A Parameterised Hierarchy of Argumentation Semantics for Extended Logic Programming and its Application to the Well-founded Semantics

Ralf Schweimeier and Michael Schroeder
Department of Computing, School of Informatics, City University
Northampton Square, London EC1V 0HB, UK
(e-mail: {ralf,msch}@soi.city.ac.uk)

Abstract

Argumentation has proved a useful tool in defining formal semantics for assumption-based reasoning by viewing a proof as a process in which proponents and opponents attack each others arguments by undercuts (attack to an argument’s premise) and rebuts (attack to an argument’s conclusion). In this paper, we formulate a variety of notions of attack for extended logic programs from combinations of undercuts and rebuts and define a general hierarchy of argumentation semantics parameterised by the notions of attack chosen by proponent and opponent. We prove the equivalence and subset relationships between the semantics and examine some essential properties concerning consistency and the coherence principle, which relates default negation and explicit negation. Most significantly, we place existing semantics put forward in the literature in our hierarchy and identify a particular argumentation semantics for which we prove equivalence to the paraconsistent well-founded semantics with explicit negation, WFSX_p. Finally, we present a general proof theory, based on dialogue trees, and show that it is sound and complete with respect to the argumentation semantics.

Keywords: Non-monotonic Reasoning, Extended Logic Programming, Argumentation semantics, Well-founded Semantics with Explicit Negation
1 Introduction
2 Extended Logic Programming and Argumentation
   2.1 Arguments
   2.2 Notions of attack
   2.3 Acceptability and justified arguments
3 Relationships between Notions of Justifiability
   3.1 Equivalence of argumentation semantics
   3.2 Distinguishing argumentation semantics
   3.3 A hierarchy of argumentation semantics
4 Properties of Argumentation Semantics
   4.1 The coherence principle
   4.2 Consistency
5 Argumentation Semantics and WFSX
   5.1 Well-founded semantics with explicit negation
   5.2 Equivalence of argumentation semantics and WFSX_p
6 Proof Theory
   6.1 Dialogue trees
7 Related Work
8 Conclusion and Further Work
References
A Proofs of Theorems
1 Introduction

Argumentation has attracted much interest in the area of Artificial Intelligence. On the one hand, argumentation is an important way of human interaction and reasoning, and is therefore of interest for research into intelligent agents. Application areas include automated negotiation via argumentation (Parsons et al., 1998; Kraus et al., 1998; Schroeder, 1999) and legal reasoning (Prakken & Sartor, 1997). On the other hand, argumentation provides a formal model for various assumption based (or non-monotonic, or default) reasoning formalisms (Bondarenko et al., 1997; Chesn´ever et al., 2000). In particular, various argumentation based semantics have been proposed for logic programming with default negation (Bondarenko et al., 1997; Dung, 1995).

Argumentation semantics are elegant since they can be captured in an abstract framework (Dung, 1995; Bondarenko et al., 1997; Vreeswijk, 1997; Jakobovits & Vermeir, 1999b), 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 (Simari et al., 1994; Dung, 1995; Jakobovits & Vermeir, 1999a).

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 (Pollock, 1987; Prakken & Sartor, 1997) or equivalently ground-attack and reductio-ad-absurdum attack (Dung, 1993). 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 (Bondarenko et al., 1997; Dung, 1995; Kakas & Toni, 1999), some of which are equivalent to existing semantics such as the stable model semantics (Gelfond & Lifschitz, 1988) or the well-founded semantics (van Gelder et al., 1991).

Extended logic programs (Gelfond & Lifschitz, 1990; Alferes & Pereira, 1996; Wagner, 1994), 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 (Dung, 1993; Prakken & Sartor, 1997; M´ora & Alferes, 1998). Dung has shown that a certain argumentation semantics is equivalent to the answer set semantics (Gelfond & Lifschitz, 1990), a generalisation of the stable model
semantics (Gelfond & Lifschitz, 1988). For the well-founded semantics with explicit negation, WFSX (Pereira & Alferes, 1992; Alferes & Pereira, 1996), there exists a scenario semantics (Alferes et al., 1993) which is similar to an argumentation semantics. This semantics applies only to non-contradictory programs; to our knowledge, no argumentation semantics has yet been found equivalent to the paraconsistent well-founded semantics with explicit negation, WFSX_p (Damásio, 1996; Alferes et al., 1995; Alferes & Pereira, 1996).

This paper makes the following contributions: we identify various notions of attack for extended logic programs. We set up a general framework of argumentation semantics, parameterised on these notions of attacks. This framework is then used to classify notions of justified arguments, and to compare them to the argumentation semantics of (Dung, 1993) and (Prakken & Sartor, 1997), among others. We examine some properties of the different semantics, concerning consistency, and the coherence principle which relates explicit and implicit negation. One particular argumentation semantics is then shown to be equivalent to the paraconsistent well-founded semantics with explicit negation (Damásio, 1996). Finally, we develop a general dialectical proof theory for the notions of justified arguments we introduce, and show how proof procedures for these proof theories can be derived. This paper builds upon an earlier conference publication (Schweimeier & Schroeder, 2002), which reports initial findings, while this article provides detailed coverage including all proofs and detailed examples.

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. Section 4 examines some properties (coherence and consistency) of these semantics. In Section 5, we recall the definition of WFSX_p, and prove the equivalence of an argumentation semantics and WFSX_p. A general dialectical proof theory for arguments is presented in Section 6; we prove its soundness and completeness and outline how a proof procedure for the proof theory may be derived.

2 Extended Logic Programming and Argumentation

We introduce extended logic programming and summarise the definitions of arguments associated with extended logic programs. We identify various notions of attack between arguments, and define a variety of semantics parameterised on these notions of attack.

Extended logic programming extends logic programming by two kinds of negation: default negation and explicit negation. The former allows the assumption of the falsity of a fact if there is no evidence for this fact. Explicit negation, on the other hand, allows to explicitly assert the falsity of a fact.

The default negation of a literal \( p \), written \( \text{not} \ p \), states the assumption of the falsity of \( p \). The assumption \( \text{not} \ p \) is intended to be true iff there is no evidence of \( p \). Thus, the truth of \( \text{not} \ p \) relies on a lack of knowledge about \( p \). An operational interpretation of default negation is given by negation as failure (Clark, 1978): the query \( \text{not} \ p \) succeeds iff the query \( p \) fails. Default negation is usually not allowed in the head of a rule: the truth value of \( \text{not} \ p \) is defined in terms of \( p \), and so there should not be any other rules that define \( \text{not} \ p \).

Default negation thus gives a way of expressing a kind of negation, based on a lack of knowledge about a fact. Sometimes, however, it is desirable to express the explicit knowl-
edge of the falsity of a fact. The explicit negation \( \neg p \) of a literal \( p \) states that \( p \) is known to be false. In contrast to default negation, an explicit negation \( \neg p \) is allowed in the head of a rule, and there is no other way of deriving \( \neg p \) except by finding an applicable rule with \( \neg p \) as its consequence.

Consider the following example\(^1\): “A school bus may cross the railway tracks under the condition that there is no approaching train.” It may be expressed using default negation as

\[
cross \leftarrow \text{not train}
\]

This is a dangerous statement, however: assume that there is no knowledge about an approaching train, e.g. because the driver’s view is blocked. In this case, the default negation \( \text{not train} \) is true, and we conclude that the bus may cross. Instead, it would be appropriate to demand the explicit knowledge that there is no approaching train, as expressed using explicit negation:

\[
cross \leftarrow \neg \text{train}
\]

The combination of default and explicit negation also allows for a more cautious statement of positive facts: while the rule

\[
\neg \text{cross} \leftarrow \text{train}
\]

states that the driver should not cross if there is a train approaching, the rule

\[
\neg \text{cross} \leftarrow \text{not } \neg \text{train}
\]

states more cautiously that the driver should not cross if it has not been established that there is no train approaching. In contrast to the former rule, the latter rule prevents a driver from crossing if there is no knowledge about approaching trains.

A connection between the two kind of negations may be made by asserting the coherence principle (Pereira & Alferes, 1992; Alferes & Pereira, 1996): it states that whenever an explicit negation \( \neg p \) is true, then the default negation \( \text{not } p \) is also true. This corresponds to the statement that if something is known to be false, then it should also be assumed to be false.

### 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 \( \text{not } L \) 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

\[
L_0 \leftarrow L_1, \ldots, L_m, \text{not } L_{m+1}, \ldots, \text{not } L_{m+n}(m, n \geq 0),
\]

where 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, \( \text{head}(r) \), and \( L_1, \ldots, \text{not } L_{m+n} \) the **body** of the rule, \( \text{body}(r) \). A rule with an empty body is called a **fact**, and we write \( L_0 \) instead of \( L_0 \leftarrow \).

Our definition of an argument associated with an extended logic program is based on (Prakken\(^1\))

---

\(^1\) Due to John McCarthy, first published in (Gelfond & Lifschitz, 1990)
Ralf Schweimeier and Michael Schroeder & Sartor, 1997). Essentially, an argument is a partial proof, resting on a number of assumptions, i.e. a set of default literals. Note that we do not consider priorities of rules, as used e.g. in (Antoniou, 2002; Kakas & Moraitis, 2002; Prakken & Sartor, 1997; Brewka, 1996; García et al., 1998; Vreeswijk, 1997). Also, we do not distinguish between strict rules, which may not be attacked, and defeasible rules, which may be attacked (Prakken & Sartor, 1997; Simari & Loui, 1992; García et al., 1998).

Definition 2
Let $P$ be an extended logic program. An argument associated with $P$ is a finite sequence $A = [r_1, \ldots, r_n]$ of ground instances 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 $\text{assm}(A)$ for the set of assumptions and $\text{conc}(A)$ 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 is minimal for some literal $L$. Given an extended logic program $P$, we denote the set of minimal arguments associated with $P$ by $\text{Args}_P$.

The restriction to minimal arguments (cf. (Simari & Loui, 1992)) 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. Furthermore, when designing a proof procedure to compute justified arguments, one generally wants to compute only minimal arguments, for reasons of efficiency.

Example 1
Consider the following program:

\[
\begin{align*}
\neg \text{cross} & \leftarrow \neg \text{train} \\
\text{cross} & \leftarrow \neg \text{train} \\
\text{train} & \leftarrow \text{see}\_\text{train} \\
\neg \text{train} & \leftarrow \neg \text{train}, \text{wear}\_\text{glasses} \\
\text{wear}\_\text{glasses} & \\
\end{align*}
\]

The program models the example from the introduction to this section. A bus is allowed to cross the railway tracks if it is known that there is no train approaching; otherwise, it is not allowed to cross. A train is approaching if the driver can see the train, and it is known that there is no train approaching if there is no evidence of a train approaching, and the driver is wearing glasses.

There is exactly one minimal argument with conclusion cross:

\[
\begin{align*}
[\text{cross} & \leftarrow \neg \text{train}; \\
\neg \text{train} & \leftarrow \neg \text{train}, \text{wear}\_\text{glasses}; \text{wear}\_\text{glasses}]
\end{align*}
\]

\[2\] In (Bondarenko et al., 1997; Dung, 1993), an argument is a set of assumptions; the two approaches are equivalent in that there is an argument with a conclusion $L$ if there is a set of assumptions from which $L$ can be inferred. See the discussion in (Prakken & Sartor, 1997).
It contains as subarguments the only minimal arguments for $\neg train$ and $wear\_glasses$:

$$[
eg train \leftarrow not\ train, wear\_glasses]$$

$$[wear\_glasses]$$

There is also exactly one minimal argument with conclusion $\neg cross$:

$$[\neg cross \leftarrow not\ \neg train]$$

There is no argument with conclusion $train$, because there is no rule for $see\_train$.

### 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 (Dung, 1993; Prakken & Sartor, 1997). 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 attack.

**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 $\neg L$ 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 (Dung, 1993; Prakken & Sartor, 1997). The notion of defeat is used in (Prakken & Sartor, 1997), 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. 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.

**Example 2**

Consider the program of example 1. There are the following minimal arguments:

- $A$ : $[cross \leftarrow \neg train; \neg train \leftarrow not\ train, wear\_glasses; wear\_glasses]$  
- $B$ : $[\neg cross \leftarrow not\ \neg train]$  
- $C$ : $[\neg train \leftarrow not\ train, wear\_glasses]$  
- $D$ : $[wear\_glasses]$
The argument $A$ and $B$ rebut each other. The subargument $C$ of $A$ also undercuts $B$, so $A$ also undercuts $B$. Therefore $A$ strongly attacks $B$, while $B$ does not strongly attack or defeat $A$.

**Example 3**

The arguments $[q \leftarrow \neg p]$ and $[p \leftarrow \neg q]$ undercut each other. As a result, they do not strongly undercut each other.

The arguments $[p \leftarrow \neg q]$ and $[\neg p \leftarrow \neg r]$ do not undercut each other, but strongly attack each other.

The argument $[\neg p \leftarrow \neg r]$ strongly undercuts $[p \leftarrow \neg p]$ and $[p \leftarrow \neg \neg p]$ attacks - but does not defeat - the argument $[\neg p \leftarrow \neg r]$.

These notions of attack define for any extended logic program a binary relation on the set of arguments associated with 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 associated with $P$, i.e. $x_P \subseteq \text{Args}_P \times \text{Args}_P$. Notions of attack are partially ordered by defining $x \subseteq y$ iff $\forall P : x_P \subseteq y_P$.

**Notation**

We will use sans-serif font for the specific notions of attack introduced in Definition 3 and their abbreviations: $r$, $u$, $a$, $d$, $sa$, and $su$. We will use $x, y, z, \ldots$ to denote variables for notions of attacks. Arguments are denoted by $A, B, C, \ldots$

The term “attack” is somewhat overloaded: 1. it is the notion of attack $a$ consisting of a rebut or an undercut; we use this terminology because it is standard in the literature (Dung, 1993; Prakken & Sartor, 1997). 2. in general, an attack is a binary relation on the set of arguments of a program; we use the term “notion of attack”. 3. if the argumentation process is viewed as a dialogue between an proponent who puts forward an argument, and an opponent who tries to dismiss it, we may choose one notion of attack for the use of the proponent, and another notion of attack for the opponent. In such a setting, we call the former notion of attack the “defence”, and refer to the latter as “attack”, in the hope that the meaning of the term “attack” will be clear from the context.

**Definition 5**

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

In this relational notation, Definition 3 can be rewritten as $a = u \cup r$, $d = u \cup (r \cup u^{-1})$, $sa = (u \cup r) \cup u^{-1}$, and $su = u \cup u^{-1}$.

**Proposition 1**

The notions of attack of Definition 3 are partially ordered according to the diagram in Figure 1.

**Proof**

A simple exercise, using the set-theoretic laws $A - B \subseteq A \subseteq A \cup C$ and $(A \cup B) - C = (A - C) \cup (B - C)$ (for any arbitrary sets $A$, $B$, and $C$).
A Hierarchy of Argumentation Semantics

As mentioned above, we will work with notions of attack as examined in previous literature. Therefore Figure 1 contains the notions of undercut (Dung, 1993; Prakken & Sartor, 1997), attack (Dung, 1993; Prakken & Sartor, 1997), defeat (Prakken & Sartor, 1997), strong undercut (Prakken & Sartor, 1997), and strong attack as an intermediate notion between strongly undercuts and defeats. All of these notions of attack are extensions of undercuts. The reason is that undercuts are asymmetric, i.e. for two arguments $A, B$, $A \cup B$ does not necessarily imply $B \cup A$. Rebuts, on the other hand, are symmetric, i.e. $A \rightarrow B$ implies $B \rightarrow A$. As a consequence, rebuts on their own always lead to a “draw” between arguments. There is, however, a lot of work on priorities between arguments (Antoniou, 2002; Kakas & Moraitis, 2002; Prakken & Sartor, 1997; Brewka, 1996; García et al., 1998; Vreeswijk, 1997), which implies that rebuts become asymmetric and therefore lead to more interesting semantics. But the original, more basic approach does not consider this extension, and hence undercuts play the prime role and notions of attack mainly based on rebuts, such as $r$ or $r^*$, are not considered.

The following example shows that the inclusions in Figure 1 are strict.

Example 4
Consider the following program:

\[
\begin{align*}
  p & \leftarrow \neg p \\
  p & \leftarrow \neg q \\
  \neg p & \leftarrow \neg r \\
  q & \leftarrow \neg p \\
  \neg q & \leftarrow \neg s
\end{align*}
\]

It has the minimal arguments $[p \leftarrow \neg p], [p \leftarrow \neg q], [\neg p \leftarrow \neg r], [q \leftarrow \neg p], [\neg q \leftarrow \neg s]$. The arguments $[p \leftarrow \neg q]$ and $[q \leftarrow \neg p]$ undercut (and hence defeat) each other, but they do not strongly undercut or strongly attack each other. The arguments $[q \leftarrow \neg r]$ and $[\neg q \leftarrow \neg s]$ strongly attack (and hence defeat) each other, but they do not undercut each other. The argument $[p \leftarrow \neg p]$ attacks $[\neg p \leftarrow \neg r]$, but it does not defeat it, because $[\neg p \leftarrow \neg r]$ (strongly) undercuts $[p \leftarrow \neg p]$. 

Fig. 1. Notions of Attack
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. Our definition of acceptability is parametrised on the notions of attack allowed for the proponent and the opponent.

Acceptability forms the basis for our argumentation semantics, which is defined as the least fixpoint of a function, which collects all acceptable arguments (Pollock, 1987; Simari & Loui, 1992; Prakken & Sartor, 1997; Dung, 1993). The least fixpoint is of particular interest, because it provides a canonical fixpoint semantics and it can be constructed inductively.

Because the semantics is based on parametrised acceptability, we obtain a uniform framework for defining a variety of argumentation semantics for extended logic programs. It can be instantiated to a particular semantics by choosing one notion of attack for the opponent, and another notion of attack as a defence for the proponent. The uniformity of the definition makes it a convenient framework for comparing different argumentation semantics.

**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 $(C, B) \in y$.

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

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

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.

Note that this definition implies that the logic associated with justified arguments is 3-valued, with justified arguments corresponding to true literals, overruled arguments to false literals, and defensible arguments to undefined literals. We could also consider arguments which are both justified and overruled; these correspond to literals with the truth value overdetermined of Belnap’s four-valued logic (Belnap, 1977).

**Proposition 2**
For any program $P$, the operator $F_{P,x/y}$ is monotone. By the Knaster-Tarski fixpoint theorem (Tarski, 1955; Birkhoff, 1967), $F_{P,x/y}$ has a least fixpoint. It can be constructed by transfinite induction as follows:

$$
\begin{align*}
J_0^{x/y} & := \emptyset \\
J_{\alpha+1}^{x/y} & := F_{P,x/y}(J_\alpha^{x/y}) \quad \text{for } \alpha + 1 \text{ a successor ordinal} \\
J_\lambda^{x/y} & := \bigcup_{\alpha < \lambda} J_\alpha^{x/y} \quad \text{for } \lambda \text{ a limit ordinal}
\end{align*}
$$
Then there exists a least ordinal $\lambda_0$ such that $F_{x/y}(J_{\lambda_0}^{x/y}) = J_{\lambda_0}^{x/y} =: J_{x/y}$.

**Proof**

Let $S_1 \subseteq S_2$, and $A \in F_{P,x/y}$, i.e. $A$ is $x/y$-acceptable wrt. $S_1$, i.e. every $x$-attack against $A$ is $y$-attacked by an argument in $S_1$. Then $A$ is also $x/y$-acceptable wrt. $S_2$, because $S_1 \subseteq S_2$, i.e. $S_2$ contains more arguments to defend $A$. □

Note that our general framework encompasses some well-known argumentation semantics for extended logic programs: Dung’s grounded semantics (Dung, 1993) is $J_{a/u}$. Prakken and Sartor’s argumentation semantics (Prakken & Sartor, 1997), without priorities or strict rules is $J_{d/su}$. If we regard explicitly negated literals $\neg L$ as new atoms, unrelated to the positive literal $L$, then we can apply the well-founded argumentation semantics of (Bondarenko et al., 1997; Kakas & Toni, 1999) to extended logic programs, and obtain $J_{u/u}$.

**Example 5**

Consider the following program $P$:

$$
p \leftarrow \neg q
\quad q \leftarrow \neg p
\quad \neg q \leftarrow \neg r
\quad r \leftarrow \neg s
\quad s
\quad \neg s \leftarrow \neg s
$$

Table 1 shows the computation of justified arguments associated with $P$. The columns show various combinations $x/y$ of attack/defence, and a row $n$ shows those arguments $A$ that get added at iteration stage $n$, i.e. $A \in J_{P,x/y}^n$ and $A \notin J_{P,x/y}^{n-1}$.

The set of arguments associated with $P$ is $\{[p \leftarrow \neg q], [q \leftarrow \neg p], [\neg q \leftarrow \neg r], [r \leftarrow \neg s], [s], [\neg s \leftarrow \neg s]\}$.

All arguments are undercut by another argument, except $[s]$; the only attack against $[s]$ is a rebut by $[\neg s \leftarrow \neg s]$, which is not a defeat. Thus, $[s]$ is identified as a justified argument.
at stage 0 in all semantics, except if attacks is allowed as an attack. In the latter case, no argument is justified at stage 0, hence the set of justified arguments $J_{\alpha/x}$ is empty.

3 Relationships between Notions of Justifiability

The definition of justified arguments provides a variety of semantics for extended logic programs, depending on which notion of attack $x$ is admitted to attack an argument, and which notion of attack $y$ may be used as a defence.

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.

3.1 Equivalence of argumentation semantics

We will prove a series of theorems, which show that some of the argumentation semantics defined above are subsumed by others, and that some of them are actually equivalent. Thus, we establish a hierarchy of argumentation semantics, which is 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.

**Theorem 3**
Let $x' \subseteq x$ and $y \subseteq y'$ be notions of attack, then $J_{x/y} \subseteq J_{x'/y'}$.

**Proof**
See Appendix A. □

**Theorem 4**
Let $x$ and $y$ be notions of attack such that $x \supseteq$ undercuts, and let $s y = y^{-1}$ undercuts. 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 $C$ is 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 Theorem 3, we have $J_{x/sy} \subseteq J_{x/y}$. We prove the inverse inclusion by showing that for all ordinals $\alpha$: $J_{x/y}^\alpha \subseteq J_{x/sy}^\alpha$ by transfinite induction on $\alpha$. See Appendix A for the detailed proof. □

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 5
Let \( x \) be a notion of attack such that \( x \supseteq \text{undercuts} \). Then \( J_{x/a} = J_{x/d} = J_{x/\text{sa}} \).

Proof
It follows from Theorems 3 and 4 that \( J_{x/\text{sa}} \subseteq J_{x/d} \subseteq J_{x/a} = J_{x/\text{sa}} \).

The following theorem states that defence with undercuts is equally strong as one with defeats or with attacks, provided the opponent’s permitted attacks include at least the strong attacks.

Theorem 6
Let \( x \) be a notion of attack such that \( x \supseteq \text{strongly attacks} \). Then \( J_{x/u} = J_{x/d} = J_{x/a} \).

Proof
It is sufficient to show that \( J_{x/a} \subseteq J_{x/u} \). Then by Theorem 3, \( J_{x/u} \subseteq J_{x/d} \subseteq J_{x/a} = J_{x/u} \). Informally, 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 \( x \supseteq \text{strongly attacks} \), there must have been an argument defending \( C \) by undercutting \( B \), thereby also defending \( A \) against \( B \).

We prove by transfinite induction that for all ordinals \( \alpha \): \( J_{x/a}^\alpha \subseteq J_{x/u}^\alpha \). See Appendix A for the detailed proof.

In analogy to Theorem 6, strong undercuts are an equivalent defence to strong attacks if the allowed attacks are strong attacks.

Theorem 7
\( J_{\text{sa}/\text{su}} = J_{\text{sa}/\text{sa}} \)

Proof
The proof is similar to the proof of Theorem 6. See Appendix A.

Theorem 8
\( J_{\text{su}/a} = J_{\text{su}/d} \)

Proof
By Theorem 3, \( J_{\text{su}/d} \subseteq J_{\text{su}/a} \). We now show the inverse inclusion. Informally, every strong undercut \( B \) to a \( \text{su}/a \)-justified argument \( A \) is attacked by some \( \text{su}/d \)-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 \( B \).

Formally, we show that for all ordinals \( \alpha \): \( J_{\text{su}/a}^\alpha \subseteq J_{\text{su}/d}^\alpha \), by transfinite induction on \( \alpha \). See Appendix A for the detailed proof.

These results are summarised in a hierarchy of argumentation semantics in Theorem 9 and Figure 2.
3.2 Distinguishing argumentation semantics

The previous section showed equality and subset relationships for a host of notions of justified arguments. In this section we complement these positive findings by negative findings stating for which semantics there are no subset relationships. We prove these negative statements by giving counter-examples distinguishing various notions of justifiability.

The first example shows that, in general, allowing only strong forms of attack for the opponent leads to a more credulous semantics, because in cases where only non-strong attacks exist, every argument is justified.

Example 6

Consider the following program:

\[
\begin{align*}
  p & \leftarrow \text{not } q \\
  q & \leftarrow \text{not } p
\end{align*}
\]

For any notion of attack \(x\), we have \(J_{su/x} = J_{sa/x} = \left\{ [p \leftarrow \text{not } q], [q \leftarrow \text{not } p] \right\}\), 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).

Thus, in general, \(J_{s/x} \not\subseteq J_{w/y}\), for \(s \in \{su, sa\}\), \(w \in \{a, u, d\}\), and any notions of attack \(x\) and \(y\).

The following example shows that some interesting properties need not hold for all argumentation semantics: a fact (i.e. a rule with an empty body) need not necessarily lead to a justified argument; this property distinguishes Dung’s (Dung, 1993) and Prakken and Sartor’s (Prakken & Sartor, 1997) semantics from most of the others.

Example 7

Consider the following program:

\[
\begin{align*}
  p & \leftarrow \text{not } q \\
  q & \leftarrow \text{not } p \\
  \neg p & \leftarrow
\end{align*}
\]

Let \(x\) be a notion of attack. Then \(J_{d/x} = J_{u/x} = \emptyset\), because every argument is defeated (hence attacked). \(J_{su/u} = J_{sa/au} = \left\{ [q \leftarrow \text{not } p] \right\}\), because \([q \leftarrow \text{not } p]\) is the only argument which is not strongly attacked, but it does not strongly attack any other argument. \(J_{u/u} = J_{a/u} = \left\{ [\neg p] \right\}\), because there is no undercut to \([\neg p]\), but \([\neg p]\) does not undercut any other argument. \(J_{a/a} = \{ [\neg p], [q \leftarrow \text{not } p] \}\), because there is no undercut to \([\neg p]\), and the undercut \([p \leftarrow \text{not } p]\) to \([q \leftarrow \text{not } p]\) is attacked by \([\neg p]\). We also have \(J_{sa/au} = \{ [\neg p], [q \leftarrow \text{not } p] \}\), because \([q \leftarrow \text{not } p]\) is not strongly attacked, and the strong attack \([p \leftarrow \text{not } q]\) on \([\neg p]\) is undercut by \([q \leftarrow \text{not } p]\).

Thus, in general, \(J_{d/x} \not\subseteq J_{a/x}\), \(J_{u/x} \not\subseteq J_{a/x}\), \(J_{sa/x} \not\subseteq J_{u/y}\) (where \(sx \in \{su, sa\}\) and \(y \in \{u, su\}\)), and \(J_{u/y} \not\subseteq J_{sa/x}\) (where \(sx \in \{su, sa\}\) and \(y \in \{u, a, d, su, sa\}\)).

The following example is similar to the previous example, except that all the undercuts are strong, whereas in the previous example there were only non-strong undercuts.

Example 8
Consider the following program:

\[
\begin{align*}
p & \leftarrow \text{not } q \\
q & \leftarrow \text{not } r \\
r & \leftarrow \text{not } s \\
s & \leftarrow \text{not } p
\end{align*}
\]

\(~p\)

Let \(x\) be a notion of attack. Then \(J_{sa/x} = \emptyset\), because every argument is strongly attacked. \(J_{su/u} = J_{su/su} = \{[\text{~p}]\}\), because all arguments except \([\text{~p}]\) are strongly undercut, but \([\text{~p}]\) does not undercut any argument. And \(J_{a/a} = J_{su/sa} = J_{su/a} = \{[\text{~p}], [q \leftarrow \text{not } r], [s \leftarrow \text{not } p]\}\), because \([\text{~p}]\) is not undercut, and it defends \([s \leftarrow \text{not } p]\) against the strong undercut \([p \leftarrow \text{not } q]\) (by rebut), and in turn, \([s \leftarrow \text{not } p]\) defends \([q \leftarrow \text{not } r]\) against the strong undercut \([r \leftarrow \text{not } s]\) (by strong undercut).

Thus, \(J_{a/a} \not\subseteq J_{su/y}, J_{su/sa} \not\subseteq J_{su/y}, \) and \(J_{su/a} \not\subseteq J_{su/y}, \) for \(y \in \{u, su\}\).

The following example shows that in certain circumstances, non-strong defence allows for more justified arguments than strong defence.

**Example 9**

Consider the following program:

\[
\begin{align*}
p & \leftarrow \text{not } q \\
q & \leftarrow \text{not } p \\
r & \leftarrow \text{not } p
\end{align*}
\]

Let \(x\) be a notion of attack. Then \(J_{a/x} = J_{d/x} = J_{a/x} = \emptyset\), because every argument is undercut. \(J_{su/su} = J_{su/sa} = J_{sa/su} = J_{sa/sa} = \{[p \leftarrow \text{not } q], [q \leftarrow \text{not } p]\}\) : In these cases, the strong attacks are precisely the strong undercuts; the argument \([r \leftarrow \text{not } p]\) is not justified, because the strong undercut \([p \leftarrow \text{not } q]\) is undercut, but not strongly undercut, by \([q \leftarrow \text{not } p]\). And finally, \(J_{su/a} = J_{sa/a} = J_{sa/a} = \{[p \leftarrow \text{not } q], [q \leftarrow \text{not } p], [r \leftarrow \text{not } p]\}\) : Again, undercuts and attacks, and strong undercuts and strong attacks, coincide; but now \([r \leftarrow \text{not } p]\) is justified, because non-strong undercuts are allowed as defence.

Thus, in general, \(J_{x/u} \not\subseteq J_{x/su}\) and \(J_{x/sa} \not\subseteq J_{x/sa}\), where \(x \in \{su, sa\}\).

The following example distinguishes the argumentation semantics of Dung (Dung, 1993) and Prakken and Sartor (Prakken & Sartor, 1997).

**Example 10**

Consider the following program:

\[
\begin{align*}
p & \leftarrow \text{not } \text{~p} \\
\text{~p}
\end{align*}
\]

Then \(J_{a/x} = \emptyset\), because both arguments attack each other, while \(J_{d/x} = \{[\text{~p}]\}\), because \([\text{~p}]\) defeats \([p \leftarrow \text{not } \text{~p}]\), but not vice versa.

Thus, \(J_{d/x} \not\subseteq J_{a/x}\).

The final example shows that if we do not allow any rebuts as attacks, then we obtain a strictly more credulous semantics.
Example 11
Consider the following program:
\[
\begin{align*}
\neg p & \leftarrow \text{not } q \\
\neg q & \leftarrow \text{not } p \\
p & \\
q &
\end{align*}
\]
Let \( x \) be a notion of attack. Then \( J_{sa/x} = J_{dj/x} = J_{sa/x} = \emptyset \), because every argument is strongly attacked (hence defeated and attacked), while \( J_{a/x} = J_{su/x} = \{[p], [q]\} \).

Thus, in general, \( J_{v/x} \nsubseteq J_{w/y} \), where \( v \in \{a, su\} \) and \( w \in \{a, d, sa\} \), and \( x \) and \( y \) are any notions of attack.

3.3 A hierarchy of argumentation semantics
We now summarise the results of this section, establishing a complete hierarchy of argumentation semantics, parametrised on a pair of notions of attack \( x/y \) where \( x \) stands for the attacks on an argument, and \( y \) for the possible defence. We locate in this hierarchy the argumentation semantics of Dung (Dung, 1993) and Prakken and Sartor (Prakken & Sartor, 1997), as well as the well-founded semantics for normal logic programs (van Gelder et al., 1991). In Section 5 we will show that the paraconsistent well-founded semantics with explicit negation, WFSX, (Damásio, 1996), can also be found in our hierarchy. As a corollary, we obtain precise relationships between these well-known semantics and our argumentation semantics.

**Theorem 9**
The notions of justifiability are ordered (by set inclusion) according to the diagram in Figure 2, where \( x/y \) lies below \( x'/y' \) iff \( J_{x/y} \subset J_{x'/y'} \).

**Proof**
All equality and subset relationships (i.e. arcs between notions of justifiability) depicted in Figure 2 are underpinned by the theorems in section 3.1. Two notions of justifiability are not subsets of each other iff they are not equal and not connected by an arc in Figure 2. These findings are underpinned by the counter-examples of section 3.2. \( \square \)

By definition, Prakken and Sartor’s semantics (Prakken & Sartor, 1997), if we disregard priorities, amounts to d/su-justifiability.

Similarly, Dung’s grounded argumentation semantics (Dung, 1993) is exactly a/u-justifiability; and if we treat explicitly negated literals as new atoms, we can apply the least fixpoint argumentation semantics for normal logic programs (Dung, 1995; Bondarenko et al., 1997) to extended logic programs, which is then, by definition, u/u-justifiability.

Note that these latter semantics use a slightly different notation to ours: arguments are sets of assumptions (i.e. default literals), and a conclusion of an argument is a literal that can be derived from these assumptions. This approach can be translated to ours by taking as arguments all those derivations of a conclusion from an argument. Then the definitions of the notions of attack and the fixpoint semantics coincide. See also the discussion in (Prakken & Sartor, 1997).
As corollaries to Theorem 9 we obtain relationships of these semantics to the other notions of justifiability.

**Corollary 10**
Let $J_{\text{Dung}}$ be the set of justified arguments according to Dung’s grounded argumentation semantics (Dung, 1993). Then $J_{\text{Dung}} = J_{\text{a/su}} = J_{\text{a/u}} = J_{\text{a/d}} = J_{\text{a/sa}}$ and $J_{\text{Dung}} \subseteq J_{x/y}$ for all notions of attack $x \neq a$ and $y$. Thus, in Dung’s semantics, it does not matter which notion of attack, $\text{su, u, a, d, sa}$, is used as a defence, and Dung’s semantics is more sceptical than the others.

**Corollary 11**
Let $J_{\text{PS}}$ be the set of justified arguments according to Prakken and Sartor’s argumentation semantics (Prakken & Sartor, 1997), where all arguments have the same priority. Then $J_{\text{PS}} = J_{\text{d/su}} = J_{\text{d/u}} = J_{\text{d/\text{a}}} = J_{\text{d/\text{sa}}}$, and $J_{\text{PS}} \subseteq J_{x/y}$ for all notions of attack $x \notin \{\text{a, d}\}$ and $y$. Thus, in Prakken and Sartor’s semantics, it does not matter which notion of attack, $\text{su, u, a, d, sa}$, is used as a defence, and $J_{\text{PS}}$ is more credulous than Dung’s semantics, but more sceptical than all the others.

**Corollary 12**
Let $J_{\text{WFS}}$ be the set of justified argument according to the well-founded argumentation semantics for normal logic programs (Dung, 1995; Bondarenko et al., 1997), where an explicitly negated atom $\neg L$ is treated as unrelated to the positive atom $L$. Then $J_{\text{WFS}} = J_{\text{u/u}} = J_{\text{u/\text{su}}} = J_{\text{u/\text{sa}}} \supseteq J_{\text{d/y}} \supseteq J_{\text{a/y}}$, and $J_{\text{WFS}} \subseteq J_{x/y}$ for all notions of attack $y$. Thus, in contrast to Dung’s and Prakken and Sartor’s semantics, for WFS it makes a difference whether rebuts are permitted in the defence ($\text{a, d, sa}$) or not ($\text{u, su}$).

**Remark 1**
1. The notions of $a/x$-, $d/x$- and $sa/x$-justifiability are particularly sceptical in that even a fact $p$ may not be justified, if there is a rule $\neg p \leftarrow B$ (where $\text{not p} \notin B$) that is not $x$-attacked. On the other hand this is useful in terms of avoiding inconsistency.
2. sz/y-justifiability is particularly credulous, because it does not take into account non-strong attacks, so e.g. the program \{p ← not q, q ← not p\} has the justified arguments [p ← not q] and [q ← 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 \(-L\). The answer to this question is no: all the examples in Section 3.2 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 paracomplete well-founded semantics WFSX\(_p\) (Damásio, 1996; Pereira & Alferes, 1992; Alferes & Pereira, 1996) in Section 5.

### 4 Properties of Argumentation Semantics

We will now state some important properties which a semantics for extended logic programs may have, and examine for which of the argumentation semantics these properties hold.

#### 4.1 The coherence principle

The coherence principle for extended logic programming (Alferes & Pereira, 1996) states that “explicit negation implies implicit negation”. If the intended meaning of \(not\ L\) is “if there is no evidence for \(L\), assume that \(L\) is false”, and the intended meaning of \(-L\) is “there is evidence for the falsity of \(L\)”, then the coherence principle states that explicit evidence is preferred over assumption of the lack of evidence. Formally, this can be stated as: if \(-L\) is in the semantics, then \(not\ L\) is also in the semantics. In an argumentation semantics, we have not defined what it means for a default literal to be “in the semantics”. This can easily be remedied, though, and for convenience we introduce the following transformation.\(^3\)

**Definition 8**
Let \(P\) be an extended logic program, and \(x\) and \(y\) notions of attack, and let \(L\) be an objective literal. Then \(L\) is \(x/y\)-justified if there exists a \(x/y\)-justified argument for \(L\).

Let \(nL\) be a fresh atom, and \(P' = P \cup \{nL ← not \ L\}\). Then \(not\ L\) is \(x/y\)-justified if \(\{nL ← not \ L\}\) is a \(x/y\)-justified argument associated with \(P'\).

Note that because \(nL\) is fresh, then either \(J_{x/y}(P') = J_{x/y}(P)\) or \(J_{x/y}(P') = J_{x/y}(P) \cup \{[nL ← not \ L]\}\).

**Definition 9**
A least fixpoint semantics \(J_{x/y}\) satisfies the coherence principle if for every objective literal \(L\), if \(-L\) is \(x/y\)-justified, then \(not\ L\) is \(x/y\)-justified.

\(^3\) The purpose of the transformation could be equally achieved by defining that \(not\ L\) is \(x/y\)-justified if all arguments for \(L\) are overruled.
The following result states that a least fixpoint semantics satisfies the coherence principle exactly in those cases where we allow any attack for the defence. Informally, this is because the only way of attacking a default literal not \( L \) is by undercut, i.e. an argument for \( L \), and in general, such an argument can only be attacked by an argument for \( \neg L \) by a rebut.

**Theorem 13**

Let \( x, y \in \{a, u, d, su, sa\} \). Then \( J_{x/y} \) satisfies the coherence principle iff \( J_{x/a} = J_{x/y} \).

**Proof**

- For the “only if” direction, we show that for those notions of justifiability \( x/y \neq x/a \), the coherence principle does not hold.
  
  Consider the program \( P \):

  \[
  \begin{align*}
  p & \leftarrow \neg q \\
  q & \leftarrow \neg r \\
  r & \leftarrow \neg s \\
  s & \leftarrow \neg p \\
  \neg p & \leftarrow
  \end{align*}
  \]

  Then \( J_{u/u}(P') = J_{su/su}(P') = J_{su/su}(P') = \{\neg p\} \), where \( P' = P \cup \{\neg p \leftarrow \neg not \ p\} \). In these cases, the coherence principle is not satisfied, because \( \neg p \) is justified, but \( \neg not \ p \) is not justified.

- Now consider the program \( Q \):

  \[
  \begin{align*}
  p & \leftarrow \neg \neg p \\
  \neg p & \leftarrow \neg p \\
  \neg p & \leftarrow \neg p \\
  \neg p & \leftarrow \neg p
  \end{align*}
  \]

  Then \( J_{su/su}(Q') = J_{su/su}(Q') = \{p \leftarrow \neg \neg p, \neg p \leftarrow \neg p\} \), where \( Q' = Q \cup \{\neg p \leftarrow \neg p\} \). Again, the coherence principle is not satisfied, because \( \neg p \) is justified, but \( \neg not \ p \) is not justified.

- For the “if” direction, let \( x \) be any notion of attack. Let \( P \) be an extended logic program, and \( \neg L \) a \( x/a \)-justified literal, i.e. there is an argument \( A = \{\neg L \leftarrow Body, \ldots\} \) and an ordinal \( \alpha \) s.t. \( A \in J^\alpha_{x/a} \).

  Let \( A' = [nL \leftarrow \neg L] \), and \( (B, A') \in x \). Because \( nL \) is fresh, the only possible attack on \( A' \) is a strong undercut, i.e. \( L \) is a conclusion of \( B \). Then \( A \) attacks \( B \), and so \( [nL \leftarrow \neg L] \in J^\alpha_{x/a} \).

\( \square \)

### 4.2 Consistency

Consistency is an important property of a logical system. It states that the system does not support contradictory conclusions. In classical logic “ex falso quodlibet”, i.e. if both \( A \) and \( \neg A \) hold, then any formula holds. In paraconsistent systems (Damásio & Pereira, 1998), this property does not hold, thus allowing both \( A \) and \( \neg A \) to hold for a particular formula \( A \), while not supporting any other contradictions.

A set of arguments is **consistent**, or **conflict-free** (Prakken & Sartor, 1997; Dung, 1995), if it does not contain two arguments such that one attacks the other. There are several notions of consistency, depending on which notion of attack is considered undesirable.
Definition 10
Let $x$ be a notion of attack, and $P$ an extended logic program. Then a set of arguments associated with $P$ is called $x$-consistent if it does not contain arguments $A$ and $B$ such that $(A, B) \in x$.

The argumentation semantics of an extended logic program need not necessarily be consistent; because of explicit negation, there exist contradictory programs such as \{p, $\neg p$\}, for which there exist sensible, but inconsistent arguments ($p$ and $\neg p$ in this case).

A general result identifies cases in which the set of justified arguments for a program is consistent. It states that if we allow the attack to be at least as strong as the defence, i.e. if we are sceptical, then the set of justified arguments is consistent.

Theorem 14
Let $x$ and $y$ be notions of attack such that $x \supseteq y$, and let $P$ be an extended logic program. Then the set of $x/y$-justified arguments is $x$-consistent.

Proof
We show that $J_{x/y}^\alpha$ is $x$-consistent for all ordinals $\alpha$, by transfinite induction on $\alpha$.

Base case $\alpha = 0$: Trivial.

Successor ordinal $\alpha \rightarrow \alpha + 1$: Assume $A, B \in J_{x/y}^{\alpha+1}$ and $(A, B) \in x$. Then there exists $C \in J_{x/y}^\alpha$ such that $(C, A) \in y \subseteq x$. Then by induction hypothesis, because $C \in J_{x/y}^\alpha$, then $A \notin J_{x/y}^\alpha$. Because $A \in J_{x/y}^{\alpha+1}$, there exists $D \in J_{x/y}^\alpha$ such that $(D, C) \in y \subseteq x$. This contradicts the induction hypothesis, so we have to retract the assumption and conclude that $J_{x/y}^{\alpha+1}$ is $x$-consistent.

Limit ordinal $\lambda$: Assume $A, B \in J_{x/y}^\lambda$ and $(A, B) \in x$. Then there exist $\alpha, \beta < \lambda$ s.t. $A \in J_{x/y}^\alpha$ and $B \in J_{x/y}^\beta$. W.l.o.g. assume that $\alpha \leq \beta$. Then because $J_{x/y}^\alpha \subseteq J_{x/y}^\beta$, we have $A \in J_{x/y}^\beta$, contradicting the induction hypothesis that $J_{x/y}^\beta$ is $x$-consistent.

The following example shows that, in general, the set of justified arguments may well be inconsistent.

Example 12
Consider the following program:

\[
q \leftarrow \neg p \\
p \\
\neg p
\]

Then $J_{u/a} = \{[q \leftarrow \neg p], [p], [\neg p]\}$, and $[p]$ and $[\neg p]$ rebut each other, and $[p]$ strongly undercut $[q \leftarrow \neg p]$.

5 Argumentation Semantics and WFSX
In this section we will prove that the argumentation semantics $J_{u/a}$ is equivalent to the paraconsistent well-founded semantics with explicit negation WFSX$_p$ (Damácio, 1996; Alferes & Pereira, 1996). First, we summarise the definition of WFSX$_p$. 
5.1 Well-founded semantics with explicit negation

We recollect the definition of the paraconsistent well-founded semantics for extended logic programs, WFSX_p. We use the definition of (Alferes et al., 1995), because it is closer to our definition of argumentation semantics than the original definition of (Pereira & Alferes, 1992).

Definition 11
The set of all objective literals of a program \( P \) is called the Herbrand base of \( P \) and denoted by \( \mathcal{H}(P) \). A paraconsistent interpretation of a program \( P \) is a set \( T \cup \text{not } F \) where \( T \) and \( F \) are subsets of \( \mathcal{H}(P) \). An interpretation is a paraconsistent interpretation where the sets \( T \) and \( F \) are disjoint. An interpretation is called two-valued if \( T \cup F = \mathcal{H}(P) \).

Definition 12
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 \( P' \) is the program obtained from \( P' \) by removing all rules containing a default literal \( \neg A \) 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 \( P'' \), i.e. \( J \) is the least fixpoint of \( T_{P''}(I) := \{ A \mid \exists A \leftarrow B_1, \ldots, B_n \in P'' \text{ s.t. } B_i \in I \} \). Then \( \Gamma P I \) is obtained from \( J \) by replacing the introduced atoms \( \neg A \) by \( \neg A \).

Definition 13
The semi-normal version of a program \( P \) is the program \( P_s \) obtained from \( P \) by replacing every rule \( L \leftarrow \text{Body} \) in \( P \) by the rule \( L \leftarrow \text{not } \neg L, \text{Body} \). If the program \( P \) is clear from the context, we write \( \Gamma I \) for \( \Gamma P I \) and \( \Gamma_s I \) for \( \Gamma P_s I \).

Note that the set \( \Gamma P I \) is just a set of literals; we will now use it to define the semantics of \( P \) as a (paraconsistent) interpretation.

Definition 14
Let \( P \) be a program whose least fixpoint of \( \Gamma_s \) is \( T \). Then the paraconsistent well-founded model of \( P \) is the paraconsistent interpretation \( WFM_p(P) = T \cup \text{not } (\mathcal{H}(P) - \Gamma_s T) \).

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

\[
I_0 := \emptyset \\
I_{\alpha+1} := \Gamma_s I_\alpha \quad \text{for successor ordinal } \alpha + 1 \\
I_\lambda := \bigcup_{\alpha < \lambda} I_\alpha \quad \text{for limit ordinal } \lambda
\]

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

5.2 Equivalence of argumentation semantics and WFSX_p

In this section, we will show that the argumentation semantics \( J_{a/s} \) 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 (Damásio et al., 1997). For non-contradictory programs, the well-founded semantics coincides with the paraconsistent well-founded semantics (Alferes & Pereira, 1996; Damásio, 1996); consequently, we obtain as a corollary that argumentation semantics and well-founded semantics coincide for non-contradictory programs.

Before we come to the main theorem, we need the following Lemma, which shows a precise connection between arguments and consequences of a program $P$.

**Lemma 15**

Let $I$ be a two-valued interpretation.

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

**Proof**

See Appendix A.

In order to compare the argumentation semantics with the well-founded semantics, we extend the definition $\text{conc}(A)$ of the conclusions of a single argument $A$ to work on a set of arguments $\mathcal{A}$. The extended definition $\text{conc}(\mathcal{A})$ includes all positive and negative conclusions of arguments in $\mathcal{A}$; i.e. those literals $L \in \text{conc}(\mathcal{A})$, as well as the default literals $\neg L$ where all arguments for $L$ are overruled by some argument $A \in \mathcal{A}$. We will use this definition of $\text{conc}$ for the set of justified arguments $J_{u/a}$ to compare the “argumentation model” $\text{conc}(J_{u/a})$ to $\text{WF}_M(P)$, the well-founded model.

**Definition 15**

Let $\mathcal{A}$ be a set of arguments. Then

$$\text{conc}(\mathcal{A}) = \bigcup_{A \in \mathcal{A}} \text{conc}(A) \cup \{\neg L \mid \text{all arguments for } L \text{ are overruled by an argument } A \in \mathcal{A}\}$$

With the above definition, we can formulate the main theorem that $u/a$-justified arguments coincide with the well-founded semantics.

**Theorem 16**

Let $P$ be an extended logic program. Then $\text{WF}_M(P) = \text{conc}(J_{u/a})$.

**Proof**

First, note that $A$ undercuts $B$ iff $\exists$ $L$ s.t. $L \in \text{conc}(A)$ and $\neg L \in \text{assm}(B)$; and $A$ rebuts $B$ iff $\exists$ $L \in \text{conc}(A) \cap \neg \text{conc}(B)$.

We show that for all ordinals $\alpha$, $I_\alpha = \text{conc}(J_{u/a}^\alpha)$, by transfinite induction on $\alpha$. The proof
proceeds in two stages. First, we show that all objective literals \( L \) in \( WFM_p(P) \) are conclusions of \( u/a \)-justified arguments and second, that for all default negated literals \( \neg L \) in \( WFM_p(P) \), all arguments for \( L \) are overruled.

**Base case** \( \alpha = 0 \): \( I_\alpha = \emptyset = \text{conc}(J_{u/a}^0) \)

**Successor ordinal** \( \alpha \rightarrow \alpha + 1 \):

\[ L \in I_{\alpha + 1} \]

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

\[ L \in \Gamma_a I_\alpha \]

iff (Lemma 15(1))

\( \exists \) argument \( A \) for \( L \) such that \( \text{assm}(A) \subseteq \Gamma_a I_\alpha \)

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

\( \exists \) argument \( A \) for \( L \) such that \( \forall \neg L \in \text{assm}(A), L \not\in \Gamma_a I_\alpha \)

iff (Lemma 15(4))

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

iff (Induction hypothesis)

\( \exists \) argument \( A \) for \( L \) such that \( \forall \neg L \in \text{assm}(A) \), for any argument \( B \) for \( L \), ( \( \exists \neg L' \in \text{assm}(B) \) s.t. \( \exists \) argument \( C \in J_{u/a}^0 \) for \( L' \), or \( \exists L'' \in \text{conc}(B) \) s.t. \( \exists \) argument \( C \in J_{u/a}^0 \) for \( \neg L'' \) )

iff (Def. of undercut and rebut)

\( \exists \) argument \( A \) for \( L \) such that for any undercut \( B \) to \( A \), ( \( \exists \) argument \( C \in J_{u/a}^0 \) s.t. \( C \) undercuts \( B \), or \( \exists \) argument \( C \in J_{u/a}^0 \) s.t. \( C \) rebuts \( B \) )

iff

\( \exists \) argument \( A \) for \( L \) such that for any undercut \( B \) to \( A \), \( \exists \) argument \( C \in J_{u/a}^0 \) s.t. \( C \) attacks \( B \)

iff (Def. of \( J_{u/a}^{\alpha + 1} \))

\( \exists \) argument \( A \in J_{u/a}^{\alpha + 1} \) for \( L \)

iff (Def. of \( \text{conc} \))

\[ L \in \text{conc}(J_{u/a}^{\alpha + 1}) \]

**Limit ordinal** \( \lambda \):

\[ I_\lambda = \bigcup_{\alpha < \lambda} I_\alpha \] and \( J_{u/a}^\lambda = \bigcup_{\alpha < \lambda} J_{u/a}^\alpha \), so by induction hypothesis \( (I_\alpha = \text{conc}(J_{u/a}^\alpha) \) for all \( \alpha < \lambda \), \( I_\lambda = \text{conc}(J_{u/a}^\lambda) \).

Next we will show that a literal \( \neg L \) is in the well-founded semantics iff every argument for \( L \) is overruled, i.e. \( \neg L \in WFM_p(P) \) implies \( \neg L \in \text{conc}(J_{u/a}) \).

\[ \neg L \in WFM_p(P) \]

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

\[ L \not\in \Gamma_a I_\lambda \]

iff (Lemma 15(4))

for all arguments \( A \) for \( L \), ( \( \exists \) argument \( C \in \text{assm}(A) \) s.t. \( L' \in I_\lambda \), or \( \exists L'' \in \text{conc}(A) \) s.t. \( \neg L'' \in I_\lambda \) )

iff \((I_\lambda = \text{conc}(J_{u/a}^\lambda))\)

for all arguments \( A \) for \( L \), ( \( \exists \) argument \( C \in \text{assm}(A) \) s.t. \( \exists \) argument \( B \in J_{u/a}^\lambda \) for \( L' \), or
∃ \mathcal{L}' \in \text{conc}(\mathcal{A}) \text{ s.t. } ∃ \text{ argument } B \in J_{u/a}^\lambda \text{ for } \neg \mathcal{L}' \\
\text{iff (Def. of undercut and rebut)}
\text{for all arguments } \mathcal{A} \text{ for } \mathcal{L}, (∃ \text{ argument } B \in J_{u/a}^\lambda \text{ s.t. } B \text{ undercuts } \mathcal{A}, \text{ or } ∃ \text{ argument } B \in J_{u/a}^\lambda \text{ s.t. } B \text{ rebuts } \mathcal{A}) \\
\text{iff}
\text{every argument for } \mathcal{L} \text{ is attacked by a justified argument in } J_{u/a}^\lambda \\
\text{iff (Def. of overruled)}
\text{not } \mathcal{L} \in \text{conc}(J_{u/a})
\text{Corollary 17}
Let \mathcal{P} \text{ be a non-contradictory program. Then } WFM(\mathcal{P}) = \text{conc}(J_{u/a}).

\text{Remark 3}
In a similar way, one can show that the } \Gamma \text{ operator corresponds to undercuts, while the } \Gamma_s \text{ operator corresponds to attacks, and so the least fixpoints of } \Gamma, \Gamma_s, \Gamma, \text{ and } \Gamma_s \Gamma_s \text{ correspond to } J_{u/u}, J_{a/u}, \text{ and } J_{a/a}, \text{ respectively. In (Alferes et al., 1995), the least fixpoints of these operators are shown to be ordered as } lfp(\Gamma_s \Gamma) \subseteq lfp(\Gamma \Gamma_s) \subseteq lfp(\Gamma \Gamma), \text{ and } lfp(\Gamma_s \Gamma) \subseteq lfp(\Gamma \Gamma_s) \subseteq lfp(\Gamma \Gamma). \text{ Because } J_{a/u} = J_{a/a} \subseteq J_{u/u} \subseteq J_{u/a} \text{ by Theorem 9, we can strengthen this statement to } lfp(\Gamma_s \Gamma) = lfp(\Gamma \Gamma_s) \subseteq lfp(\Gamma \Gamma) \subseteq lfp(\Gamma \Gamma).

\text{The following corollary summarises the results so far.}

\text{Corollary 18}
The least fixpoint argumentation semantics of Dung (Dung, 1993), denoted } J_{\text{Dung}}, \text{ of Prakken and Sartor (Prakken & Sartor, 1997), denoted } J_{\text{PS}}, \text{ and the well-founded semantics for normal logic programs } WFS (Bondarenko et al., 1997; van Gelder et al., 1991) \text{ and for logic programs with explicit negation } WFSX_p (Pereira & Alferes, 1992; Alferes & Pereira, 1996) \text{ are related to the other least fixpoint argumentation semantics as illustrated in Figure 3.}
6 Proof Theory

One of the benefits of relating the argumentation semantics $J_{u/a}$ to WFSX$_p$ is the existence of an efficient top-down proof procedure for WFSX$_p$ (Alferes et al., 1995), 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 (Simari et al., 1994; Prakken & Sartor, 1997; Jakobovits & Vermeir, 1999a; Kakas & Toni, 1999). 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 $y$.

6.1 Dialogue trees

We adapt the dialectical proof theory of (Prakken & Sartor, 1997) to develop a general sound and complete proof theory for $x/y$-justified arguments.

**Definition 16**

Let $P$ be an extended logic program. An $x/y$-dialogue is a finite nonempty sequence of moves $move_i = (\text{Player}_i, \text{Arg}_i)$ ($i > 0$), such that $\text{Player}_i \in \{P, O\}$, $\text{Arg}_i \in \text{Args}_P$, and

1. $\text{Player}_i = P$ iff $i$ is odd; and $\text{Player}_i = O$ iff $i$ is even.
2. If $\text{Player}_i = \text{Player}_j = P$ and $i \neq j$, then $\text{Arg}_i \neq \text{Arg}_j$.
3. If $\text{Player}_i = P$ and $i > 1$, then $\text{Arg}_i$ is a minimal argument such that $(\text{Arg}_i, \text{Arg}_{i-1}) \in y$.
4. If $\text{Player}_i = O$, then $(\text{Arg}_i, \text{Arg}_{i-1}) \in x$.

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. Note that the minimality condition in 3 is redundant, because all arguments in $\text{Args}_P$ are required to be minimal by Definition 2. We have explicitly repeated this condition, because it is important in that it prevents the proponent from repeating an argument by adding irrelevant rules to it.

**Definition 17**

An $x/y$-dialogue tree is a tree of moves such that every branch is an $x/y$-dialogue, and for all moves $move_i = (P, \text{Arg}_i)$, the children of $move_i$ are all those moves $(O, \text{Arg}_j)$ such that $(\text{Arg}_j, \text{Arg}_i) \in x$.

The height of a dialogue tree is 0 if it consists only of the root, and otherwise $\text{height}(t) = \sup \{\text{height}(t_i)\} + 1$ where $t_i$ are the trees rooted at the grandchildren of $t$.

**Example 13**

Consider the following program:
A a/u-dialogue tree rooted at the argument \([p \leftarrow q, not \ r; q \leftarrow not \ s]\) is given by Figure 4. Each node is marked with \(P\) for proponent or \(O\) for opponent, and an edge \(A \xrightarrow{x} B\) denotes that \(A\) attacks \(B\) with the notion of attack \(x\), i.e. \((A, B) \in x\).

\[
\begin{align*}
0 & \leftarrow q, not \ r \\
q & \leftarrow not \ s \\
\neg q & \leftarrow u \\
r & \leftarrow not \ t \\
s & \leftarrow not \ t \\
t & \leftarrow not \ w \\
u & \leftarrow not v \\
v & \leftarrow not \ r \\
\neg v & \leftarrow not \ t
\end{align*}
\]

Note that although dialogues are required to be finite, dialogue trees may be infinitely branching. Therefore dialogue trees need not be finite, nor need their height be finite.

**Example 14**
Consider the following program \(P^4\):

\[
\begin{align*}
p(0) & \\
p(s(X)) & \leftarrow not q(X) \\
q(X) & \leftarrow not p(X) \\
r & \leftarrow q(X) \\
s & \leftarrow not r
\end{align*}
\]

\(4\) Note that by definition, programs are not allowed to contain variables. Here, \(X\) denotes a variable, and \(P\) is an abbreviation for the (infinite) program obtained by substituting the terms \(s^n(0)\) for the variable \(X\), in all the rules.
For each $n \in \mathbb{N}$, there is exactly one minimal argument $A_n$ with conclusion $p(s^n(0))$, namely $[p(0)]$ for $n = 0$, and $[p(s^n(0)) \leftarrow \text{not } q(s^{n-1}(0))]$ for $n > 0$. Similarly, there is exactly one minimal argument $B_n$ with conclusion $q(s^n(0))$, namely $[q(s^n(0)) \leftarrow \text{not } p(s^n(0))]$.

Therefore, a u/u-dialogue tree rooted at $A_{n+1}$ consists of just one dialogue $T_{n+1}$ of the form $((P, A_{n+1}), (O, B_n), T_n)$. A u/u-dialogue tree rooted at $A_0$ consists only of the root, because there are no undercuts to $A_0$. Thus, the height of the dialogue tree $T_n$ is $n$.

Now consider the u/u-dialogue tree rooted at the argument $C = [s \leftarrow \text{not } r]$. The argument $C$ is undercut by infinitely many arguments $D_n = [r \leftarrow q(s^n(0)); q(s^n(0)) \leftarrow \text{not } p(s^n(0))]$; each $D_n$ is undercut by exactly one argument: $A_n$. A dialogue in the u/u-dialogue tree $T_C$ rooted at argument $C$ is therefore a sequence $((P, C), (O, B_n), T_n)$. Because $\text{height}(T_n) = n$, then by Definition 17: $\text{height}(T_C) = \sup \{\text{height}(T_n) | n \in \mathbb{N}\} + 1 = \omega + 1$.

Definition 18
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.

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

Theorem 19
An argument $A$ is $x/y$-justified iff there exists a $x/y$-dialogue tree with $A$ as its root, and won by the proponent.

Proof
We show by transfinite induction that for all arguments $A$, for all ordinals $\alpha$: $A \in J_{x/y}^\alpha$ if and only if there exists a winning $x/y$-dialogue tree of height $\leq \alpha$ for $A$. See Appendix A for the detailed proof.

7 Related Work
There has been much work on argument-theoretic semantics for normal logic programs, i.e. logic programs with default negation (Bondarenko et al., 1997; Dung, 1995; Kakas & Toni, 1999). Because there is no explicit negation, there is only one form of attack, the undercut in our terminology. An abstract argumentation framework has been defined, which captures other default reasoning mechanisms besides normal logic programming. Within this framework, a variety of semantics may be defined, such as preferred extensions; stable extensions, which are equivalent to stable models (Gelfond & Lifschitz, 1988); and a least fixpoint semantics based on the acceptability of arguments, which is equivalent to the well-founded semantics (van Gelder et al., 1991). The latter fixpoint semantics forms the basis of our argumentation semantics. Proof theories and proof procedures for some of these argumentation semantics have been developed in (Kakas & Toni, 1999).

There has been some work extending this argumentation semantics to logic programs with explicit negation. Dung (Dung, 1995) adapts the framework of (Dung, 1993), by distinguishing between ground attacks and reductio-ad-absurdum-attacks, in our terminology undercuts and rebuts. Argumentation semantics analogous to those of normal logic
programs are defined, and the stable extension semantics is shown to be equivalent to the answer set semantics (Gelfond & Lifschitz, 1990), an adaptation of the stable model semantics to extended logic programs. A least fixpoint semantics (called grounded semantics) based on a notion of acceptability is defined, and related to the well-founded semantics of (van Gelder et al., 1991), although only for the case of programs without explicit negation.

Prakken and Sartor (Prakken & Sartor, 1997) define an argumentation semantics for extended logic programs similar to that of Dung. Their language is more expressive in that it distinguishes between strict rules, which may not be attacked, and defeasible rules, which may be attacked. Furthermore, rules have priorities, and rebuts are only permitted against a rule of equal or lower priority. Thus, rebuts are not necessarily symmetric, as in our setting. Our language corresponds to Prakken and Sartor’s without strict rules, and either without priorities, or, equivalently, if all rules have the same priority. The semantics is given as a least fixpoint of an acceptability operator, analogous to Dung’s grounded semantics. A proof theory, similar to those of Kakas and Toni (Kakas & Toni, 1999) is developed. This proof theory formed the basis of our general proof theory for justified arguments.

In (Móra & Alferes, 1998), an argumentation semantics for extended logic programs, similar to Prakken and Sartor’s, is proposed; it is influenced by WFSX, and distinguishes between sceptical and credulous conclusions of an argument. It also provides a proof theory based on dialogue trees, similar to Prakken and Sartor’s.

Defeasible Logic Programming (García & Simari, 2003; Simari et al., 1994; García et al., 1998) is a formalism very similar to Prakken and Sartor’s, based on the first order logic argumentation framework of (Simari & Loui, 1992). It includes logic programming with two kinds of negation, distinction between strict and defeasible rules, and allowing for various criteria for comparing arguments. Its semantics is given operationally, by proof procedures based on dialectical trees (García & Simari, 2003; Simari et al., 1994). In (Chesñevar et al., 2002), the semantics of Defeasible Logic Programming is related to the well-founded semantics, albeit only for the restricted language corresponding to normal logic programs (van Gelder et al., 1991).

The answer set semantics for extended logic programs (Gelfond & Lifschitz, 1990) is defined via extensions which are stable under a certain program transformation. While this semantics is a natural extension of stable models (Gelfond & Lifschitz, 1988) and provides an elegant model-theoretic semantics, there are several drawbacks which the answer set semantics inherits from the stable models. In particular, there is no efficient top-down proof procedure for the answer set semantics, because the truth value of a literal $L$ may depend on the truth value of a literal $L'$ which does not occur in the proof tree below $L$.\footnote{See the extensive discussion in (Alferes & Pereira, 1996) for details.}

The well-founded semantics (van Gelder et al., 1991) is an approximation of the stable model semantics, for which an efficient top-down proof procedure exists. In (Przymusinski, 1990), the well-founded semantics is adapted to extended logic programs. However, this semantics does not comply with the coherence principle, which states that explicit negation implies implicit negation. In order to overcome this, (Pereira & Alferes, 1992; Alferes & Pereira, 1996) developed WFSX, a well-founded semantics for extended logic
programs, which satisfies the coherence principle. It has several desirable properties not enjoyed by the answer set semantics; in particular, an efficient goal-oriented top-down proof procedure for WFSX is presented in (Alferes et al., 1995). WFSX is well established and e.g. widely available through Prolog implementations such as XSB Prolog (Freire et al., 1997).

Our own work is complementary to these approaches, in that we fill a gap by bringing argumentation and WFSX together in our definition of $u/a$-justified arguments, which are equivalent to WFSX$_p$ (Damásio, 1996; Alferes & Pereira, 1996; Alferes et al., 1995), the paraconsistent version of WFSX. Furthermore, the generality of our framework allows us to relate existing argumentation semantics such as Dung’s and Prakken and Sartor’s approach and thus provide a concise characterisation of all the existing semantics mentioned above.

A number of authors (Kraus et al., 1998; Parsons & Jennings, 1996; Sierra et al., 1997; Parsons et al., 1998; Sadri et al., 2001; Torroni, 2002; Schroeder, 1999; Móra & Alferes, 1998) work on argumentation for negotiating agents. Of these, the approaches of (Sadri et al., 2001; Torroni, 2002; Schroeder, 1999) are based on logic programming. The advantage of the logic programming approach for arguing agents is the availability of goal-directed, top-down proof procedures. This is vital when implementing systems which need to react in real-time and therefore cannot afford to compute all justified arguments, as would be required when a bottom-up argumentation semantics would be used.

In (Sadri et al., 2001; Torroni, 2002), abduction is used to define agent negotiation focusing on the generation of negotiation dialogues using abduction. This work is relevant in that it shows how to embed an argumentation proof procedure into a dialogue protocol, which is needed to apply proof procedures of argumentation semantics as defined in this paper into agent communication languages such as KQML (Finin et al., 1994) or FIPA ACL (Chiariglione et al., 1997).

With a variety of argument-based approaches being pursued to define negotiating agents, the problem of how these agents may inter-operate arises. This paper could serve as a first step towards inter-operation as existing approaches can be placed in our framework, thus making it easier to compare them.

### 8 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. The contributions of this paper are five-fold.

- First, we defined a parameterised hierarchy of argumentation semantics by establishing a lattice of justified arguments based on set inclusion. We showed which argumentation semantics are equal, which are subsets of one another and which are neither.
- Second, we examined some properties of the different semantics, and gave a necessary and sufficient condition for a semantics to satisfy the coherence principle (Alferes & Pereira, 1996), and a sufficient criterion for a semantics to be consistent.
- Third, we identified an argumentation semantics $J_{u/a}$ equal to the paraconsistent
well-founded semantics for logic programs with explicit negation, WFS\textsubscript{X\textsubscript{p}} (Damásio, 1996; Alferes & Pereira, 1996) and proved this equivalence.

- Forth, we established relationships between existing semantics, in particular that $J_{\text{Dung}} \subseteq J_{\text{PS}} \subseteq J_{u/a} = WFS \subseteq J_{u/a} = WFS\textsubscript{X\textsubscript{p}}$, where $J_{\text{Dung}}$ and $J_{\text{PS}}$ are the least fixpoint argumentation semantics of Dung (Dung, 1993) and Prakken and Sartor (Prakken & Sartor, 1997), and $WFS$ is the well-founded semantics without explicit negation (van Gelder et al., 1991).

- Fifth, 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.

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 (Dung, 1993). It is also an open question how the hierarchy changes when priorities are added as defined in (Antoniou, 2002; Kakas & Moraitis, 2002; Prakken & Sartor, 1997; Brewka, 1996; García et al., 1998; Vreeswijk, 1997).

Acknowledgement

Thanks to Iara Carnevale de Almeida and José Júlio Alferes for fruitful discussions on credulous and sceptical argumentation semantics for extended logic programming.

This work has been supported by EPSRC grant GRM88433.

References


Theorem 3
Let \( x' \subseteq x \) and \( y \subseteq y' \) be notions of attack, then \( J_{x/y} \subseteq J_{x'/y'} \).

Proof
We show by transfinite induction that \( J_{x/y}^\alpha \subseteq J_{x'/y'}^\alpha \), for all \( \alpha \).

Base case: \( \alpha = 0 \): Then \( J_{x/y} = \emptyset = J_{x'/y'} \).

Successor ordinal: \( \alpha \rightarrow \alpha + 1 \):
Let \( A \in J_{x/y}^\alpha \), and \( (B, A) \in x' \). Then also \( (B, A) \in x \), and so there exists \( C \in J_{x/y}^\alpha \) such that \( (C, B) \in y \), so also \( (C, B) \in y' \). By induction hypothesis, \( C \in J_{x'/y'}^\alpha \), and so \( A \in J_{x'/y'}^{\alpha + 1} \).
Theorem 4
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
By Theorem 3, we have $J_{x/sy} \subseteq J_{x/y}$. We prove the inverse inclusion by showing that for all ordinals $\alpha$: $J_{x/y}^\alpha \subseteq J_{x/sy}^\alpha$, by transfinite induction on $\alpha$.

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

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

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

Limit ordinal $\lambda$: Assume $J_{x/y}^\alpha \subseteq J_{x/sy}^\alpha$ for all $\alpha < \lambda$. Then $J_{x/y}^\lambda = \bigcup_{\alpha < \lambda} J_{x/y}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{x/sy}^\alpha = J_{x/sy}^\lambda$.

Theorem 6
Let $x$ be a notion of attack such that $x \supseteq$ strongly attacks. Then $J_{x/a} = J_{x/d} = J_{x/u}$.

Proof
It is sufficient to show that $J_{x/a} \subseteq J_{x/u}$. Then by Theorem 3, $J_{x/a} \subseteq J_{x/d} \subseteq J_{x/a} = J_{x/a}$.

We prove by transfinite induction that for all ordinals $\alpha$: $J_{x/a}^\alpha \subseteq J_{x/a}^\alpha$.

Base case: $\alpha = 0$

$J_{x/a}^0 = \emptyset = J_{x/a}^0$.

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

Let $A \in J_{x/a}^{\alpha + 1}$, and $(B, A) \in x$. By definition, there exists $C \in J_{x/a}^\alpha$ such that $C$ undercuts or rebuts $B$. By induction hypothesis, $C \in J_{x/a}^\alpha$.

If $C$ undercuts $B$, then we are done. If, however, $C$ does not undercut $B$, then $C$ rebuts $B$, and so $B$ also rebuts $C$, i.e. $B$ strongly attacks $C$. Because strongly attacks $\subseteq x$ and $C \in J_{x/a}^\alpha$, there exists $D \in J_{x/a}^\alpha \subseteq J_{x/a}^\alpha (\alpha_0 < \alpha)$ such that $D$ undercuts $B$. It follows that $A \in J_{x/a}^{\alpha + 1}$.

Limit ordinal $\lambda$:

Assume $J_{x/a}^\alpha \subseteq J_{x/a}^\alpha$ for all $\alpha < \lambda$. Then $J_{x/a}^\lambda = \bigcup_{\alpha < \lambda} J_{x/a}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{x/a}^\alpha = J_{x/a}^\lambda$. 

Limit ordinal $\lambda$:

Assume $J_{x/a}^\alpha \subseteq J_{x/a}^\alpha$ for all $\alpha < \lambda$. Then $J_{x/a}^\lambda = \bigcup_{\alpha < \lambda} J_{x/a}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{x/a}^\alpha = J_{x/a}^\lambda$. 

\[ J_{x/a}^\lambda = \bigcup_{\alpha < \lambda} J_{x/a}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{x/a}^\alpha = J_{x/a}^\lambda. \]
Theorem 7
\[ J_{sa/su} = J_{sa/sa} \]

Proof
By Theorem 3, \( J_{sa/su} \subseteq J_{sa/sa} \).
We prove the inverse inclusion by showing that for all ordinals \( \alpha \): \( J_{sa/sa}^\alpha \subseteq J_{sa/su}^\alpha \), by transfinite induction on \( \alpha \).

Base case: \( n = 0 \)
\[ J_0^{sa/sa} = \emptyset = J_0^{sa/su} \]

Successor ordinal: \( \alpha \mapsto \alpha + 1 \)
Let \( A \in J_{sa/sa}^{\alpha + 1} \) and \( B \) strongly attacks \( A \). By definition, there exists \( C \in J_{sa/sa}^\alpha \) such that \( C \) attacks \( B \) and \( B \) does not undercut \( C \). By induction hypothesis, \( C \in J_{sa/su}^\alpha \).

If \( C \) undercut \( B \), then we are done. If, however, \( C \) rebuts \( B \) and \( C \) does not undercut \( B \), then \( B \) also rebuts \( C \), i.e. \( B \) strongly attacks \( C \), and so because \( C \in J_{sa/su}^\alpha \) there exists \( D \in J_{sa/su}^{\alpha_0} \subseteq J_{sa/su}^\alpha \) such that \( D \) strongly undercuts \( B \). It follows that \( A \in J_{sa/su}^{\alpha + 1} \).

Limit ordinal \( \lambda \):
Assume \( J_{sa/su}^\alpha \subseteq J_{sa/su}^\lambda \) for all \( \alpha < \lambda \). Then \( J_{sa/su}^\lambda = \bigcup_{\alpha < \lambda} J_{sa/sa}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{sa/su}^\alpha = J_{sa/su}^\lambda \) \( \square \)

Theorem 8
\[ J_{su/a} = J_{su/d} \]

Proof
By Theorem 3, \( J_{su/d} \subseteq J_{su/a} \).
For the inverse inclusion, we show that for all ordinals \( \alpha \): \( J_{su/a}^\alpha \subseteq J_{su/d}^\alpha \), by transfinite induction on \( \alpha \).

Base case: \( \alpha = 0 \)
\[ J_0^{su/a} = \emptyset = J_0^{su/d} \]

Successor ordinal: \( \alpha \mapsto \alpha + 1 \)
Let \( A \in J_{su/a}^{\alpha + 1} \), and \( B \) strongly undercuts \( A \). By definition, there exists \( C \in J_{su/a}^\alpha \) such that \( C \) undercuts or rebuts \( B \). By induction hypothesis, \( C \in J_{su/d}^\alpha \).

If \( C \) undercuts \( B \), or \( B \) does not undercut \( C \), then we are done.

Otherwise, \( B \) strongly undercuts \( C \), and so there exists \( D \in J_{su/d}^{\alpha_0} \subseteq J_{su/d}^\alpha \) such that \( D \) defeats \( B \). It follows that \( A \in J_{su/d}^{\alpha + 1} \).
Limit ordinal $\lambda$:

Assume $J_{su/a}^\alpha \subseteq J_{su/d}^\alpha$ for all $\alpha < \lambda$. Then

$$J_{su/a}^\lambda = \bigcup_{\alpha < \lambda} J_{su/a}^\alpha \subseteq \bigcup_{\alpha < \lambda} J_{su/d}^\alpha = J_{su/d}^\lambda$$

\[ \square \]

Lemma 15

Let $I$ be a two-valued interpretation.

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

Proof

1. “Only If”-direction: Induction on the length $n$ of the derivation of $L \in \Gamma(I)$.
   
   **Base case:** $n = 1$
   
   Then there exists a rule $L \leftarrow \neg L_1, \ldots, \neg L_n$ in $P$ s.t. $L_1, \ldots, L_n \notin I$, and $[L \leftarrow \neg L_1, \ldots, \neg L_n]$ is an argument for $L$ whose assumptions are contained in $I$.

   **Induction step:** $n \rightarrow n + 1$:
   
   Let $L \in \Gamma_{n+1}(I)$. Then there exists a rule $r = L \leftarrow L_1, \ldots, L_n, \neg L'_1, \ldots, \neg L'_m$ 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 $\text{assm}(A_i) \subseteq I$. Then $A = [r] \cdot A_1 \cdots A_n$ is an argument for $L$ such that $\text{assm}(A) \subseteq I$.

   “If” direction: Induction on the length of the argument.

   **Base case:** $n = 1$
   
   Then $A = [L \leftarrow \neg L_1, \ldots, \neg L_n]$, and $L_1, \ldots, L_n \notin I$. Then $L \leftarrow \in \mathcal{P}_T$, and $L \in \Gamma^1(I)$.

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

2. “Only 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 $L \leftarrow \neg L_1, \ldots, \neg L_n$ in $P$ s.t. $\neg L, L_1, \ldots, L_n \notin I$, and $[L \leftarrow \neg L_1, \ldots, \neg L_n]$ is an argument for $L$ whose assumptions are contained in $I$, and $\neg L \notin I$. 
\textit{Induction step: }$n \sim n + 1$:

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. $L_i \in \Gamma^n(I)$, $L'_i \not\in I$, and $\neg L \not\in I$. By induction hypothesis, there exists arguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$ with $\text{assm}(A_i) \subseteq I$ and $\neg \text{conc}(A_i) \cap I = \emptyset$. Then $A = [r] : A_1 \cdots A_n$ is an argument for $L$ such that $\text{assm}(A) \subseteq I$, and $\neg \text{conc}(A) \cap I = \emptyset$.

\textit{“If” direction: Induction on the length of the argument.}

\textbf{Base case: }$n = 1$:

Then $A = [L \leftarrow \text{not } L_1, \ldots, \text{not } L_n]$, and $\neg L, L_1, \ldots, L_n \not\in I$. Then $L \leftarrow \emptyset \not\in P$, and $L \in \Gamma(I)$.

\textbf{Induction step: }$n \sim n + 1$:

Let $A = [L \leftarrow L_1, \ldots, L_n$, not $L'_1, \ldots, \text{not } L'_m; r_2, \ldots, r_n]$ be an argument s.t. $\text{assm}(A) \subseteq I$, and $\neg \text{conc}(A) \cap I = \emptyset$. $A$ contains subarguments $A_1, \ldots, A_n$ for $L_1, \ldots, L_n$, with $\text{assm}(A_i) \subseteq I$, and $\neg \text{conc}(A_i) \cap I = \emptyset$. Because $L'_1, \ldots, L'_m \not\in I$, and $\neg L \not\in I$, then $L \leftarrow L_1, \ldots, L_n \in P$. 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. 

\hfill $\square$

\textbf{Theorem 19}

An argument $A$ is $x/y$-justified iff there exists a $x/y$-dialogue tree with $A$ as its root, and won by the proponent.

\textbf{Proof}

\textit{“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 $\leq \alpha$ for $A$.

\textbf{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 $0$.

\textbf{Successor ordinal }$\alpha + 1$:

If $A \in J_\alpha^{x/y} + 1$, 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 $C_i$.

Furthermore, if any of the $C_i$ contains a move $m = (P, A)$, then it also contains a winning subtree for $A$ rooted at $m$ and we are done. Otherwise, we have a winning tree rooted at $A$, with children $B_i$, whose children are the winning trees for $C_i$.

\textbf{Limit ordinal }$\lambda$:

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 $\alpha$ for $A$.

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

Note that by definition, the height of a dialogue tree is either $0$ or a successor ordinal $\alpha + 1$. So we prove the base case 0, and for the induction step, we assume that the induction
hypothesis holds for all $\beta < \alpha + 1$.

**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 $A \in J^\alpha_{x/y}$. $\square$