Well-Founded Argumentation Semantics for Extended Logic Programming

Ralf Schweimeier and Michael Schroeder
Department of Computing
City University, London, UK
{ralf,msch}@soi.city.ac.uk

Abstract

This paper defines an argumentation semantics for extended logic programming and shows its equivalence to the well-founded semantics with explicit negation. We set up a general framework in which we extensively compare this semantics to other argumentation semantics, including those of Dung, and Prakken and Sartor. We present a general dialectical proof theory for these argumentation semantics.

1 Introduction

Argumentation has attracted much interest in the area of AI. On the one hand, argumentation is an important way of human interaction and reasoning, and is therefore of interest for research into intelligent systems. Application areas include automated negotiation via argumentation [15, 14, 18] and legal reasoning [17]. On the other hand, argumentation provides a formal model for various assumption based (or non-monotonic, or default) reasoning formalisms [4, 5]. In particular, various argumentation based semantics have been proposed for logic programming with default negation [4, 7].

Argumentation semantics are elegant since they can be captured in an abstract framework [7, 4, 20, 12], for which an elegant theory of attack, defence, acceptability, and other notions can be developed, without recourse to the concrete instance of the reasoning formalism at hand. This framework can then be instantiated to various assumption based reasoning formalisms. Similarly, a dialectical proof theory, based on dialogue trees, can be defined for an abstract argumentation framework, and then applied to any instance of such a framework [7, 11].

In general, an argument A is a proof which may use a set of defeasible assumptions. Another argument B may have a conclusion which contradicts the assumptions or the conclusions of A, and thereby B *attacks* A. There are two fundamental notions of such attacks: undercut and rebut [17] or equivalently *ground-attack* and *reductio-ad-absurdum attack* [6]. We will use the terminology of undercuts and rebuts. Both attacks differ in that an undercut attacks a premise of an argument, while a rebut attacks a conclusion.

Given a logic program we can define an argumentation semantics by iteratively collecting those arguments which are acceptable to a proponent, i.e. they can be defended against all opponent attacks. In fact, such a notion of acceptability can be defined in a number of ways depending on which attacks we allow the proponent and opponent to use.

Normal logic programs do not have negative conclusions, which means that we cannot use rebuts. Thus both opponents can only launch undercuts on each other's assumptions. Various argumentation semantics have been defined for normal logic programs [4, 7, 13], some of which are equivalent to existing semantics such as the stable model semantics [9] or the well-founded semantics [8].

Extended logic programs [10, 2, 21], on the other hand, introduce explicit negation, which states that a literal is explicitly false. As a result, both undercuts and rebuts are possible forms of attack; there are further variations depending on whether any kind of counter-attack is admitted. A variety of argumentation semantics arise if one allows one notion of attack as defence for the proponent, and another as attack for the opponent. Various argumentation semantics have been proposed for extended logic programs [6, 17]. Dung has shown that a certain argumentation semantics is equivalent to the answer set semantics [10], a generalisation of the stable model semantics [9]. To our knowledge, no argumentation semantics has yet been found equivalent to the well-founded semantics for extended logic programs, WFSX [16, 2].

This paper makes the following contributions: we define a least fixpoint argumentation semantics for extended logic programs, and show its equivalence to the well-founded semantics with explicit negation [16, 2, 1]. In order to relate this semantics to other argumentation semantics, we set up a general framework to classify notions of justified arguments, and use it to compare our argumentation semantics to those of Dung [6] and Prakken and Sartor [17] among others. We develop a general dialectical proof theory for the notions of justified arguments we introduce.

The paper is organised as follows: First we define arguments and notions of attack and acceptability. Then we set up a framework for classifying different least fixpoint argumentation semantics, based on different notions of attack. In Section 4, we recall the definition of WFSX, and in Section 5, we prove the equivalence of an argumentation semantics and WFSX. A general dialectical proof theory for arguments is presented in Section 6, and its soundness and completeness is proven.

2 Extended Logic Programming and Argumentation

We summarise the definitions of arguments for extended logic programs, and define various notions of attack between arguments.

2.1 Arguments

Definition 1 *An* objective literal *is an atom A or its explicit negation $\neg A$. We define $\neg\neg L = L$. A* default literal *is of the form not L where L is an objective literal. A* literal *is either an objective or a default literal. An* extended logic program *is a (possibly infinite) set of rules of the form*

$$L_0 \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_{m+n}$$

where $m, n \geq 0$, and each L_i is an objective literal $(0 \leq i \leq m+n)$.
For such a rule r, we call L_0 the head *of the rule, $head(r)$, and $L_1, \ldots, not\ L_{m+n}$ the* body *of the rule, $body(r)$.*

Our definition of an argument for an extended logic program is based on [17]. Essentially, an argument is a partial proof, resting on a number of *assumptions*, i.e. a set of default literals.[1]

[1] In [4, 6], an argument *is* a set of assumptions; the two approaches are equivalent in that there is an argument with a conclusion L iff there is a set of assumptions from which L can be inferred. See the discussion in [17].

Note that we do not consider priorities of arguments, as used e.g. in [17, 20].

Definition 2 *Let P be an extended logic program.*
An argument *for P is a finite sequence $A = [r_1, \ldots r_n]$ of rules $r_i \in P$ such that for every $1 \leq i \leq n$, for every objective literal L_j in the body of r_i there is a $k > i$ such that $head(r_k) = L_j$.*
A subargument *of A is a subsequence of A which is an argument. The head of a rule in A is called a* conclusion *of A, and a default literal not L in the body of a rule of A is called an* assumption *of A. We write $assm(A)$ for the set of assumptions and $conc(A)$ for the set of conclusions of an argument A.*
An argument A with a conclusion L is a minimal argument for *L if there is no subargument of A with conclusion L. An argument is* minimal *if it minimal for some literal L. Given an extended logic program P, we denote the set of minimal arguments for P by $Args_P$.*

The restriction to minimal arguments is not essential, but convenient, since it rules out arguments constructed from several unrelated arguments. Generally, one is interested in the conclusions of an argument, and wants to avoid having rules in an argument which do not contribute to the desired conclusion.

2.2 Notions of Attack

There are two fundamental notions of attack: *undercut*, which invalidates an assumption of an argument, and *rebut*, which contradicts a conclusion of an argument [6, 17]. From these, we may define further notions of attack, by allowing either of the two fundamental kinds of attack, and considering whether any kind of counter-attack is allowed or not. We will now formally define these notions of attacks.

Definition 3 *Let A_1 and A_2 be arguments.*

1. *A_1* undercuts *A_2 if there is an objective literal L such that L is a conclusion of A_1 and not L is an assumption of A_2.*

2. *A_1* rebuts *A_2 if there is an objective literal L such that L is a conclusion of A_1 and $\neg L$ is a conclusion of A_2.*

3. *A_1* attacks *A_2 if A_1 undercuts or rebuts A_2.*

4. *A_1* defeats *A_2 if A_1 undercuts A_2, or (A_1 rebuts A_2 and A_2 does not undercut A_1).*

5. A_1 strongly attacks A_2 *if* A_1 *attacks* A_2 *and* A_2 *does not undercut* A_1.

6. A_1 strongly undercuts A_2 *if* A_1 *undercuts* A_2 *and* A_2 *does not undercut* A_1.

The notions of *undercut* and *rebut*, and hence *attack* are fundamental for extended logic programs [6, 17]. The notion of *defeat* is used in [17], along with a notion of *strict defeat*, i.e. a defeat that is not counter-defeated. For arguments without priorities, rebuts are symmetrical, and therefore strict defeat coincides with strict undercut, i.e. an undercut that is not counter-undercut. Similarly, strict attack coincides with strict undercut. For this reason, we use the term *strong undercut* instead of *strict undercut*, and similarly define *strong attack* to be an attack which is not counter-undercut. We will use the following abbreviations for these notions of attack. r for rebuts, u for undercuts, a for attacks, d for defeats, sa for strongly attacks, and su for strongly undercuts.

These notions of attack define for any extended logic program a binary relation on the set of arguments of that program.

Definition 4 *A* notion of attack *is a function x which assigns to each extended logic program P a binary relation x_P on the set of arguments of P, i.e. $x_P \subseteq Args_P^2$. Notions of attack can be partially ordered by defining $x \subseteq y$ iff $\forall P : x_P \subseteq y_P$*

Definition 5 *Let x be a notion of attack. Then the* inverse *of x, denoted by x^{-1}, is defined as $x_P^{-1} = \{(B, A) \mid (A, B) \in x_P\}$.*

In this relational notation, Definition 3 can be rewritten as $a = u \cup r$, $d = u \cup (r - u^{-1})$, $sa = (u \cup r) - u^{-1}$, and $su = u - u^{-1}$. Using the set-theoretic laws $A - B \subseteq A \subseteq A \cup C$ and $(A \cup B) - C = (A - C) \cup (B - C)$ (for all sets A, B, and C), it is easy to see that the notions of attack of Definition 3 are partially ordered according to the following Hasse diagram.

$$\begin{array}{ccc} & \textit{attacks} = a = u \cup r & \\ & | & \\ & \textit{defeats} = d = u \cup (r - u^{-1}) & \\ \diagup & & \diagdown \\ \textit{undercuts} = u & & \textit{strongly attacks} = sa = (u \cup r) - u^{-1} \\ \diagdown & & \diagup \\ & \textit{strongly undercuts} = su = u - u^{-1} & \end{array}$$

This diagram contains the notions of attack used in [6, 17], plus *strongly attacks* which seemed a natural intermediate notion between *strongly undercuts* and *defeats*. We have not included *rebuts*, because in the absence of priorities, *rebuts* is somewhat weaker than *undercuts*, because it is symmetric: a rebut is always counter-rebutted, while the same does not hold for *undercuts*.

2.3 Acceptability and Justified Arguments

Given the above notions of attack, we define acceptability of an argument. Basically, an argument is acceptable if it can be defended against any attack. Depending on which particular notion of attack we use as defence and which for the opponent's attacks, we obtain a host of acceptability notions.

Acceptability forms the basis for our argumentation semantics, which is defined as the least fixpoint of a function, which collects all acceptable arguments. The *least* fixpoint is of particular interest [17, 6], because it provides a canonical fixpoint semantics and it can be constructed inductively.

Definition 6 *Let x and y be notions of attack. Let A be an argument, and S a set of arguments. Then A is x/y-*acceptable wrt. *S if for every argument B such that $(B, A) \in x$ there exists an argument $C \in S$ such that $(C, B) \in y$.*

Based on the notion of acceptability, we can then define a fixpoint semantics for arguments.

Definition 7 *Let x and y be notions of attack, and P an extended logic program. The operator $F_{P,x/y} : \mathcal{P}(Args_P) \to \mathcal{P}(Args_P)$ is defined as*

$$F_{P,x/y}(S) = \{A \mid A \textit{ is } x/y\textit{-acceptable wrt. } S\}$$

We denote the least fixpoint of $F_{P,x/y}$ by $J_{P,x/y}$. If the program P is clear from the context, we omit the subscript P. An argument A is called x/y-justified if $A \in J_{x/y}$; an argument is called x/y-overruled if it is attacked by an x/y-justified argument; and an argument is called x/y-defensible if it is neither x/y-justified nor x/y-overruled.

For any program P, the least fixpoint exists by the Knaster-Tarski fixpoint theorem [19, 3], because $F_{P,x/y}$ is monotone. It can be constructed by transfinite induction as follows:

$$\begin{array}{lll} J^0_{x/y} & = \emptyset & \\ J^{\alpha+1}_{x/y} & = F_{P,x/y}(J^{\alpha}_{x/y}) & \text{for } \alpha+1 \text{ a successor ordinal} \\ J^{\lambda}_{x/y} & = \bigcup_{\alpha<\lambda} J^{\alpha}_{x/y} & \text{for } \lambda \text{ a limit ordinal} \end{array}$$

Then there exists a least ordinal λ_0 such that $F_{x/y}(J^{\lambda_0}_{x/y}) = J^{\lambda_0}_{x/y} = J_{x/y}$.

3 Relationships of Notions of Justifiability

This section is devoted to an analysis of the relationship between the different notions of justifiability, leading to a hierarchy of notions of justifiability illustrated in Figure 2.

First of all, it is easy to see that the least fixpoint increases if we weaken the attacks, or strengthen the defence.

Proposition 1 *Let $x' \subseteq x$, $y \subseteq y'$ be notions of attack, then $J_{x/y} \subseteq J_{x'/y'}$.*

Theorem 2 states that it does not make a difference if we allow only the strong version of the defence. This is because an argument need not defend itself on its own, but it may rely on other arguments to defend it.

We only give a formal proof for the first theorem; the proofs for the other theorems are similar, and we provide an intuitive informal explanation instead.

Theorem 2 *Let x and y be notions of attack such that $x \supseteq$ undercuts, and let $sy = y -$ undercuts^{-1}. Then $J_{x/y} = J_{x/sy}$.*

Proof. Informally, every x-attack B to an x/y-justified argument A is y-defended by some x/sy-justified argument C (by induction). Now if C was *not* a sy-attack, then it is undercut by B, and because $x \supseteq$ *undercuts* and C is justified, there exists a *strong* defence for C against B, which is also a defence of the original argument A against C.

The formal proof is by transfinite induction. By Proposition 1, we have $J_{x/sy} \subseteq J_{x/y}$. We prove the inverse inclusion by showing that for all ordinals α: $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x/sy}$, by transfinite induction on α.

Base case $\alpha = 0$: $J_{x/y} = \emptyset = J_{x/sy}$.

Successor ordinal $\alpha \rightsquigarrow \alpha + 1$: Let $A \in J^{\alpha+1}_{x/y}$, and $(B, A) \in x$. By definition, there exists $C \in J^{\alpha}_{x/y}$ such that $(C, B) \in y$. By induction hypothesis, $C \in J^{\alpha}_{x/sy}$.

If B does not undercut C, then we are done. If, however, B undercuts C, then because $C \in J^{\alpha}_{x/sy}$, and *undercuts* $\subseteq x$, there exists $D \in J^{\alpha_0}_{x/sy}(\emptyset)(\alpha_0 < \alpha)$ such that $(D, B) \in sy$. It follows that $A \in J^{\alpha+1}_{x/sy}$.

Limit ordinal λ*:* Assume $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x/sy}$ for all $\alpha < \lambda$. Then $J^{\lambda}_{x/y} = \bigcup_{\alpha<\lambda} J^{\alpha}_{x/y} \subseteq \bigcup_{\alpha<\lambda} J^{\alpha}_{x/sy} = J^{\lambda}_{x/sy}$ □

In particular, the previous Theorem states that undercut and strong undercut are equivalent as a defence, as are attack and strong attack. This may be useful in an implementation, where we may use the stronger notion of defence without changing the semantics, thereby decreasing the number of arguments to be checked. The following Corollary shows that because defeat lies between attack and strong attack, it is equivalent to both as a defence.

Corollary 3 *Let x be a notion of attack such that $x \supseteq$ undercuts. Then $J_{x/a} = J_{x/d} = J_{x/sa}$.*

Proof. With Proposition 1 and Theorem 2, we have $J_{x/a} \subseteq J_{x/d} \subseteq J_{x/sa} = J_{x/a}$. □

Theorem 4 *Let x be a notion of attack such that $x \supseteq$ strongly attacks. Then $J_{x/u} = J_{x/d} = J_{x/a}$.*

Proof. Every x-attack B to a x/a-justified argument a is attacked by some x/u-justified argument C (by induction). If C is a rebut, but not an undercut, then because B strongly attacks C, and because $x \supseteq$ *strongly attacks*, there must have been an argument defending C by undercutting B, thereby also defending A against B.

The statement for *defeats* follows in a similar way to Corollary 3.

□

Theorem 5 $J_{sa/su} = J_{sa/sa}$

The proof is similar to Theorem 4.

Theorem 6 $J_{su/a} = J_{su/d}$

Proof. Every strong undercut B to a su/a-justified argument A is attacked by some su/d-justified argument C (by induction). If C does not defeat A, then there is some argument D defending C by defeating B, thereby also defending A against B. □

We will now present some example programs which distinguish various notions of justifiability.

Example 1 *Consider P_1 in Figure 1. For any notion of attack x, we have $J_{su/x} = J_{sa/x} = \{[p \leftarrow not\ q], [q \leftarrow not\ p]\}$, because there is no strong undercut or strong attack to any of the arguments. However, $J_{a/x} = J_{d/x} = J_{u/x} = \emptyset$, because every argument is undercut (and therefore defeated and attacked).*

Example 2 *Consider P_2 in Figure 1. Let x be a notion of attack. Then $J_{d/x} = J_{a/x} = \emptyset$, because every argument is defeated (hence attacked).*

$P_1 =$ $p \leftarrow not\ q$ $q \leftarrow not\ p$	$P_2 =$ $p \leftarrow not\ q$ $q \leftarrow not\ p$ $\neg p$	$P_3 =$ $p \leftarrow not\ q$ $q \leftarrow not\ r$ $r \leftarrow not\ s$ $s \leftarrow not\ p$ $\neg p$
$P_4 =$ $p \leftarrow not\ q$ $q \leftarrow not\ p$ $r \leftarrow not\ p$	$P_5 =$ $p \leftarrow not\ \neg p$ $\neg p$	$P_6 =$ $\neg p \leftarrow not\ q$ $\neg q \leftarrow not\ p$ p q

Figure 1: Examples

$J_{sa/su} = J_{sa/sa} = \{[q \leftarrow not\ p]\}$, because $[q \leftarrow not\ p]$ is the only argument which is not strongly attacked, but it does not strongly attack any other argument. $J_{u/su} = J_{u/u} = \{[\neg p]\}$, because there is no undercut to $[\neg p]$, but $[\neg p]$ does not undercut any other argument. $J_{u/a} = \{[\neg p], [q \leftarrow not\ p]\}$, because there is no undercut to $[\neg p]$, and the undercut $[p \leftarrow not\ p]$ to $[q \leftarrow not\ p]$ is attacked by $[\neg p]$. We also have $J_{sa/u} = \{[\neg p], [q \leftarrow not\ p]\}$, because $[q \leftarrow not\ p]$ is not strongly attacked, and the strong attack $[p \leftarrow not\ q]$ on $[\neg p]$ is undercut by $[q \leftarrow not\ p]$.

Example 3 *Consider P_3 in Figure 1. Let x be a notion of attack. Then $J_{sa/x} = \emptyset$, because every argument is strongly attacked.*

$J_{su/u} = J_{su/su} = \{[\neg p]\}$, because all arguments except $[\neg p]$ are strongly undercut, but $[\neg p]$ does not undercut any argument. And $J_{u/a} = J_{su/sa} = J_{su/a} = \{[\neg p], [q \leftarrow not\ r], [s \leftarrow not\ p]\}$.

Example 4 *Consider P_4 in Figure 1. Let x be a notion of attack. Then $J_{u/x} = J_{d/x} = J_{a/x} = \emptyset$, because every argument is undercut. $J_{su/su} = J_{su/sa} = J_{sa/su} = J_{sa/sa} = \{[p \leftarrow not\ q], [q \leftarrow not\ p]\}$ In this case, the strong attacks are precisely the strong undercuts; The argument $[r \leftarrow not\ p]$ is not justified, because the strong undercut $[p \leftarrow not\ q]$ is undercut, but not strongly undercut, by $[q \leftarrow not\ p]$. $J_{su/u} = J_{su/a} = J_{sa/u} = J_{sa/a} = \{[p \leftarrow not\ q], [q \leftarrow not\ p], [r \leftarrow not\ p]\}$ Again, undercuts and attacks, and strong undercuts and strong attacks, coincide; but now $[r \leftarrow not\ p]$ is justified, because non-strong undercuts are allowed as defence.*

Example 5 *Consider P_5 in Figure 1. Then $J_{a/x} = \emptyset$, because both arguments attack each other, while $J_{d/x} = \{[\neg p]\}$, because $[\neg p]$ defeats $[p \leftarrow not\ \neg p]$, but not vice versa.*

Example 6 *Consider P_6 in Figure 1. Let x be a notion of attack. Then $J_{sa/x} = J_{d/x} = J_{a/x} = \emptyset$, because every argument is strongly attacked (hence defeated and attacked), while $J_{u/x} = J_{su/x} = \{[p], [q]\}$.*

Theorem 7 *The notions of justifiability are ordered (by set inclusion) according to the Hasse diagram in Figure 2.*

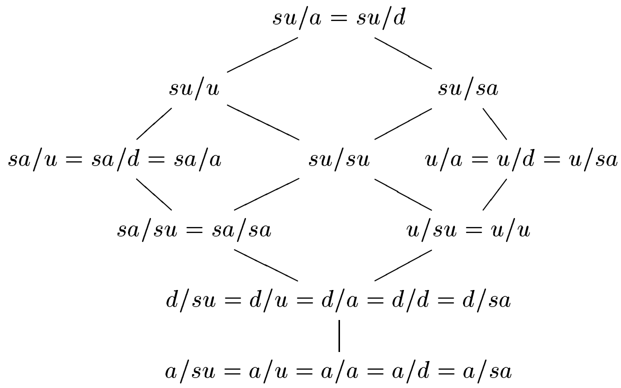

Figure 2: Hierarchy of Notions of Justifiability

By definition, Dung's grounded argumentation semantics [6] is exactly a/u-justifiability, while Prakken and Sartor's semantics [17], if we disregard priorities, amounts to d/su-justifiability. As corollaries to Theorem 7, we obtain relationships of these semantics to the other notions of justifiability.

Corollary 8 *Let J_{Dung} be the set of justified arguments according to Dung's grounded argumentation semantics [6]. Then $J_{Dung} = J_{a/su} = J_{a/u} = J_{a/a} = J_{a/d} = J_{a/sa}$ and $J_{Dung} \subseteq J_{x/y}$ for all notions of attack x and y.*

Corollary 9 *Let J_{PS} be the set of justified arguments according to Prakken and Sartor's argumentation semantics [17], where all arguments have the same priority. Then $J_{PS} = J_{d/su} = J_{d/u} = J_{d/a} = J_{d/d} = J_{d/sa}$, $J_{PS} \subseteq J_{x/y}$ for all notions of attack $x \neq a$ and y, and $J_{PS} \supseteq J_{a/y}$ for all notions of attack y.*

Remark 1 *1. The notions of a/x-, d/x- and sa/x-justifiability are very sceptical in that a* fact *p may not be justified, if there is a rule $\neg p \leftarrow B$ (where $not\ p \notin B$) that is not x-attacked. On the other hand this is useful in terms of avoiding inconsistency.*

2. sx/y-justifiability is very credulous, because it does not take into account non-strong attacks, so e.g. the program $\{p \leftarrow not\ q, q \leftarrow not\ p\}$ has the justified arguments $[p \leftarrow not\ q]$ and $[q \leftarrow not\ p]$.

Remark 2 *One might ask whether any of the semantics in Figure 2 are equivalent for* non-contradictory *programs, i.e. programs for which there is no literal L such that there exist justified arguments for both L and $\neg L$. The answer to this question is no: all the examples above distinguishing different notions of justifiability involve only non-contradictory programs.*

In particular, even for non-contradictory programs, Dung's and Prakken and Sartor's semantics differ, and both differ from u/a-justifiability, which will be shown equivalent to the well-founded semantics WFSX [16, 2] in the following section.

4 Well-founded semantics

We recollect the definition of the well-founded semantics for extended logic programs, WFSX. We use the definition of [1], because it is closer to our definition of argumentation semantics than the original definition of [16, 2].

Definition 8 *The set of all objective literals of a program P is called the* Herbrand base *of P and denoted by $\mathcal{H}(P)$. A* pseudo-interpretation *of a program P is a set $T \cup not\ F$ where T and F are subsets of $\mathcal{H}(P)$. An* interpretation *is a pseudo-interpretation where the sets T and F are disjoint. An interpretation is called* two-valued *if $T \cup F = \mathcal{H}(P)$.*

Definition 9 *Let P be an extended logic program, I an interpretation, and let P' (resp. I') be obtained from P (resp. I) by replacing every literal $\neg A$ by a new atom, say $\neg\_A$. The GL-transformation $\frac{P'}{I'}$ is the program obtained from P' by removing all rules containing a default literal not A such that $A \in I'$, and then removing all remaining default literals from P', obtaining a definite program P''. Let J be the least model of P''. $\Gamma_P I$ is obtained from J by replacing the introduced atoms $\neg\_A$ by $\neg A$.*

Definition 10 *The* semi-normal *version of a program P is the program P_s obtained from P by replacing every rule $L \leftarrow Body$ in P by the rule $L \leftarrow not\ \neg L, Body$.*

If the program P is clear from the context, we write ΓI for $\Gamma_P I$ and $\Gamma_s I$ for $\Gamma_{P_s} I$.

Definition 11 *Let P be a program whose least fixpoint of $\Gamma\Gamma_s$ is T. Then the* paraconsistent well-founded model of P *is the pseudo-interpretation $WFM_p(P) = T \cup not\ (\mathcal{H}(P) - \Gamma_s T)$. If $WFM_p(P)$ is an interpretation, then P is called* non-contradictory, *and $WFM_p(P)$ is the* well-founded model of P, *denoted by $WFM(P)$.*

The paraconsistent well-founded model can by defined iteratively by the transfinite sequence $\{I_\alpha\}$:

$$\begin{array}{lcll} I_0 & = & \emptyset & \\ I_{\alpha+1} & = & \Gamma\Gamma_s I_\alpha & \text{for successor ordinal } \alpha+1 \\ I_\lambda & = & \bigcup_{\alpha<\lambda} I_\alpha & \text{for limit ordinal } \lambda \end{array}$$

There exists a smallest ordinal λ_0 such that I_{λ_0} is the least fixpoint of $\Gamma\Gamma_s$, and $WFM_p(P) = I_{\lambda_0} \cup not\ (\mathcal{H}(P) - \Gamma_s I_{\lambda_0})$.

5 Equivalence of argumentation semantics and WFSX

In this section, we will show that the argumentation semantics $J_{u/a}$ and the well-founded model coincide. That is, the conclusions of justified arguments are exactly the objective literals which are true in the well-founded model; and those objective literals all of whose arguments are overruled are exactly the literals which are false in the well-founded model. The result holds also for contradictory programs under the *paraconsistent* well-founded semantics. This is important, because it shows that contradictions in the argumentation semantics are precisely the contradictions under the well-founded semantics, and allows the application of contradiction removal (or avoidance) methods to the argumentation semantics. Because for non-contradictory programs, the well-founded semantics coincides with the paraconsistent well-founded semantics [1], we obtain as a corollary that argumentation semantics and well-founded semantics coincide for non-contradictory programs.

In order to compare the argumentation semantics with the well-founded semantics, we define the set of literals which are a consequence of the argumentation semantics.

Definition 12 *$A(P) = T\ \cup\ not\ F$, where*
$T = \{L \mid$ there is a justified argument for $L\}$ and
$F = \{L \mid$ all arguments for L are overruled $\}$.

The following Proposition shows a precise connection between arguments and consequences of a program $\frac{P}{I}$.

Proposition 10 *Let I be a two-valued interpretation.*

1. *$L \in \Gamma(I)$ iff $\exists$ argument A with conclusion L such that $assm(A) \subseteq I$.*
2. *$L \in \Gamma_s(I)$ iff $\exists$ argument A with conclusion L such that $assm(A) \subseteq I$ and $\neg conc(A) \cap I = \emptyset$.*
3. *$L \notin \Gamma(I)$ iff $\forall$ arguments A with conclusion L, $assm(A) \cap I \neq \emptyset$.*
4. *$L \notin \Gamma_s(I)$ iff $\forall$ arguments A with conclusion L, $assm(A) \cap I \neq \emptyset$ or $\neg conc(A) \cap I \neq \emptyset$.*

Proof. See Appendix. □

Theorem 11 *Let P be an extended logic program. Then $WFM_p(P) = A(P)$.*

Proof. First, note that A undercuts B iff $\exists$ L s.t. *not* $L \in assm(A)$ and $L \in conc(B)$; and A rebuts B iff $\exists$ $L \in conc(A) \cap \neg conc(B)$.

We show that for all ordinals α, $I_\alpha = A_\alpha$, by transfinite induction on α.

Base case $\alpha = 0$: $I_\alpha = \emptyset = A_\alpha$

Successor ordinal $\alpha \rightsquigarrow \alpha + 1$:

$L \in I_{\alpha+1}$

iff (Def. of $I_{\alpha+1}$)

$L \in \Gamma\Gamma_s I_\alpha$

iff (Prop. 10(1))

$\exists$ argument A for L such that $assm(A) \subseteq \Gamma_s I_\alpha$

iff (Def. of $\subseteq$, and $\Gamma_s I_\alpha$ is two-valued)

$\exists$ argument A for L such that
$\forall$ *not* $L \in assm(A), L \notin \Gamma_s I_\alpha$

iff (Prop. 10(4))

$\exists$ argument A for L such that $\forall$ *not* $L \in assm(A)$, for any argument B for L,
($\exists$ *not* $L' \in assm(B)$ *s.t.* $L' \in I_\alpha$ or
$\exists$ $L'' \in conc(B)$ *s.t.* $\neg L'' \in I_\alpha$)

iff (Induction hypothesis)

$\exists$ argument A for L such that $\forall$ *not* $L \in assm(A)$, for any argument B for L, ($\exists$ *not* $L' \in assm(B)$ *s.t.* $\exists$ argument $C \in J_\alpha$ for L', or
$\exists$ $L'' \in conc(B)$ *s.t.* $\exists$ argument $C \in J_\alpha$ for $\neg L''$)

iff (Def. of undercut and rebut)

$\exists$ argument A for L such that for any undercut B to A, ($\exists$ argument $C \in J_\alpha$ s.t. C undercuts B, or
$\exists$ argument $C \in J_\alpha$ s.t. C rebuts B)

iff

$\exists$ argument A for L such that for any undercut B to A, $\exists$ argument $C \in J_\alpha$ s.t. C attacks B

iff (Def. of $J_{\alpha+1}$)

$\exists$ argument $A \in J_{\alpha+1}$ for L

iff (Def. of $A_{\alpha+1}$)

$L \in A_{\alpha+1}$

Limit ordinal λ:
$I_\lambda = \bigcup_{\alpha<\lambda} I_\alpha$ and $A_\lambda = \bigcup_{\alpha<\lambda} A_\alpha$, so by induction hypothesis ($I_\alpha = A_\alpha$ for all $\alpha < \lambda$), $I_\lambda = A_\lambda$.

Now, we show that a literal *not* L is in the well-founded semantics iff every argument for L is overruled.

not $L \in WFM_p(P)$

iff (Def. of $WFM_p(P)$)

$L \notin \Gamma_s I$

iff (Prop. 10(4))

for all arguments A for L,
($\exists$ *not* $L' \in assm(A)$ *s.t.* $L' \in I$, or
$\exists$ $L'' \in conc(A)$ *s.t.* $\neg L'' \in I$)

iff (I=A)

for all arguments A for L,
($\exists$ *not* $L' \in assm(A)$ *s.t.* $\exists$ argument $B \in J$ for L',
or $\exists$ $L'' \in conc(A)$ *s.t.* $\exists$ argument $B \in J$ for $\neg L''$)

iff (Def. of undercut and rebut)

for all arguments A for L, ($\exists$ argument $B \in J$ s.t. B undercuts A, or $\exists$ argument $B \in J$ s.t. B rebuts A)

iff

every argument for L is attacked by a justified argument in J

iff (Def. of overruled)

every argument for L is overruled

iff (Def. of $A(P)$)

not $L \in A(P)$ □

Corollary 12 *Let P be a non-contradictory program. Then $WFM(P) = A(P)$.*

Remark 3 *In a similar way, one can show that the Γ operator corresponds to undercuts, while the Γ_s operator corresponds to attacks, and so the least fixpoints of $\Gamma\Gamma$, $\Gamma_s\Gamma$, and $\Gamma_s\Gamma_s$ correspond to $J_{u/u}$, $J_{a/u}$, and $J_{a/a}$, respectively. In [1], the least fixpoints of these operators are shown to be ordered as $lfp(\Gamma_s\Gamma) \subseteq lfp(\Gamma_s\Gamma_s) \subseteq lfp(\Gamma\Gamma_s)$, and $lfp(\Gamma_s\Gamma) \subseteq lfp(\Gamma\Gamma) \subseteq lfp(\Gamma\Gamma_s)$. Because $J_{a/u} = J_{a/a} \subseteq J_{u/u} \subseteq J_{u/a}$ by Theorem 7, we can strengthen this statement to $lfp(\Gamma_s\Gamma) = lfp(\Gamma_s\Gamma_s) \subseteq lfp(\Gamma\Gamma) \subseteq lfp(\Gamma\Gamma_s)$.*

6 Proof theory

One of the benefits of relating the argumentation semantics $J_{u/a}$ to WFSX is the existence of an efficient top-down proof procedure for WFSX [1], which we can use to compute justified arguments in $J_{u/a}$. On the other hand, *dialectical* proof theories, based on dialogue trees, have been defined for a variety of argumentation semantics [17, 11, 13]. In this section, we present a sound and complete dialectical proof theory for the least fixpoint argumentation semantics $J_{x/y}$ for any notions of attack x and y. Our presentation closely follows [17]. As a further consequence, we obtain an equivalence of the proof theory for WFSX and the dialectical proof theory for arguments.

Definition 13 *An x/y-*dialogue *is a finite nonempty sequence of moves $move_i = (Player_i, Arg_i) (i > 0)$, such that*

1. *$Player_i = P$ iff i is odd; and $Player_i = O$ iff i is even.*

2. *If $Player_i = Player_j$ and $i \neq j$, then $Arg_i \neq Arg_j$.*

3. *If $Player_i = P$ and $i > 1$, then Arg_i is a minimal argument such that $(Arg_i, Arg_{i-1}) \in y$.*

4. *If $Player_i = O$, then $(Arg_i, Arg_{i-1}) \in x$.*

The first condition states that the players P (Proponent) and O (Opponent) take turns, and P starts. The second condition prevents the proponent from repeating a move. The third and fourth conditions state that both players have to attack the other player's last move, where the opponent is allowed to use the notion of attack x, while the proponent may use y to defend its arguments.

Definition 14 *An* x/y-dialogue tree *is a tree of moves such that every branch is a dialogue, and for all moves $move_i = (P, Arg_i)$, the children of $move_i$ are all those moves (O, Arg_j) such that $(Arg_j, Arg_i) \in x$.*

Definition 15 *A player* wins an x/y-dialogue *iff the other player cannot move. A player* wins an x/y-dialogue tree *iff it wins all branches of the tree. An x/y-dialogue tree which is won by the proponent is called a* winning x/y-dialogue tree. *An argument A is* provably x/y-justified *iff there exists a x/y-tree with A as its root, and won by the proponent. A literal L is a* provably justified conclusion *iff it is a conclusion of a provably x/y-justified argument. The* height *of a dialogue tree is 0 if it consists only of the root, and otherwise $height(t) = \bigcup height(t_i) + 1$ where t_i are the trees rooted at the grandchildren of t.*

We show that the proof theory of x/y-dialogue trees is sound and complete for any notions of attack x and y.

Theorem 13 *An argument is provably x/y-justified iff it is x/y-justified.*

Proof. "If"-direction. We show by transfinite induction: If $A \in J^{\alpha}_{x/y}$, then there exists a winning x/y-dialogue tree of height $< \alpha$ for A.

Base case $\alpha = 0$:
Then there exists no argument B such that $(B, A) \in x$, and so A is a winning x/y-dialogue tree for A of height 0.

Successor ordinal $\alpha + 1$:
If $A \in J^{\alpha+1}_{x/y}$, then for any B_i such that $(B_i, A) \in x$ there exists a $C_i \in J^{\alpha}_{x/y}$ such that $(C_i, B_i) \in y$. By induction hypothesis, there exist winning x/y-dialogue trees for the C_i. Thus, we have a winning tree rooted for A, with children B_i, whose children are the winning trees for C_i.

Limit ordinal λ:
If $A \in J^{\lambda}_{x/y}$, then there exists an $\alpha < \lambda$ such that $A \in J^{\alpha}_{x/y}$; by induction hypothesis, there exists a winning x/y-dialogue tree of height α for A.

"Only-if"-direction. We prove by transfinite induction: If there exists a winning tree of height α for A, then $A \in J^{\alpha}_{x/y}$.

Base case $\alpha = 0$:
Then there are no arguments B such that $(B, A) \in x$, and so $A \in J^{0}_{x/y}$.

Successor ordinal $\alpha + 1$:
Let T be a tree with root A, whose children are B_i, and the children of B_i are winning trees rooted at C_i. By induction hypothesis, $C_i \in J^{\alpha}_{x/y}$. Because the B_i are all those arguments such that $(B_i, A) \in x$, then A is defended against each B_i by C_i, and so $A \in J^{\alpha+1}_{x/y}$. □

As a corollary, we can relate the proof theory of WFSX and the u/a-proof theory.

Corollary 14 *L is a provably u/a-justified conclusion iff there exists a successful T-tree [2] for L.*

Proof. Follows from the fact that u/a-dialogue trees are sound and complete for u/a-justifiability (Theorem 13), that T-trees are sound and complete for WFSX [2], and that u/a-justifiability and WFSX are equivalent (Theorem 11). □

7 Conclusion and Further Work

We have identified various notions of attack for extended logic programs. Based on these notions of attack, we defined notions of acceptability and least fixpoint semantics. These fixpoint semantics were related by establishing a lattice of justified arguments, based on set inclusion. We identified an argumentation semantics $J_{u/a}$ equal to the well-founded semantics for logic programs with explicit negation, $WFSX$ [2], and established that $J_{Dung} \subseteq J_{PS} \subseteq J_{u/a} = WFSX$, where J_{Dung} and J_{PS} are the least fixpoint argumentation semantics of Dung [6] and Prakken and Sartor [17]. We have defined a dialectical proof theory for argumentation. For all notions of justified arguments introduced, we prove that the proof theory is sound and complete wrt. the corresponding fixpoint argumentation semantics. In particular, we showthe equivalence of successful T-trees [2] in WFSX to provably u/a justified arguments.

Finally, it remains to be seen whether a variation in the notion of attack yields interesting variations of alternative argumentation semantics for extended logic programs such as preferred extensions or stable extensions [6]. It is also an open question how the hierarchy changes when priorities are added as defined in [17, 20].

Acknowledgement This work has been supported by EPSRC grant GRM88433.

References

[1] J. J. Alferes, C. V. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.

[2] J. J. Alferes and L. M. Pereira. *Reasoning with Logic Programming.* (LNAI 1111), Springer-Verlag, 1996.

[3] Garrett Birkhoff. *Lattice Theory.* American Mathematical Society, 3rd edition, 1967.

[4] A. Bondarenko, P.M. Dung, R.A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(1-2):63–101, 1997.

[5] Carlos Iván Chesñevar, Ana Gabriela Maguitman, and Ronald Prescott Loui. Logical models of argument. *ACM Computing Surveys*, 32(4):337–383, December 2000.

[6] P. M. Dung. An argumentation semantics for logic programming with explicit negation. In *Proc. of the 10th International Conference on Logic Programming*, pages 616–630. MIT Press, 1993.

[7] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[8] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, July 1991.

[9] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[10] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proc. of ICLP90*, pages 579–597. MIT Press, 1990.

[11] H. Jakobovits and D. Vermeir. Dialectic semantics for argumentation frameworks. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL '99)*, pages 53–62, 1999.

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

[13] A.C. Kakas and F. Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9:515–562, 1999.

[14] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 104(1-2):1–69, 1998.

[15] Simon Parsons, Carlos Sierra, and Nick Jennings. Agents that reason and negotiate by arguing. *Journal of Logic and Computation*, 8(3):261–292, 1998.

[16] Luis Moniz Pereira and José Júlio Alferes. Well founded semantics for logic programs with explicit negation. In *B. Neumann (Ed.), European Conference on Artificial Intelligence*, pages 102–106. Wiley, 1992.

[17] Henry Prakken and Giovanni Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.

[18] Michael Schroeder. An efficient argumentation framework for negotiating autonomous agents. In *Proceedings of Modelling Autonomous Agents in a Multi-Agent World MAAMAW99.* LNAI1647, Springer-Verlag, 1999.

[19] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[20] Gerard A. W. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90(1–2):225–279, 1997.

[21] Gerd Wagner. *Vivid Logic – Knowledge-Based Reasoning with Two Kinds of Negation*, volume LNAI 764. Springer–Verlag, 1994.

Appendix

Proof of Proposition 10

1. "If"-direction: Induction on the length n of the derivation of $L \in \Gamma(I)$.

   Base case: $n = 1$:
   Then there exists a rule $L \leftarrow not\ L_1, \ldots, not\ L_n$ in P s.t. $L_1, \ldots, L_n \notin I$, and $[L \leftarrow not\ L_1, \ldots, not\ L_n]$ is an argument for L whose assumptions are contained in I.

   Induction step: $n \rightsquigarrow n + 1$:
   Let $L \in \Gamma^{n+1}(I)$. Then there exists a rule $r = L \leftarrow L_1, \ldots, L_n, not\ L'_1, \ldots, L'_m$ in P s.t. $L_i \in \Gamma^n(I)$, and $L'_i \notin I$. By induction hypothesis, there exists arguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$ with $assm(A_i) \subseteq I$. Then $A = [r] \cdot A_1 \cdots A_n$ is an argument for L such that $assm(A) \subseteq I$.

   "Only-if" direction: Induction on the length of the argument.

   Base case: $n = 1$:
   Then $A = [L \leftarrow not\ L_1, \ldots, not\ L_n]$, and $L_1, \ldots, L_n \notin I$. Then $L \leftarrow \in \frac{P}{I}$, and $L \in \Gamma^1(I)$.

   Induction step: $n \rightsquigarrow n + 1$:
   Let $A = [L \leftarrow L_1, \ldots, L_n, not\ L'_1, \ldots, not\ L'_m; r_2, \ldots, r_n]$ be an argument s.t. $assm(A) \subseteq I$. A contains subarguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$, with $assm(A_i) \subseteq I$. Because $L'_1, \ldots, L'_m \notin I$, then $L \leftarrow L_1, \ldots, L_n \in \frac{P}{I}$. By induction hypothesis, $L_i \in \Gamma(I)$. so also $L \in \Gamma(I)$.

2. "If"-direction: Induction on the length n of the derivation of $L \in \Gamma_s(I)$.

   Base case: $n = 1$:
   Then there exists a rule $L \leftarrow not\ L_1, \ldots, not\ L_n$ in P s.t. $\neg L, L_1, \ldots, L_n \notin I$, and $[L \leftarrow not\ L_1, \ldots, not\ L_n]$ is an argument for L whose assumptions are contained in I, and $\neg L \notin I$.

   Induction step: $n \rightsquigarrow n + 1$:
   Let $L \in \Gamma^{n+1}(I)$. Then there exists a rule $r = L \leftarrow L_1, \ldots, L_n, not\ L'_1, \ldots, L'm$ in P s.t. $L_i \in \Gamma^n(I)$, $L'_i \notin I$, and $\neg L \notin I$. By induction hypothesis, there exists arguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$ with $assm(A_i) \subseteq I$ and $\neg conc(A_i) \cap I = \emptyset$. Then $A = [r] \cdot A_1 \cdots A_n$ is an argument for L such that $assm(A) \subseteq I$, and $\neg conc(A) \cap I = \emptyset$.

   "Only-if" direction: Induction on the length of the argument.

   Base case: $n = 1$:
   Then $A = [L \leftarrow not\ L_1, \ldots, not\ L_n]$, and $\neg L, L_1, \ldots, L_n \notin I$. Then $L \leftarrow \in \frac{P_s}{I}$, and $L \in \Gamma^1(I)$.

   Induction step: $n \rightsquigarrow n + 1$:
   Let $A = [L \leftarrow L_1, \ldots, L_n, not\ L'_1, \ldots, not\ L'_m; \ldots]$ be an argument s.t. $assm(A) \subseteq I$, and $\neg conc(A) \cap I = \emptyset$. A contains subarguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$, with $assm(A_i) \subseteq I$, and $\neg conc(A_i) \cap I = \emptyset$. Because $L'_1, \ldots, L'_m \notin I$, and $\neg L \notin I$, then $L \leftarrow L_1, \ldots, L_n \in \frac{P}{I}$. By induction hypothesis, $L_i \in \Gamma(I)$. so also $L \in \Gamma(I)$.

3. and 4. follow immediately from 1. and 2., because I is two-valued.

$\square$