# *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*

*Department of Computer Science, Technische Universität Dresden*
*01062 Dresden, Germany*

(*e-mail:* {ralf,msch}@soi.city.ac.uk)

*submitted 4 December 2002; revised 29 June 2003; accepted 12 September 2003*

**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, $\text{WFSX}_p$. 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

**Contents**

## 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 and 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; Chesñevar 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 and 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 and 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 and 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 and Toni 1999), some of which are equivalent to existing semantics such as the stable model semantics (Gelfond and Lifschitz 1988) or the well-founded semantics (van Gelder et al. 1991).

Extended logic programs (Gelfond and Lifschitz 1990; Alferes and 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 and Sartor 1997; Móra and Alferes 1998; **?**). Dung has shown that a certain argumentation semantics is equivalent to the answer set semantics (Gelfond and Lifschitz 1990), a generalisation of the stable model seman-

tics (Gelfond and Lifschitz 1988). For the well-founded semantics with explicit negation, WFSX (Pereira and Alferes 1992; Alferes and 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 and 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 and 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 and 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 parametrised 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 $not\ p$, states the assumption of the falsity of $p$. The assumption $not\ p$ is intended to be true iff there is no evidence of $p$. Thus, the truth of $not\ p$ relies on a lack of knowledge about $p$. An operational interpretation of default negation is given by *negation as failure* (Clark 1978): the query $not\ p$ succeeds iff the query $p$ fails. Default negation is usually not allowed in the head of a rule: the truth value of $not\ p$ is defined in terms of $p$, and so there should not be any other rules that define $not\ p$.

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

$$cross \leftarrow not\ train$$

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 $not\ train$ 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:

$$cross \leftarrow \neg train$$

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

$$\neg cross \leftarrow train$$

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

$$\neg cross \leftarrow not\ \neg train$$

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 and Alferes 1992; Alferes and Pereira 1996): it states that whenever an explicit negation $\neg p$ is true, then the default negation $not\ p$ 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 $not\ L$ where $L$ is an objective literal. A *literal* is either an objective or a default literal.

An *extended logic program* is a (possibly infinite) set of rules of the form

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

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, $head(r)$, and $L_1, \ldots, not\ L_{m+n}$ the *body* of the rule, $body(r)$. 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 and Sartor 1997).

[1] Due to John McCarthy, first published in (Gelfond and Lifschitz 1990)

Essentially, an argument is a partial proof, resting on a number of *assumptions*, i.e. a set of default literals.[2] Note that we do not consider priorities of rules, as used e.g. in (Antoniou 2002; Kakas and Moraitis 2002; Prakken and 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 and Sartor 1997; Simari and 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 $assm(A)$ for the set of assumptions and $conc(A)$ for the set of conclusions of an argument $A$.

An argument $A$ with a conclusion $L$ is a *minimal argument for* $L$ if there is no subargument of $A$ with conclusion $L$. An argument is *minimal* if it is minimal for some literal $L$. Given an extended logic program $P$, we denote the set of minimal arguments associated with $P$ by $Args_P$.

The restriction to minimal arguments (cf. (Simari and 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:

$$\begin{aligned} \neg cross &\leftarrow not\ \neg train \\ cross &\leftarrow \neg train \\ train &\leftarrow see\_train \\ \neg train &\leftarrow not\ train, wear\_glasses \\ wear\_glasses & \end{aligned}$$

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 $cross$:

$$[cross \leftarrow \neg train; \neg train \leftarrow not\ train, wear\_glasses; wear\_glasses]$$

[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 $L$ iff there is a set of assumptions from which $L$ can be inferred. See the discussion in (Prakken and Sartor 1997).

It contains as subarguments the only minimal arguments for $\neg train$ and $wear\_glasses$:

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

$$[wear\_glasses]$$

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 and 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 $not\ L$ is an assumption of $A_2$.
2. $A_1$ *rebuts* $A_2$ if there is an objective literal $L$ such that $L$ is a conclusion of $A_1$ and $\neg L$ is a conclusion of $A_2$.
3. $A_1$ *attacks* $A_2$ if $A_1$ undercuts or rebuts $A_2$.
4. $A_1$ *defeats* $A_2$ if
   - $A_1$ undercuts $A_2$, or
   - $A_1$ rebuts $A_2$ and $A_2$ does not undercut $A_1$.
5. $A_1$ *strongly attacks* $A_2$ if $A_1$ attacks $A_2$ and $A_2$ does not undercut $A_1$.
6. $A_1$ *strongly undercuts* $A_2$ if $A_1$ undercuts $A_2$ and $A_2$ does not undercut $A_1$.

The notions of *undercut* and *rebut*, and hence *attack* are fundamental for extended logic programs (Dung 1993; Prakken and Sartor 1997). The notion of *defeat* is used in (Prakken and 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 $[q \leftarrow not\ p]$ and $[p \leftarrow not\ q]$ undercut each other. As a result, they do not strongly undercut each other.

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

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

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. $x_P \subseteq Args_P \times Args_P$. 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: $\mathsf{r}$, $\mathsf{u}$, $\mathsf{a}$, $\mathsf{d}$, $\mathsf{sa}$, and $\mathsf{su}$. 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 $\mathsf{a}$ consisting of a rebut or an undercut; we use this terminology because it is standard in the literature (Dung 1993; Prakken and 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 $x_P^{-1} = \{(B, A) \mid (A, B) \in x_P\}$.

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

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

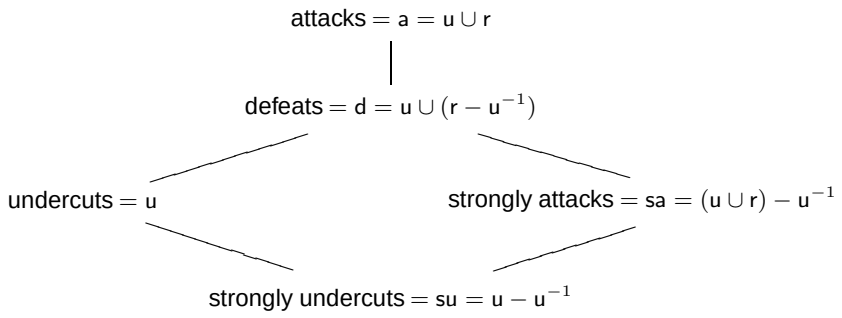


Fig. 1. Notions of Attack

*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 $C$). □

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 and Sartor 1997), attack (Dung 1993; Prakken and Sartor 1997), defeat (Prakken and Sartor 1997), strong undercut (Prakken and 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 $A$, $B$, $A\mathsf{u}B$ does not necessarily imply $B\mathsf{u}A$. Rebuts, on the other hand, are symmetric, i.e. $A\mathsf{r}B$ implies $B\mathsf{r}A$. 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 and Moraitis 2002; Prakken and 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 $\mathsf{r}$ or $\mathsf{r} - \mathsf{u}^{-1}$, are not considered.

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

*Example 4*
Consider the following program:

$$\begin{array}{rcl} p & \leftarrow & not\ \neg p \\ p & \leftarrow & not\ q \\ \neg p & \leftarrow & not\ r \\ q & \leftarrow & not\ p \\ \neg\, q & \leftarrow & not\ s \end{array}$$

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

do not undercut each other. The argument $[p \leftarrow not\ \neg p]$ attacks $[\neg p \leftarrow not\ r]$, but it does not defeat it, because $[\neg p \leftarrow not\ r]$ (strongly) undercuts $[p \leftarrow not\ \neg p]$.

### *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 and Loui 1992; Prakken and 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 $(C, B) \in y$.

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

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

| | a/$x$ | d/$x$ | u/u = u/su | u/a = u/d = u/sa | sa/sa = sa/su | sa/a = sa/d = sa/u | su/$x$ |
|---|---|---|---|---|---|---|---|
| 1 | $\emptyset$ | $[s]$ | $[s]$ | $[s]$ | $[p \leftarrow not\ q]$, $[s]$ | $[p \leftarrow not\ q]$, $[s]$ | $[p \leftarrow not\ q]$, $[q \leftarrow not\ p]$, $[s]$ |
| 2 | $\emptyset$ | $\emptyset$ | $[\neg q \leftarrow not\ r]$ | $[\neg q \leftarrow not\ r]$ | $\emptyset$ | $[\neg q \leftarrow not\ r]$ | $[\neg q \leftarrow not\ r]$ |
| 3 | $\emptyset$ | $\emptyset$ | $\emptyset$ | $[p \leftarrow not\ q]$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| 4 | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |

Table 1. Computing justified arguments – the $n$-th row shows the justified arguments added at the $n$-th iteration

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

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 $J_{\mathsf{a/u}}$. Prakken and Sartor's argumentation semantics (Prakken and Sartor 1997), without priorities or strict rules is $J_{\mathsf{d/su}}$. 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 and Toni 1999) to extended logic programs, and obtain $J_{\mathsf{u/u}}$.

*Example 5*
Consider the following program $P$:

$$
\begin{array}{rcl}
p & \leftarrow & not\ q \\
q & \leftarrow & not\ p \\
\neg q & \leftarrow & not\ r \\
r & \leftarrow & not\ s \\
s & & \\
\neg s & \leftarrow & not\ s
\end{array}
$$

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. $A \in J^n_{P,x/y}$ and $A \notin J^{n-1}_{P,x/y}$.

The set of arguments associated with $P$ is $\{[p \leftarrow not\ q], [q \leftarrow not\ p], [\neg q \leftarrow not\ r], [r \leftarrow not\ s], [s], [\neg s \leftarrow not\ s]\}$.

All arguments are undercut by another argument, except $[s]$; the only attack against $[s]$ is a rebut by $[\neg s \leftarrow not\ s]$, 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 $J_{\mathsf{a}/x}$ 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 states that it does not make a difference if we allow only the strong version of the defence. This is because an argument need not defend itself on its own, but it may rely on other arguments to defend it.

*Theorem 4*
Let $x$ and and $y$ be notions of attack such that $x \supseteq \mathsf{undercuts}$, and let $sy = y - \mathsf{undercuts}^{-1}$. Then $J_{x/y} = J_{x/sy}$.

*Proof*
Informally, every $x$-attack $B$ to an $x/y$-justified argument $A$ is $y$-defended by some $x/sy$-justified argument $C$ (by induction). Now if $C$ is *not* a $sy$-attack, then it is undercut by $B$, and because $x \supseteq \mathsf{undercuts}$ 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 $\alpha$: $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x/sy}$, 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 $x \supseteq$ undercuts. Then $J_{x/\mathsf{a}} = J_{x/\mathsf{d}} = J_{x/\mathsf{sa}}$.

*Proof*
It follows from Theorems 3 and 4 that $J_{x/\mathsf{sa}} \subseteq J_{x/\mathsf{d}} \subseteq J_{x/\mathsf{a}} = J_{x/\mathsf{sa}}$. □

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 $x \supseteq$ strongly attacks. Then $J_{x/\mathsf{u}} = J_{x/\mathsf{d}} = J_{x/\mathsf{a}}$.

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

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

We prove by transfinite induction that for all ordinals $\alpha$: $J^{\alpha}_{x/\mathsf{a}} \subseteq J^{\alpha}_{x/\mathsf{u}}$. 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*
$J_{\mathsf{sa}/\mathsf{su}} = J_{\mathsf{sa}/\mathsf{sa}}$

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

*Theorem 8*
$J_{\mathsf{su}/\mathsf{a}} = J_{\mathsf{su}/\mathsf{d}}$

*Proof*
By Theorem 3, $J_{\mathsf{su/d}} \subseteq J_{\mathsf{su/a}}$.

We now show the inverse inclusion. Informally, every strong undercut $B$ to a $\mathsf{su/a}$-justified argument $A$ is attacked by some $\mathsf{su/d}$-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 $\alpha$: $J^{\alpha}_{\mathsf{su/a}} \subseteq J^{\alpha}_{\mathsf{su/d}}$, 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:

$$\begin{array}{rcl} p & \leftarrow & not\ q \\ q & \leftarrow & not\ p \end{array}$$

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

Thus, in general, $J_{s/x} \not\subseteq J_{w/y}$, for $s \in \{\mathsf{su}, \mathsf{sa}\}$, $w \in \{\mathsf{a}, \mathsf{u}, \mathsf{d}\}$, 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 and Sartor 1997) semantics from most of the others.

*Example 7*
Consider the following program:

$$\begin{array}{rcl} p & \leftarrow & not\ q \\ q & \leftarrow & not\ p \\ \neg p & & \end{array}$$

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

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

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:

$$\begin{array}{rcl} p & \leftarrow & not\ q \\ q & \leftarrow & not\ r \\ r & \leftarrow & not\ s \\ s & \leftarrow & not\ p \\ \neg p & & \end{array}$$

Let $x$ be a notion of attack. Then $J_{\mathsf{sa}/x} = \emptyset$, because every argument is strongly attacked.

$J_{\mathsf{su}/\mathsf{u}} = J_{\mathsf{su}/\mathsf{su}} = \{[\neg p]\}$, because all arguments except $[\neg p]$ are strongly undercut, but $[\neg p]$ does not undercut any argument. And $J_{\mathsf{u}/\mathsf{a}} = J_{\mathsf{su}/\mathsf{sa}} = J_{\mathsf{su}/\mathsf{a}} = \{[\neg p], [q \leftarrow not\ r], [s \leftarrow not\ p]\}$, because $[\neg p]$ is not undercut, and it defends $[s \leftarrow not\ p]$ against the strong undercut $[p \leftarrow not\ q]$ (by rebut), and in turn, $[s \leftarrow not\ p]$ defends $[q \leftarrow not\ r]$ against the strong undercut $[r \leftarrow not\ s]$ (by strong undercut).

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

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:

$$\begin{array}{rcl} p & \leftarrow & not\ q \\ q & \leftarrow & not\ p \\ r & \leftarrow & not\ p \end{array}$$

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

Thus, in general, $J_{x/\mathsf{u}} \not\subseteq J_{x/\mathsf{su}}$ and $J_{x/\mathsf{a}} \not\subseteq J_{x/\mathsf{sa}}$, where $x \in \{\mathsf{su}, \mathsf{sa}\}$.

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

*Example 10*
Consider the following program:

$$\begin{array}{rcl} p & \leftarrow & not\ \neg p \\ \neg p & & \end{array}$$

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

Thus, $J_{\mathsf{d}/x} \not\subseteq J_{\mathsf{a}/x}$.

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:

$$\begin{array}{rcl} \neg p & \leftarrow & not\ q \\ \neg q & \leftarrow & not\ p \\ p & & \\ q & & \end{array}$$

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

Thus, in general, $J_{v/x} \not\subseteq J_{w/y}$, where $v \in \{\mathsf{u}, \mathsf{su}\}$, $w \in \{\mathsf{a}, \mathsf{d}, \mathsf{sa}\}$, 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 and 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$_p$ (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 $J_{x/y} \subsetneq J_{x'/y'}$.

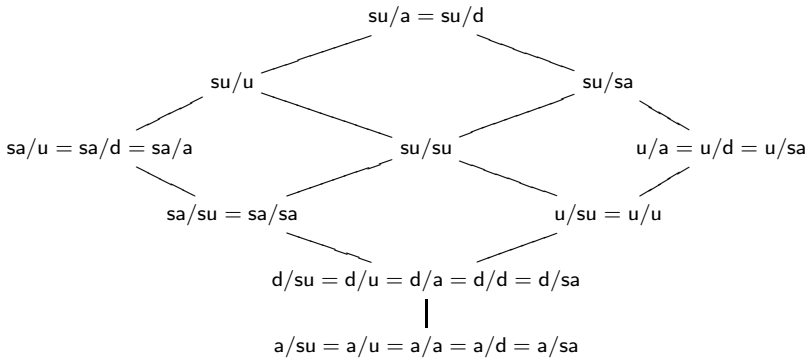


Fig. 2. Hierarchy of Notions of Justifiability

*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. □

By definition, Prakken and Sartor's semantics (Prakken and 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 and Sartor 1997).

As corollaries to Theorem 9 we obtain relationships of these semantics to the other notions of justifiability.

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

*Corollary 11*
Let $J_{PS}$ be the set of justified arguments according to Prakken and Sartor's argumentation semantics (Prakken and Sartor 1997), where all arguments have the same priority. Then $J_{PS} = J_{\mathsf{d/su}} = J_{\mathsf{d/u}} = J_{\mathsf{d/a}} = J_{\mathsf{d/d}} = J_{\mathsf{d/sa}}$, $J_{PS} \subsetneq J_{x/y}$ for all notions of attack

$x \notin \{\mathsf{a}, \mathsf{d}\}$ and $y$, and $J_{PS} \supsetneq J_{\mathsf{a}/y}$ for all notions of attack $y$. Thus, in Prakken and Sartor's semantics, it does not matter which notion of attack, su,u,a,d,sa, is used as a defence, and $J_{PS}$ is more credulous than Dung's semantics, but more sceptical than all the others.

*Corollary 12*
Let $J_{WFS}$ 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 $J_{WFS} = J_{\mathsf{u}/\mathsf{u}} = J_{\mathsf{u}/\mathsf{su}}$, $J_{WFS} \supsetneq J_{\mathsf{d}/y} \supsetneq J_{\mathsf{a}/y}$, $J_{WFS} \subsetneq J_{\mathsf{su}/y}$, and $J_{WFS} \subsetneq J_{\mathsf{u}/\mathsf{a}} = J_{\mathsf{u}/\mathsf{d}} = J_{\mathsf{u}/\mathsf{sa}}$, 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 (a,d,sa) or not (u,su).

*Remark 1*
1. The notions of $\mathsf{a}/x$-, $\mathsf{d}/x$- and $\mathsf{sa}/x$-justifiability are particularly sceptical in that even a fact $p$ may not be justified, if there is a rule $\neg p \leftarrow B$ (where $not\ p \notin B$) that is not $x$-attacked. On the other hand this is useful in terms of avoiding inconsistency.
2. $sx/y$-justifiability is particularly credulous, because it does not take into account non-strong attacks, so e.g. the program $\{p \leftarrow not\ q, q \leftarrow not\ p\}$ has the justified arguments $[p \leftarrow not\ q]$ and $[q \leftarrow not\ p]$.

*Remark 2*
One might ask whether any of the semantics in Figure 2 are equivalent for *non-contradictory* programs, i.e. programs for which there is no literal $L$ such that there exist justified arguments for both $L$ and $\neg L$. The answer to this question is no: all the examples 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 $\mathsf{u}/\mathsf{a}$-justifiability, which will be shown equivalent to the paraconsistent well-founded semantics WFSX$_p$ (Damásio 1996; Pereira and Alferes 1992; Alferes and 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 and 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 $\neg L$ 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 $\neg L$

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 $P' = P \cup \{nL \leftarrow not\ L\}$. Then $not\ L$ is $x/y$-*justified* if $[nL \leftarrow not\ L]$ 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 $J_{x/y}(P') = J_{x/y}(P) \cup \{[nL \leftarrow not\ L]\}$.

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

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

*Proof*

- For the "only if" direction, we show that for those notions of justifiability $x/y \neq x/\mathsf{a}$, the coherence principle does not hold.
  - — Consider the program $P$:
    $$\begin{array}{rcl} p & \leftarrow & not\ q \\ q & \leftarrow & not\ r \\ r & \leftarrow & not\ s \\ s & \leftarrow & not\ p \\ \neg p & & \end{array}$$
    Then $J_{\mathsf{u}/\mathsf{u}}(P') = J_{\mathsf{su}/\mathsf{u}}(P') = J_{\mathsf{su}/\mathsf{su}}(P') = \{[\neg p]\}$, where $P' = P \cup \{np \leftarrow not\ p\}$. In these cases, the coherence principle is not satisfied, because $\neg p$ is justified, but $not\ p$ is not justified.
  - — Now consider the program $Q$:
    $$\begin{array}{rcl} p & \leftarrow & not\ \neg p \\ \neg p & \leftarrow & not\ p \end{array}$$
    Then $J_{\mathsf{su}/\mathsf{sa}}(Q') = J_{\mathsf{sa}/\mathsf{sa}}(Q') = \{[p \leftarrow not\ \neg p], [\neg p \leftarrow not\ p]\}$, where $Q' = Q \cup \{np \leftarrow not\ p\}$. Again, the coherence principle is not satisfied, because $\neg p$ is justified, but $not\ p$ is not 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.

- For the "if" direction, let $x$ be any notion of attack. Let $P$ be an extended logic program, and $\neg L$ a $x/\mathsf{a}$-justified literal, i.e. there is an argument $A = [\neg L \leftarrow Body, \ldots]$ and an ordinal $\alpha$ s.t. $A \in J^{\alpha}_{x/\mathsf{a}}$.
  Let $A' = [nL \leftarrow not\ L]$, 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 $[nL \leftarrow not\ L] \in J^{\alpha+1}_{x/\mathsf{a}}$.

□

### *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 and 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 and 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 $(A, B) \in x_P$.

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 $J^{\alpha}_{x/y}$ is $x$-consistent for all ordinals $\alpha$, by transfinite induction on $\alpha$.
*Base case* $\alpha = 0$: Trivial.
*Successor ordinal* $\alpha \rightsquigarrow \alpha + 1$: Assume $A, B \in J^{\alpha+1}_{x/y}$ and $(A, B) \in x$. Then there exists $C \in J^{\alpha}_{x/y}$ such that $(C, A) \in y \subseteq x$. Then by induction hypothesis, because $C \in J^{\alpha}_{x/y}$, then $A \notin J^{\alpha}_{x/y}$. Because $A \in J^{\alpha+1}_{x/y}$, there exists $D \in J^{\alpha}_{x/y}$ such that $(D, C) \in y \subseteq x$. This contradicts the induction hypothesis, so we have to retract the assumption and conclude that $J^{\alpha+1}_{x/y}$ is $x$-consistent.
*Limit ordinal* $\lambda$*:* Assume $A, B \in J^{\lambda}_{x/y}$ and $(A, B) \in x$. Then there exist $\alpha, \beta < \lambda$ s.t. $A \in J^{\alpha}_{x/y}$ and $B \in J^{\beta}_{x/y}$. W.l.o.g. assume that $\alpha \leq \beta$. Then because $J^{\alpha}_{x/y} \subseteq J^{\beta}_{x/y}$, we have $A \in J^{\beta}_{x/y}$, contradicting the induction hypothesis that $J^{\beta}_{x/y}$ 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:

$$\begin{array}{rcl} q & \leftarrow & not\ p \\ p & & \\ \neg p & & \end{array}$$

Then $J_{\mathsf{u/a}} = \{[q \leftarrow not\ p], [p], [\neg p]\}$, and $[p]$ and $[\neg p]$ rebut each other, and $[p]$ strongly undercuts $[q \leftarrow not\ p]$.

## 5 Argumentation Semantics and WFSX

In this section we will prove that the argumentation semantics $J_{\mathsf{u/a}}$ is equivalent to the paraconsistent well-founded semantics with explicit negation WFSX$_p$ (Damásio 1996; Alferes and 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 and 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 $T \cup not\ F$ 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 $\frac{P'}{I'}$ is the program obtained from $P'$ by removing all rules containing a default literal $not\ A$ such that $A \in I'$, and then removing all remaining default literals from $P'$, obtaining a definite program $P''$. Let $J$ be the least model of $P''$, 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 $L \leftarrow Body$ in $P$ by the rule $L \leftarrow not\ \neg L, Body$. 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 $\Gamma\Gamma_s$ is $T$. Then the *paraconsistent well-founded model of* $P$ is the paraconsistent interpretation $WFM_p(P) = T \cup not\ (\mathcal{H}(P) - \Gamma_s T)$. If $WFM_p(P)$ is an interpretation, then $P$ is called *non-contradictory*, and $WFM_p(P)$ is the *well-founded model of* $P$, denoted by $WFM(P)$.

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

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

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

### *5.2 Equivalence of argumentation semantics and WFSX$_p$*

In this section, we will show that the argumentation semantics $J_{\mathsf{u/a}}$ and the well-founded model coincide. That is, the conclusions of justified arguments are exactly the objective literals which are true in the well-founded model; and those objective literals all of whose arguments are overruled are exactly the literals which are false in the well-founded model. The result holds also for contradictory programs under the *paraconsistent* well-founded semantics. This is important, because it shows that contradictions in the argumentation semantics are precisely the contradictions under the well-founded semantics, and allows the application of contradiction removal (or avoidance) methods to the argumentation semantics (Damásio et al. 1997). For non-contradictory programs, the well-founded semantics coincides with the paraconsistent well-founded semantics (Alferes and 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 $\frac{P}{I}$.

*Lemma 15*
Let $I$ be a two-valued interpretation.

1. $L \in \Gamma(I)$ iff $\exists$ argument $A$ with conclusion $L$ such that $assm(A) \subseteq I$.
2. $L \in \Gamma_s(I)$ iff $\exists$ argument $A$ with conclusion $L$ such that $assm(A) \subseteq I$ and $\neg conc(A) \cap I = \emptyset$.
3. $L \notin \Gamma(I)$ iff $\forall$ arguments $A$ with conclusion $L$, $assm(A) \cap I \not\subseteq \emptyset$.
4. $L \notin \Gamma_s(I)$ iff $\forall$ arguments $A$ with conclusion $L$, $assm(A) \cap I \not\subseteq \emptyset$ or $\neg conc(A) \cap I \neq \emptyset$.

*Proof*
See Appendix A. □

In order to compare the argumentation semantics with the well-founded semantics, we extend the definition $conc(A)$ of the conclusions of a single argument $A$ to work on a set of arguments $\mathcal{A}$. The extended definition $conc(\mathcal{A})$ includes all positive and negative conclusions of arguments in $\mathcal{A}$; i.e. those literals $L \in conc(\mathcal{A})$, as well as the default literals

$not\ L$ where all arguments for $L$ are overruled by some argument $A \in \mathcal{A}$. We will use this definition of $conc$ for the set of justified arguments $J_{\mathsf{u/a}}$ to compare the "argumentation model" $conc(J_{\mathsf{u/a}})$ to $WFM_p(P)$, the well-founded model.

*Definition 15*
Let $\mathcal{A}$ be a set of arguments. Then

$$conc(\mathcal{A}) = \bigcup_{A \in \mathcal{A}} conc(A) \cup \{not\ L \mid \text{all arguments for } L \text{ are overruled by an argument } A \in \mathcal{A}\}$$

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 $WFM_p(P) = conc(J_{\mathsf{u/a}})$.

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

We show that for all ordinals $\alpha$, $I_\alpha = conc(J^\alpha_{\mathsf{u/a}})$, 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 $not\ L$ in $WFM_p(P)$, all arguments for $L$ are overruled.

*Base case* $\alpha = 0$*:* $I_\alpha = \emptyset = conc(J^\alpha_{\mathsf{u/a}})$

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

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

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

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

iff (Lemma 15(1))

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

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

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

iff (Lemma 15(4))

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

iff (Induction hypothesis)

$\exists$ argument $A$ for $L$ such that $\forall\ not\ L \in assm(A)$, for any argument $B$ for $L$, ( $\exists\ not\ L' \in assm(B)\ s.t.\ \exists$ argument $C \in J^\alpha_{\mathsf{u/a}}$ for $L'$, or $\exists\ L'' \in conc(B)\ s.t.\ \exists$ argument $C \in J^\alpha_{\mathsf{u/a}}$ for $\neg L''$)

iff (Def. of undercut and rebut)

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

iff

$\exists$ argument $A$ for $L$ such that for any undercut $B$ to $A$, $\exists$ argument $C \in J^\alpha_{\mathsf{u/a}}$ s.t. $C$ attacks $B$

iff (Def. of $J^{\alpha+1}_{\mathsf{u/a}}$)

$\exists$ argument $A \in J^{\alpha+1}_{\mathsf{u/a}}$ for $L$

iff (Def. of $conc$)

$L \in conc(J^{\alpha+1}_{\mathsf{u/a}})$

*Limit ordinal $\lambda$:*

$I_\lambda = \bigcup_{\alpha<\lambda} I_\alpha$ and $J^{\lambda}_{\mathsf{u/a}} = \bigcup_{\alpha<\lambda} J^{\alpha}_{\mathsf{u/a}}$, so by induction hypothesis ($I_\alpha = conc(J^{\alpha}_{\mathsf{u/a}})$ for all $\alpha < \lambda$), $I_\lambda = conc(J^{\lambda}_{\mathsf{u/a}})$.

Next we will show that a literal $not\ L$ is in the well-founded semantics iff every argument for $L$ is overruled, i.e. $not\ L \in WFM_p(P)$ implies $not\ L \in conc(J_{\mathsf{u/a}})$.

$not\ L \in WFM_p(P)$

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

$L \notin \Gamma_s I_\lambda$

iff (Lemma 15(4)

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

iff ($I_\lambda = conc(J^{\lambda}_{\mathsf{u/a}})$)

for all arguments $A$ for $L$, ( $\exists\ not\ L' \in assm(A)\ s.t.\ \exists$ argument $B \in J^{\lambda}_{\mathsf{u/a}}$ for $L'$, or $\exists\, L'' \in conc(A)\ s.t.\ \exists$ argument $B \in J^{\lambda}_{\mathsf{u/a}}$ for $\neg L''$ )

iff (Def. of undercut and rebut)

for all arguments $A$ for $L$, ( $\exists$ argument $B \in J^{\lambda}_{\mathsf{u/a}}$ s.t. $B$ undercuts $A$, or $\exists$ argument $B \in J^{\lambda}_{\mathsf{u/a}}$ s.t. $B$ rebuts $A$ )

iff

every argument for $L$ is attacked by a justified argument in $J^{\lambda}_{\mathsf{u/a}}$

iff (Def. of overruled)

every argument for $L$ is overruled

iff (Def. of $conc(J_{\mathsf{u/a}})$)

$not\ L \in conc(J_{\mathsf{u/a}})$ □

*Corollary 17*

Let $P$ be a non-contradictory program. Then $WFM(P) = conc(J_{\mathsf{u/a}})$.

*Remark 3*

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

The following corollary summarises the results so far.

*Corollary 18*

The least fixpoint argumentation semantics of Dung (Dung 1993), denoted $\mathbf{J_{Dung}}$, of Prakken and Sartor (Prakken and Sartor 1997), denoted $\mathbf{J_{PS}}$, and the well-founded semantics for normal logic programs **WFS** (Bondarenko et al. 1997; van Gelder et al. 1991) and

for logic programs with explicit negation **WFSX**$_p$ (Pereira and Alferes 1992; Alferes and Pereira 1996) are related to the other least fixpoint argumentation semantics as illustrated in Figure 3.

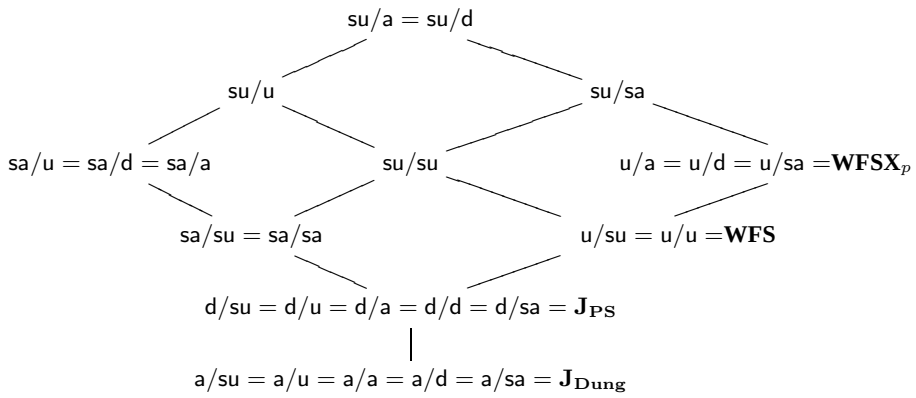


Fig. 3. Hierarchy of Notions of Justifiability and Existing Semantics

## 6 Proof Theory

One of the benefits of relating the argumentation semantics $J_{\mathsf{u/a}}$ 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 $J_{\mathsf{u/a}}$. 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 and Sartor 1997; Jakobovits and Vermeir 1999a; Kakas and 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 and 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 $move_i = (Player_i, Arg_i)(i > 0)$, such that $Player_i \in \{P, O\}$, $Arg_i \in Args_P$, and

1. $Player_i = P$ iff $i$ is odd; and $Player_i = O$ iff $i$ is even.
2. If $Player_i = Player_j = P$ and $i \neq j$, then $Arg_i \neq Arg_j$.
3. If $Player_i = P$ and $i > 1$, then $Arg_i$ is a minimal argument such that $(Arg_i, Arg_{i-1}) \in y$.
4. If $Player_i = O$, then $(Arg_i, Arg_{i-1}) \in x$.

The first condition states that the players $P$ (Proponent) and $O$ (Opponent) take turns, and $P$ starts. The second condition prevents the proponent from repeating a move. The third and fourth conditions state that both players have to attack the other player's last move, where the opponent is allowed to use the notion of attack $x$, while the proponent may use $y$ to defend its arguments. Note that the minimality condition in 3 is redundant, because *all* arguments in $Args_P$ 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 a $x/y$-dialogue, and for all moves $move_i = (P, Arg_i)$, the children of $move_i$ are all those moves $(O, Arg_j)$ such that $(Arg_j, Arg_i) \in x$.
The *height* of a dialogue tree is 0 if it consists only of the root, and otherwise $height(t) = sup\{height(t_i)\} + 1$ where $t_i$ are the trees rooted at the grandchildren of $t$.

*Example 13*
Consider the following program:

$$\begin{array}{rcl} p & \leftarrow & q, not\ r \\ q & \leftarrow & not\ s \\ \neg q & \leftarrow & u \\ r & \leftarrow & not\ t \\ s & \leftarrow & not\ t \\ t & \leftarrow & not\ w \\ u & \leftarrow & not\ v \\ v & \leftarrow & not\ r \\ \neg v & \leftarrow & not\ t \end{array}$$

A $\mathsf{a}/\mathsf{u}$-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. $(A, B) \in x$.

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]:

$$\begin{array}{rcl} p(0) & & \\ p(s(X)) & \leftarrow & not\ q(X) \\ q(X) & \leftarrow & not\ p(X) \\ r & \leftarrow & q(X) \\ s & \leftarrow & not\ r \end{array}$$

[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.

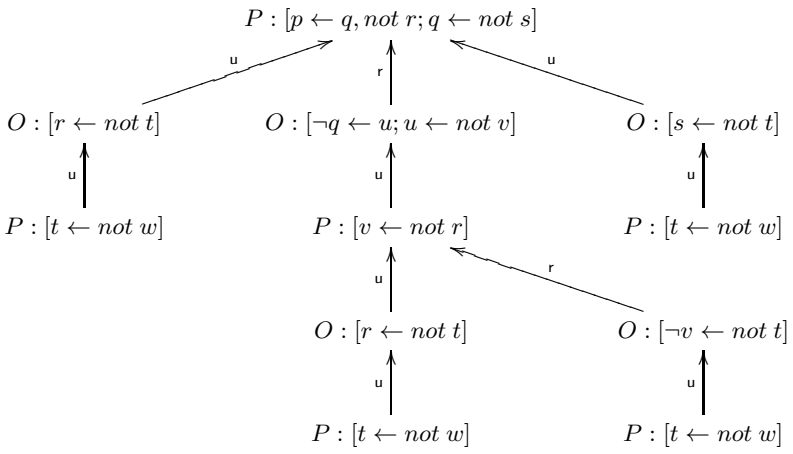


Fig. 4. An a/u-dialogue tree

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 $[p(s^n(0)) \leftarrow not\ q(s^{n-1}(0))]$ for $n > 0$. Similarly, there is exactly one minimal argument $B_n$ with conclusion $q(s^n(0))$, namely $[q(s^n(0)) \leftarrow not\ p(s^n(0))]$.
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 $C = [s \leftarrow not\ r]$. The argument $C$ is undercut by infinitely many arguments $D_n = [r \leftarrow q(s^n(0)); q(s^n(0)) \leftarrow not\ p(s^n(0))]$; 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 $height(T_n) = n$, then by Definition 17: $height(T_C) = sup\{height(T_n) \mid n \in \mathbb{N}\} + 1 = \omega + 1$.

*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 $\alpha$: $A \in J^{\alpha}_{x/y}$ 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 and 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 and 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 and 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 and 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 and 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 and Toni 1999) is developed. This proof theory formed the basis of our general proof theory for justified arguments.

In (Móra and 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 and Simari 2004; 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 and 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 and Simari 2004; 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 and 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 and 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 $L$ [5]. 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 and Alferes 1992; Alferes and 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 $\mathsf{u}/\mathsf{a}$-justified arguments, which are equivalent to WFSX$_p$ (Damásio 1996; Alferes and 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 and Jennings 1996; Sierra et al. 1997; Parsons et al. 1998; Sadri et al. 2001; Torroni 2002; Schroeder 1999; Móra and 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.

[5] See the extensive discussion in (Alferes and Pereira 1996) for details.

## 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 and Pereira 1996), and a sufficient criterion for a semantics to be consistent.
- Third, we identified an argumentation semantics $J_{\mathsf{u/a}}$ equal to the paraconsistent well-founded semantics for logic programs with explicit negation, WFSX$_p$ (Damásio 1996; Alferes and Pereira 1996) and proved this equivalence.
- Forth, we established relationships between existing semantics, in particular that $J_{Dung} \subsetneq J_{PS} \subsetneq J_{\mathsf{u/u}} = WFS \subsetneq J_{\mathsf{u/a}} = WFSX_p$, where $J_{Dung}$ and $J_{PS}$ are the least fixpoint argumentation semantics of Dung (Dung 1993) and Prakken and Sartor (Prakken and 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 and Moraitis 2002; Prakken and 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

ALFERES, J. J., DAMÁSIO, C. V., AND PEREIRA, L. M. 1995. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning 14*, 1, 93–147.

ALFERES, J. J., DUNG, P. M., AND PEREIRA, L. M. 1993. Scenario semantics for extended logic programming. In *Proceedings of the Second International Workshop on Logic Programming and Non-monotonic Reasoning (LPNMR'93)*. MIT Press, 334–348.

ALFERES, J. J. AND PEREIRA, L. M. 1996. *Reasoning with Logic Programming*. LNAI 1111, Springer-Verlag.

ANTONIOU, G. 2002. Defeasible logic with dynamic priorities. In *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, Lyon, France, 521–525.

BELNAP, N. D. 1977. A useful four-valued logic. In *Modern Uses of Many-valued Logic*, G. Epstein and J. M. Dunn, Eds. Reidel Publishing Company, 8–37.

BIRKHOFF, G. 1967. *Lattice Theory*, 3rd ed. American Mathematical Society.

BONDARENKO, A., DUNG, P., KOWALSKI, R., AND TONI, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence 93,* 1-2, 63–101.

BREWKA, G. 1996. Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research 4,* 19–36.

CHESÑEVAR, C. I., DIX, J., STOLZENBURG, F., AND SIMARI, G. R. 2002. Relating defeasible and normal logic programming through transformation properties. *Theoretical Computer Science 290,* 1, 499–529.

CHESÑEVAR, C. I., MAGUITMAN, A. G., AND LOUI, R. P. 2000. Logical models of argument. *ACM Computing Surveys 32,* 4 (December), 337–383.

CHIARIGLIONE, L. ET AL. 1997. Specification version 2.0. Tech. rep., Foundations of Intelligent Physical Agents. http://www.fipa.org.

CLARK, K. L. 1978. Negation as failure. In *Logic and Databases*, Gallaire and Minker, Eds. Plenum Press, New York, 293–322.

DAMÁSIO, C. V. 1996. Paraconsistent extended logic programming with constraints. Ph.D. thesis, Universidade Nova de Lisboa.

DAMÁSIO, C. V. AND PEREIRA, L. M. 1998. A survey on paraconsistent semantics for extended logic programs. In *Handbook of Defeasible Reasoning and Uncertainty Management*, D. M. Gabbay and P. Smets, Eds. Vol. 2. Kluwer Academic Publishers, 241–320.

DAMÁSIO, C. V., PEREIRA, L. M., AND SCHROEDER, M. 1997. REVISE: Logic programming and diagnosis. In *Proceedings of the Conference on Logic Programming and Non-monotonic Reasoning LPNMR97*. LNAI 1265, Springer–Verlag, 353–362.

DUNG, P. M. 1993. An argumentation semantics for logic programming with explicit negation. In *Proc. of the 10th International Conference on Logic Programming ICLP'93*. MIT Press, 616–630.

DUNG, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence 77,* 2, 321–357.

FININ, T., FRITZSON, R., MCKAY, D., AND MCENTIRE, R. 1994. KQML as an agent communication lanugage. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*. ACM Press, 456–463.

FREIRE, J., RAO, P., SAGONAS, K., SWITFT, T., AND WARREN, D. S. 1997. XSB: A system for efficiently computing the well-founded semantics. In *International Workshop on Logic Programming and Non-monotonic Reasoning*. 431–441.

GARCÍA, A. J. AND SIMARI, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming 4,* 1.

GARCÍA, A. J., SIMARI, G. R., AND CHESÑEVAR, C. I. 1998. An argumentative framework for reasoning with inconsistent and incomplete information. In *ECAI'98 Workshop on Practical Reasoning and Rationality*. Brighton, UK.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference on Logic Programming*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1990. Logic programs with classical negation. In *Proceedings of the 7th International Conference on Logic Programming*. MIT Press, 579–597.

JAKOBOVITS, H. AND VERMEIR, D. 1999a. Dialectic semantics for argumentation frameworks. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law (ICAIL '99)*. 53–62.

JAKOBOVITS, H. AND VERMEIR, D. 1999b. Robust semantics for argumentation frameworks. *Journal of Logic and Computation 9,* 2, 215–261.

KAKAS, A. AND TONI, F. 1999. Computing argumentation in logic programming. *Journal of Logic and Computation 9*, 4, 515–562.

KAKAS, A. C. AND MORAITIS, P. 2002. Argumentative agent deliberation, roles and context. In *Proceedings of the ICLP-Workshop Computational Logic in Multi-Agent Systems*.

KRAUS, S., SYCARA, K., AND EVENCHIK, A. 1998. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence 104*, 1-2, 1–69.

MÓRA, I. A. AND ALFERES, J. J. 1998. Argumentative and cooperative multi-agent system for extended logic programming. In *Proceedings of the 14th Brazilian Symposium on Artificial Intelligence (SBIA'98)*. 161–170.

PARSONS, S. AND JENNINGS, N. 1996. Negotiation through argumentation-a preliminary report. In *Proceedings of the Second International Conference on Multi-Agent Systems*. Kyoto, Japan, 267–274.

PARSONS, S., SIERRA, C., AND JENNINGS, N. 1998. Agents that reason and negotiate by arguing. *Journal of Logic and Computation 8*, 3, 261–292.

PEREIRA, L. M. AND ALFERES, J. J. 1992. Well founded semantics for logic programs with explicit negation. In *B. Neumann (Ed.), European Conference on Artificial Intelligence*. Wiley, 102–106.

POLLOCK, J. L. 1987. Defeasible reasoning. *Cognitive Science 11*, 481–518.

PRAKKEN, H. AND SARTOR, G. 1997. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics 7*, 1, 25–75.

PRZYMUSINSKI, T. 1990. Extended stable semantics for normal and disjunctive programs. In *Proceedings of the 7th International Conference on Logic Programming*. MIT Press, 459–477.

SADRI, F., TONI, F., AND TORRONI, P. 2001. Logic agents, dialogue, negotiation - an abductive approach. In *Proceedings of the AISB Symposium on Information Agents for E-commerce*.

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

SCHWEIMEIER, R. AND SCHROEDER, M. 2002. Notions of attack and justified arguments for extended logic programs. In *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, Lyon, France, 536–540.

SIERRA, C., JENNINGS, N., NORIEGA, P., AND PARSONS, S. 1997. A framework for argumentation-based negotiation. In *Proc. Fourth Int. Workshop on Agent Theories, Architectures and Languages (ATAL-97)*. Springer-Verlag, 167–182.

SIMARI, G. R., CHESÑEVAR, C. I., AND GARCÍA, A. J. 1994. The role of dialectics in defeasible argumentation. In *Anales de la XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación*. Universidad de Concepción, Concepción (Chile).

SIMARI, G. R. AND LOUI, R. P. 1992. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence 53*, 125–157.

TARSKI, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics 5*, 285–309.

TORRONI, P. 2002. A study on the termination of negotiation dialogues. In *Proceedings of Autonomous Agents and Multi Agent Systems 2002*. ACM Press, 1223–1230.

VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM 38*, 3 (July), 620–650.

VREESWIJK, G. A. W. 1997. Abstract argumentation systems. *Artificial Intelligence 90*, 1–2, 225–279.

WAGNER, G. 1994. *Vivid Logic – Knowledge-Based Reasoning with Two Kinds of Negation*. Vol. LNAI 764. Springer–Verlag.

## Appendix A Proofs of Theorems

*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 $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x'/y'}$, for all $\alpha$.
*Base case*: $\alpha = 0$: Then $J_{x/y} = \emptyset = J_{x'/y'}$.
*Successor ordinal*: $\alpha \rightsquigarrow \alpha + 1$:

Let $A \in J^{\alpha+1}_{x/y}$, and $(B, A) \in x'$. Then also $(B, A) \in x$, and so there exists $C \in J^{\alpha}_{x/y}$ such that $(C, B) \in y$, so also $(C, B) \in y'$. By induction hypothesis, $C \in J^{\alpha}_{x'/y'}$, and so $A \in J^{\alpha+1}_{x'/y'}$.
*Limit ordinal* $\lambda$:

Assume $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x'/y}$ for all $\alpha < \lambda$. Then
$J^{\lambda}_{x/y} = \bigcup_{\alpha<\lambda} J^{\alpha}_{x/y} \subseteq \bigcup_{\alpha<\lambda} J^{\alpha}_{x'/y'} = J^{\lambda}_{x'/y'}$ □

*Theorem 4*
Let $x$ and and $y$ be notions of attack such that $x \supseteq \mathsf{undercuts}$, and let $sy = y{-}\mathsf{undercuts}^{-1}$. 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 $\alpha$: $J^{\alpha}_{x/y} \subseteq J^{\alpha}_{x/sy}$, by transfinite induction on $\alpha$.

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

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

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

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

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

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

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

*Base case:* $\alpha = 0$

$J^{\alpha}_{x/\mathsf{a}} = \emptyset = J^{\alpha}_{x/\mathsf{u}}$.

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

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

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 $C \in J^{\alpha}_{x/\mathsf{u}}$, there exists $D \in J^{\alpha_0}_{x/\mathsf{u}} \subseteq J^{\alpha}_{x/\mathsf{u}}$ ($\alpha_0 < \alpha$) such that $D$ undercuts $B$. It follows that $A \in J^{\alpha+1}_{x/\mathsf{u}}$.

*Limit ordinal $\lambda$:*

Assume $J^{\alpha}_{x/\mathsf{a}} \subseteq J^{\alpha}_{x/\mathsf{u}}$ for all $\alpha < \lambda$. Then $J^{\lambda}_{x/\mathsf{a}} = \bigcup_{\alpha<\lambda} J^{\alpha}_{x/\mathsf{a}} \subseteq \bigcup_{\alpha<\lambda} J^{\alpha}_{x/\mathsf{u}} = J^{\lambda}_{x/\mathsf{u}}$. □

*Theorem 7*
$J_{\mathsf{sa/su}} = J_{\mathsf{sa/sa}}$

*Proof*
By Theorem 3, $J_{\mathsf{sa/su}} \subseteq J_{\mathsf{sa/sa}}$.

We prove the inverse inclusion by showing that for all ordinals $\alpha$: $J^{\alpha}_{\mathsf{sa/sa}} \subseteq J^{\alpha}_{\mathsf{sa/su}}$, by transfinite induction on $\alpha$.

*Base case:* $n = 0$

$J^{0}_{\mathsf{sa/sa}} = \emptyset = J^{0}_{\mathsf{sa/su}}$

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

Let $A \in J^{\alpha+1}_{\mathsf{sa/sa}}$, and $B$ strongly attacks $A$. By definition, there exists $C \in J^{\alpha}_{\mathsf{sa/sa}}$ such that $C$ attacks $B$ and $B$ does not undercut $C$. By induction hypothesis, $C \in J^{\alpha}_{\mathsf{sa/su}}$.

If $C$ undercuts $B$, 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 $C \in J^{\alpha}_{\mathsf{sa/su}}$ there exists $D \in J^{\alpha_0}_{\mathsf{sa/su}} \subseteq J^{\alpha}_{\mathsf{sa/su}}$ ($\alpha_0 < \alpha$) such that $D$ strongly undercuts $B$. It follows that $A \in J^{\alpha+1}_{\mathsf{sa/su}}(\emptyset)$.

*Limit ordinal $\lambda$:*

Assume $J^{\alpha}_{\mathsf{sa/sa}} \subseteq J^{\alpha}_{\mathsf{sa/su}}$ for all $\alpha < \lambda$. Then $J^{\lambda}_{\mathsf{sa/sa}} = \bigcup_{\alpha<\lambda} J^{\alpha}_{\mathsf{sa/sa}} \subseteq \bigcup_{\alpha<\lambda} J^{\alpha}_{\mathsf{sa/su}} = J^{\lambda}_{\mathsf{sa/su}}$. □

*Theorem 8*
$J_{\mathsf{su/a}} = J_{\mathsf{su/d}}$

*Proof*
By Theorem 3, $J_{\mathsf{su/d}} \subseteq J_{\mathsf{su/a}}$.

For the inverse inclusion, we show that for all ordinals $\alpha$: $J^{\alpha}_{\mathsf{su/a}} \subseteq J^{\alpha}_{\mathsf{su/d}}$, by transfinite induction on $\alpha$.

*Base case:* $\alpha = 0$

$J^0_{\mathsf{su/a}} = \emptyset = J^0_{\mathsf{su/d}}$

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

Let $A \in J^{\alpha+1}_{\mathsf{su/a}}$, and $B$ strongly undercuts $A$. By definition, there exists $C \in J^{\alpha}_{\mathsf{su/a}}$ such that $C$ undercuts or rebuts $B$. By induction hypothesis, $C \in J^{\alpha}_{\mathsf{su/d}}$.

If $C$ undercuts $B$, or $B$ does not undercut $C$, then we are done.

Otherwise, $B$ strongly undercuts $C$, and so there exists $D \in J^{\alpha_0}_{\mathsf{su/d}} \subseteq J^{\alpha}_{\mathsf{su/d}}$ ($\alpha_0 < \alpha$) such that $D$ defeats $B$. It follows that $A \in J^{\alpha+1}_{\mathsf{su/d}}$.

*Limit ordinal $\lambda$:*

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

$$J^{\lambda}_{\mathsf{su/a}} = \bigcup_{\alpha<\lambda} J^{\alpha}_{\mathsf{su/a}} \subseteq \bigcup_{\alpha<\lambda} J^{\alpha}_{\mathsf{su/d}} = J^{\lambda}_{\mathsf{su/d}}$$

□

*Lemma 15*
Let $I$ be a two-valued interpretation.

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

*Proof*

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 $L \leftarrow not\ L_1, \ldots, not\ L_n$ in $P$ s.t. $L_1, \ldots, L_n \notin I$, and $[L \leftarrow not\ L_1, \ldots, not\ L_n]$ is an argument for $L$ whose assumptions are contained in $I$.

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

   "If" direction: Induction on the length of the argument.

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

   *Induction step*: $n \rightsquigarrow n + 1$:

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

2. "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 $L \leftarrow not\ L_1, \ldots, not\ L_n$ in $P$ s.t. $\neg L, L_1, \ldots, L_n \notin I$, and $[L \leftarrow not\ L_1, \ldots, not\ L_n]$ is an argument for $L$ whose assumptions are contained in $I$, and $\neg L \notin I$.

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

   "If" direction: Induction on the length of the argument.

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

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

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

□

*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*
"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 $A$.

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

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

*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 $A$.

"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 $A \in J^{\alpha+1}_{x/y}$. □