an $L$-stable model of $P$ iff $v$ is an $L$-stable model of $\Xi(P)$.

From Theorems 32 and 33, we see the NLP $P$ from Example 12 and the corresponding $ADF^+ \Xi(P)$ from Example 31 produce the same semantics. This result sheds light on the
connections between \( ADFs \) and \( NLPs \). Until now, it was unclear if any \( ADF \) semantics could capture a 3-valued one for \( NLPs \). Theorem 33 ensures the translation from \( NLP \) to \( ADF \) in Definition 28 is robust enough to guarantee at least the equivalence between any semantics based on partial stable models (at the \( NLP \) side) with any semantics based on complete models (at the \( ADF \) side).

## 5 Related Works

The relation between \( NLP \) and formal argumentation goes back to works such as (Prakken and Sartor 1997; Simari and Loui 1992; Dung 1995). In the sequel, we will describe previous attempts to translate \( ADFs \) to \( NLPs \) (Subsection 5.1) and \( NLPs \) to \( ADFs \) (Subsection 5.2) and the main connections between \( NLPs \) and other argument-based frameworks such as Abstract Argumentation Frameworks (\( AAFs \)) (Dung 1995) and Assumption-Based Argumentation (ABA) (Dung et al. 2009) in Subsection 5.3. Afterwards, we compare an extension of \( AAF \), called \( SETAF \) (Nielsen and Parsons 2006), with \( ADF^+ \).

### 5.1 From \( ADF \) to Logic Programming

As pointed out by (Strass 2013), there is a direct translation from \( ADFs \) to \( NLPs \):

**Definition 34** ((Strass 2013))

Let \( \Xi = (S, L, C^t) \) be an \( ADF \). Define the corresponding \( NLP \) \( P(\Xi) = \{ s \leftarrow a_1, \ldots, a_m, \text{not } b_1, \ldots, \text{not } b_n \mid s \in S, \{a_1, \ldots, a_m\} \in C^t_s \text{ and } \{b_1, \ldots, b_n\} = \text{par}(s) - \{a_1, \ldots, a_m\} \}. \)

Note the body of a rule for \( s \) is satisfied by an interpretation \( I \) whenever for some \( R \subseteq C^t_s \), the statements in \( R \) are \( t \) in \( I \) and the remaining parents of \( s \) are \( f \) in \( I \).

**Example 35**

The \( NLP \) \( P(\Xi) \) corresponding to the \( ADF \) \( \Xi \) of Example 7 is given by

\[
P(\Xi) = \{ \begin{array}{lcl}
a & \leftarrow & \text{not } b \\
d & \leftarrow & \text{not } c \\
c & \leftarrow & e, \text{not } b \\
b & \leftarrow & \text{not } a \\
e & \leftarrow & \text{not } d \\
\end{array} \}.
\]

An \( ADF \) \( \Xi \) and the corresponding \( NLP \) \( P(\Xi) \) are equivalent under various well-known semantics (Strass 2013). Indeed, the complete models, grounded models, preferred models and stable models of \( \Xi \) correspond respectively to the partial stable models, grounded models, regular models and stable models of \( P(\Xi) \). This result allows us to say \( ADFs \) are as expressive as \( NLPs \). From an \( NLP \) \( P(\Xi) \), we obtain an \( ADF \) \( \Xi(P) \) via Definition 28, and then again an \( NLP \) \( P(\Xi(P)) \) via Definition 34. Although \( P(\Xi) \) and \( P(\Xi(P)) \) are equivalent according to the aforementioned semantics, it is not guaranteed \( P = P(\Xi(P)) \):

Recall the \( NLP \) \( P(\Xi) \) in Example 12 and the corresponding \( ADF^+ \) \( \Xi(P) \) in Example 31. From \( \Xi(P) \) via Definition 34, we obtain the \( NLP \) \( P(\Xi(P)) \) (note \( P \neq P(\Xi(P)) \)):

\[
\begin{array}{lcl}
b & \leftarrow & \text{not } a \\
a & \leftarrow & \text{not } b \\
c & \leftarrow & p, \text{not } p \\
p & \leftarrow & \text{not } a \\
d & \leftarrow & .
\end{array}
\]

Similarly, from an \( ADF \) \( \Xi \), we can obtain the \( NLP \) \( P(\Xi) \) (Definition 34), and then again an \( ADF \) \( \Xi(P(\Xi)) \) (Definition 28). As above, they will be equivalent according to the aforementioned semantics, however, it does not guarantee \( \Xi = \Xi(P(\Xi)) \).

Recall the \( ADF \) \( \Xi \) in Example 7 and the corresponding \( NLP \) \( P(\Xi) \) in Example 35. From \( P(\Xi) \) via Definition 28, we obtain the \( ADF \) \( \Xi(P(\Xi)) \) (note \( \Xi \neq \Xi(P(\Xi)) \)):

\[
\begin{array}{lcl}
a[-b] & b[-a] & c[-b \land \neg d] \\
d[-c] & e[-d].
\end{array}
\]
5.2 From Logic Programming to ADF

As we have mentioned, previous attempts to identify a semantics for ADFs equivalent to a 3-valued semantics for NLPs have failed (Brewka and Woltran 2010; Strass 2013).

Definition 36 ((Brewka and Woltran 2010))

Let $P$ be an NLP over a set $A$ of atoms. Define an ADF, $Ξ_2(P) = (A, L, C^t)$, in which

- $L = \{(c, a) \mid a \leftarrow a_1, \ldots, a_m; \text{not } b_1, \ldots, \text{not } b_n \in P \text{ and } c \in \{a_1, \ldots, a_m, b_1, \ldots, b_n\}\}$;
- For each $a \in A$, $C^t_a = \{B \in \text{par}(a) \mid a \leftarrow a_1, \ldots, a_m; \text{not } b_1, \ldots, \text{not } b_n \in P, \{a_1, \ldots, a_m\} \subseteq B, \{b_1, \ldots, b_n\} \cap B = \emptyset\}$.

Alternatively, we could define the acceptance condition of each $a \in A$ as

$$\varphi_a \equiv \bigvee_{a \leftarrow a_1, \ldots, a_m; \text{not } b_1, \ldots, \text{not } b_n \in P} (a_1 \land \cdots \land a_m \land \neg b_1 \land \cdots \land \neg b_n). \quad (2)$$

As noted in (Strass 2013), by Definition 36, the NLPs, $P_1 = \{c \leftarrow b; a \leftarrow b; a \leftarrow c\}$ and $P_2 = \{c \leftarrow b; a \leftarrow \neg b; a \leftarrow b, \text{not } c; a \leftarrow c, \text{not } b; a \leftarrow b, c\}$ produce the same ADFs: $(Ξ_2(P_1) = Ξ_2(P_2))$. For any $s \in A$, its corresponding acceptance condition is $c[t] \rightarrow b \rightarrow b \rightarrow a[b \lor c]$. But the unique partial stable model (PSM) of $P_1$ is $\{a, c\}$, whereas $\{c\}$ is the unique PSM of $P_2$. Hence, this translation is inadequate to distinguish these two non-equivalent programs, according to PSMs. In contradistinction, our translation works accordingly and produces respectively the ADFs below, which has the same semantics as their corresponding original programs: $Ξ(P_1)$ is given by $c[t] \rightarrow b \rightarrow b \rightarrow a[b \lor c]$, and $Ξ(P_2)$ is given by $c[t] \rightarrow b \rightarrow b \rightarrow a[\neg(b \lor \neg c) \lor \neg b]$. However, when restricting to the class of NLPs where each rule is as $a \leftarrow \text{not } b_1, \ldots, \text{not } b_m$, $m \geq 0$, the translation of Definition 36 coincides with the translation of Definition 28 and is robust enough to capture 3-valued semantics as PSM and well-founded models.

Proposition 37

Let $P$ be an NLP, where each rule is either a fact or its body has only default literals as in $a \leftarrow \text{not } b_1, \ldots, \text{not } b_n$. Let $Ξ(P)$ be the ADF obtained from $P$ via Definition 28 and $Ξ_2(P)$ the ADF obtained from $P$ via Definition 36. Then $Ξ(P) = Ξ_2(P)$.

This result shows how Definition 36 could be employed to capture 3-valued semantics as PSMs: firstly, one could take an NLP $P$ and apply any program transformation (preserving PSMs) that transforms an NLP into one as that of Proposition 37. Then, one could apply the translation in Definition 36 to the resulting program (say $P'$) to obtain $Ξ_2(P')$. From Proposition 37, it holds $P$ and $Ξ_2(P')$ have the same PSMs.

5.3 On the connections between Logic Programming and Argumentation

Logic programming has long served as an inspiration for argumentation theory. Indeed, one can see the seminal work of Dung (Dung 1995) on Abstract Argumentation Frameworks (AAFs) as an abstraction of some aspects of logic programming. In (Caminada et al. 2015a), the authors pointed out that the translation from logic programming to these frameworks described in (Wu et al. 2009) is able to account for

---

3 By Equation 2, for $Ξ_2(P)$, we have $\varphi_a \equiv (b \land \neg c) \lor (\neg b \land c) \lor (b \land c) \equiv b \lor c$.

4 See (Brass and Dix 1995) for some program transformations.
the equivalences between Partial Stable Models, Well-Founded Models, Regular Models, Stable Models Semantics for NLPs and respectively Complete Models, grounded models, Preferred Models, Stable Models for AAFs. However, unlike we have done for ADFs, they have showed that, with their proposed translation from NLPs to AAFs, there cannot be a semantics for AAFs equivalent to L-Stable Semantics for NLPs.

When translating AAFs to NLPs, the connection between their semantics is stronger than when translating in the opposite direction as for any of the mentioned semantics for AAFs; there exists an equivalent semantics for NLPs (Caminada et al. 2015a).

In (Caminada and Schulz 2017), the authors showed how to translate Assumption-Based Argumentation (ABA) (Bondarenko et al. 1997; Dung et al. 2009; Toni 2014) to NLPs and how this translation can be re-applied for a reverse translation from NLPs to ABA. Curiously, the problematic direction here is from ABA to NLP. In (Caminada and Schulz 2017), they have showed that with their proposed translation, there cannot be a semantics for NLPs equivalent to the the semi-stable semantics (Caminada et al. 2015b; Schulz and Toni 2015) for ABA.

5.4 A Comparison between SETAF and ADF+

In (Nielsen and Parsons 2006), they proposed an extension of Dung’s Abstract Argumentation Frameworks (AAF) to allow joint attacks on arguments. The resulting framework, called SETAF, is displayed below:

**Definition 38 ((Nielsen and Parsons 2006))**

A framework with sets of attacking arguments (SETAF) is a pair $SF = (A, R)$, where $A$ is the set of arguments and $R \subseteq (2^A - \emptyset) \times A$ is the attack relation.

In an AAF, the unique relation between arguments is given by the attack relation, where an (individual) argument attacks another. In a SETAF (as well as in an ADF+), the novelty is that a set of arguments can attack an argument. For a translation from SETAF to ADF refer to (Polberg 2016):

**Translation.** Let $SF = (A, R)$ be a SETAF. The ADF corresponding to $SF$ is $DF^{SF} = (A, L, C)$, where $L = \{(x, y) \mid \exists X \subseteq A \text{ such that } x \in X \text{ and } (X, y) \in R\}$, $C = \{C_a\}, a \in A$ and every $C_a$ gets constructed in the following way: for every $B \subseteq \text{par}(a)$, if $\exists(X_i, a) \in R$ such that $X_i \subseteq B$, then $C_a(B) = f$; otherwise, $C_a(B) = t$.

The following result is immediate:

**Proposition 39**

Let $SF = (A, R)$ be a SETAF and $DF^{SF} = (A, L, C)$ be the corresponding ADF. Then, $DF^{SF}$ is an ADF+.

On the other hand, not every ADF+ will correspond to a SETAF according to the translation above. A noticeable difference between them is that for every argument $a \in A$ in a SETAF $SF = (A, R)$, it holds $(\emptyset, a) \not\in R$. Then, for every statement $s$ in the corresponding $DF^{SF}$, it holds $C_s(\emptyset) = t$, while $C_s(\emptyset) = f$ is allowed in ADF+. Indeed, when $C_s(\emptyset) = f$ in an ADF+, we have $C_s(R) = f$ for every $R \subseteq \text{par}(s)$, i.e., $\varphi_s \equiv f$. 
6 Conclusions and Future Works

In this paper, we have investigated the connections between Abstract Dialectical Frameworks (ADFs) and Normal Logic Programs (NLPs). Unlike previous works (Brewka and Woltran 2010; Strass 2013), we have provided a translation from NLPs to ADFs robust enough to capture the equivalence between several frameworks for these formalisms, including 3-valued semantics. In particular, after resorting to our translation, we have proved the equivalence between partial stable models, well-founded models, regular models, stable models semantics for NLPs and respectively complete models, grounded models, preferred models, stable models for ADFs.

Curiously, we have obtained these equivalence results by translating an NLP into a fragment of ADF, called Attacking Dialectical Frameworks ($ADF^+$), in which the unique relation involving statements is the attack relation. A distinguishing aspect of our translation when compared with related works as (Caminada et al. 2015a; Strass 2013) is that it is made in two steps: in the first step each NLP rule is translated into a substate- ment, and then, substatements corresponding to rules with the same head are gathered to constitute a unique statement. With this procedure, our intention is to simulate the semantics for NLPs, where the truth-value of an atom $b$ is the disjunction of the truth-values of the bodies of the rules whose head is $b$. Besides, we have defined a new semantics for $ADF^+$, called $L$-stable, and showed it is equivalent to the $L$-stable semantics (defined in (Eiter et al. 1997)) for NLPs.

An essential element to define these semantics for ADF is $\Gamma_D$, a kind of immediate consequences operator. When restricted to $ADF^+$, we have proved $\Gamma_D$ is equivalent to a noticeably simpler version. Indeed, owing to this simplicity, verifying whether a given labelling is complete is of complexity $P$, whereas this verification problem is DP-complete for ADF (Brewka et al. 2013). This is a promising result as it might also mean the complexity of many reasoning tasks on $ADF^+$s may have the same complexity as standard Dung’s Abstract Argumentation Frameworks (Dung 1995).

In a future work, we intend to complete a thorough investigation on the connections between ADFs and $ADF^+$s. Regarding the observed equivalences between NLP and $ADF^+$, one can claim that $ADF^+$s are as general as ADFs, and that the attack relation suffices to express these relations concerning statements in ADFs. Given the results unveiled in the current paper, we also envisage unfolding semantic connections between NLPs and SETAFs (Nielsen and Parsons 2006), an extension of Dung’s Abstract Argumentation Frameworks that allows joint attacks on arguments. We expect to find several correspondences between their semantics.

Supplementary material

To view supplementary material for this article, please visit https://doi.org/10.1017/S1471068419000280.

References


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


