On the Relationship Between Defeasible Logic and Well-Founded Semantics

Gerhard Brewka

University of Leipzig
Dep. of Computer Science
Augustusplatz 10-11
04109 Leipzig, Germany
brewka@informatik.uni-leipzig.de

Abstract. We investigate in this paper the relationship between an ambiguity propagating defeasible logic recently proposed by Antoniou et al. [3] and well-founded semantics with priorities [6] under a straightforward translation from defeasible theories to extended logic programs. It turns out that a slightly restricted version of defeasible logic is correct wrt well-founded semantics yet incomplete. We also investigate the sources of the incompleteness and argue that the additional conclusions obtained by prioritized well-founded semantics are indeed desired.

1 Introduction

Defeasible Logic was originally proposed by Donald Nute in 1987 [13] (for an overview see the handbook article [14]). The logic was never as prominent as, say, default logic [17] or circumscription [11]. Yet it has received considerable attention in recent years. This is at least partly due to a very active group of researchers at Griffith University which has worked on theoretical foundations, further development and implementations of defeasible logic(s) [1–3, 12].

The main advantage of defeasible logic is certainly computational: the computation of conclusions is polynomial and highly efficient implementations exist [12]. A second advantage are its built in preference handling facilities.

In the meantime several variants of Defeasible Logic have been proposed. All of them are defined proof theoretically. Defeasible logic(s) belong to a class of nonmonotonic approaches which can be called directly sceptical. By this we mean sceptical approaches where the conclusions, rather than being defined as the intersection of extensions or answer sets, are constructed directly.

In the area of logic programming well-founded semantics can be viewed as a directly sceptical semantics. It is interesting to see, therefore, what the exact relationship between these two approaches is. Since the preference handling techniques of defeasible logic have no counterpart in standard well-founded semantics the comparison will be based on an extension of well-founded semantics which was recently proposed by the author of this paper.[1]

To be precise, we will compare one of the arguably most interesting defeasible logics, an ambiguity propagating variant presented in [3], with the prioritized version of well-founded semantics for extended logic programs. This semantics was originally proposed in [6]. In this paper we use a considerably simplified version. The simplification is possible since for the purposes of this paper we do not need the ability to represent preferences in the logical language.

The major result of this paper is the correctness of the considered defeasible logic under the condition that no defeasible rule is preferred to a strict rule. We also investigate reasons for the incompleteness of the logic and argue that the additional conclusions obtained by well-founded semantics are desirable. The paper may thus be read as a critique of defeasible logic.

The rest of the paper is organized as follows. Sect. 2 describes the ambiguity propagating defeasible logic used for comparison in this paper. Sect. 3 presents a simplified version of the preferred well-founded semantics in [6]. Sect. 4 introduces the translation from defeasible logic to extended logic programs. Sect. 5 establishes the correctness result and Sect. 6 incompleteness. Sect. 7 concludes.

The analysis in the paper is performed in a propositional setting, that is, we consider propositional defeasible theories and propositional extended logic programs.

2 Defeasible Logic

Defeasible logic was first introduced by Nute [13]. It is based on strict rules of the form $A \rightarrow p$ and defeasible rules of the form $A \Rightarrow p$. In both cases A is a set of literals and p a literal. We omit set brackets whenever A is a singleton set. Facts are represented as strict rules with empty set of antecedents (in which case the arrow is left out). Nute also introduced a third type of rules called defeaters which can block the derivation of a literal without giving rise to the derivation of the complementary literal. In [2] it is shown that defeaters are not essential in the sense that they can be simulated by the other rules. We will therefore not discuss defeaters in this paper.

To solve conflicts among rules Nute used a preference relation $>$ among rules: $r > r'$ intuitively stands for: r has higher priority than r'. The preference relation is required to be acyclic, i.e. its transitive closure must be irreflexive.

Nute's original defeasible logic was not ambiguity propagating. Consider the following example:

Example 1:
1) $\Rightarrow p$
2) $\Rightarrow \neg p$

[1] Although numerous prioritized version of logic programs under stable model or answer set semantics exist (see [7] for a discussion of some of these approaches) not much work has been done on prioritizing well-founded semantics.

3) $\Rightarrow q$
4) $p \Rightarrow \neg q$

Since p is not defeasibly provable (because of the conflicting second rule) rule 4) is disregarded and q is defeasibly provable in Nute's logic. This seems highly questionable since, although p is not accepted, there is an argument supporting $\neg q$ which should not be disregarded in a sceptical approach.

For this reason Antoniou and colleagues [3] defined an "ambiguity propagating defeasible logic without team defeat" which behaves as desired in the example. We consider this logic as one of the most interesting variants of defeasible logic and use it for our comparison in this paper.

A defeasible theory is a pair $T = (R, >)$ where R is a finite set of strict and defeasible rules and $>$ is the preference relation among R. A conclusion of T is a tagged literal: $+\Delta q$ means q is strictly provable, $+\delta q$ means q is defeasibly provable, and $+\sigma q$ means q is supported.[2] The tags preceded by minus-signs stand for corresponding negated expressions. A proof is a finite sequence P of tagged literals. $P(i)$ denotes the i-th element in the sequence, $P(1..i)$ its initial segment of length i. The complement of a literal q is denoted $-q$. $R[q]$ is the set of rules with head q, $R_s[q]$ the subset of $R[q]$ consisting of all strict rules. The antecedents of a rule r are denoted $A(r)$.

Inference rules are phrased as conditions on proofs as follows:[3]

$+\Delta$: If $P(i+1) = +\Delta q$ then
    $\exists r \in R_s[q]\ \forall a \in A(r) : +\Delta a \in P(1..i)$.

$-\Delta$: If $P(i+1) = -\Delta q$ then
    $\forall r \in R_s[q]\ \exists a \in A(r) : -\Delta a \in P(1..i)$.

$+\delta$: If $P(i+1) = +\delta q$ then
    (1) $+\Delta q \in P(1..i)$ or
    (2.1) $-\Delta - q \in P(1..i)$ and
    (2.2) $\exists r \in R[q]$ such that
        $\forall a \in A[r] : +\delta a \in P(1..i)$ and
        $\forall s \in R[-q]$:
            $\exists a \in A[s] : -\sigma a \in P(1..i)$ or
            $r > s$.

$-\delta$: If $P(i+1) = -\delta q$ then
    (1) $-\Delta q \in P(1..i)$ and
    (2.1) $+\Delta - q \in P(1..i)$ or
    (2.2) $\forall r \in R[q]$:
        $\exists a \in A[r] : -\delta a \in P(1..i)$ or
        $\exists s \in R[-q]$ such that
            $\forall a \in A[s] : +\sigma a \in P(1..i)$ and
            $r \not> s$.

[2] We use σ here rather than the less readable and less mnemonic symbol used in [3].
[3] The rule for $+\sigma$ in [3] had a mistake in the last line (G. Antoniou, personal communication) which is corrected here.

$+\sigma$: If $P(i+1) = +\sigma q$ then
$+\Delta q \in P(1..i)$ or
$\exists r \in R[q]$ such that
$\forall a \in A[r] : +\sigma a \in P(1..i)$ and
$\forall s \in R[-q]$:
$\exists a \in A[s] : -\delta a \in P(1..i)$ or
$s \not> r$.

$-\sigma$: If $P(i+1) = -\sigma q$ then
$-\Delta q \in P(1..i)$ and
$\forall r \in R[q]$:
$\exists a \in A[r] : -\sigma a \in P(1..i)$ or
$\exists s \in R[-q]$ such that
$\forall a \in A[s] : +\delta a \in P(1..i)$ and
$s > r$.

Consider the following example:

Example 2:
1) p
2) $p \rightarrow q$
3) $q \Rightarrow r$
4) $\Rightarrow \neg$ r

Assume there are no priorities. Here is a proof for $+\sigma r$:

$$+\Delta p, +\Delta q, +\sigma q, +\sigma r.$$

If we add the preference $4 > 3$ the last step in the proof does not go through. Indeed, we now have the following proof for $+\delta \neg r$:

$$-\Delta r, +\delta \neg r.$$

3 Prioritized well-founded semantics

In this section we present a simplified version of the well-founded semantics for extended logic programs with priorities which was defined in [6]. The simplification is possible because we are not interested here in expressing preference information in the logical language (for a discussion why this may be useful see [6]).

A (propositional) extended logic program is a finite set of rules of the form

$$c \leftarrow a_1, \ldots, a_n, not \ \ b_1, \ldots, not \ \ b_m$$

where the a_i, b_j and c are propositional literals, i.e., either propositional atoms or such atoms preceded by the classical negation sign. The symbol *not* denotes negation by failure (default negation), $\neg$ denotes strong negation. An extended logic program is a finite set P of rules. A prioritzed logic program is a pair $(P, >)$ where P is an extended logic program and $>$ an acyclic preference relation on P: as in defeasible logic $r > r'$ stands for r is preferred over r'. In [6] $>$ was required to be transitive. This restriction is not necessary and dropped here for the purpose of comparison. Note also that in the earlier paper the "smaller" rules were preferred rather than the "bigger" rules as in this paper.

Well-founded semantics is an inherently sceptical semantics that refrains from drawing conclusions whenever there is a potential conflict. Its original formulation for general logic programs by Gelder, Ross and Schlipf [9] is based on a certain partial model. Przymusinski reconstructed this definition in 3-valued logic [15, 16]. A reformulation based on the least fixed point of a monotone operator, namely the twofold application of the Gelfond/Lifschitz γ-operator [8], was first given by Baral and Subrahmanian [5]. The straightforward extension of this formulation to extended logic programs that underlies our approach was used by several authors, e.g. [4, 10].

Let us first introduce the γ-operator. We say a rule r of the form above is defeated by a literal l if $l = b_i$ for some $i \in \{1, \dots, m\}$. We say r is defeated by a set of literals X if X contains a literal that defeats r.

Let P be a logic program, X a set of literals. The X-reduct of P, denoted P^X, is the program obtained from P by deleting each rule defeated by X. For a set of rules R, the closure $Cl(R)$ is the smallest set of literals closed under R and the consequences $Cn(R)$ the smallest set of literals that is (1) closed under R, and (2) logically closed, i.e., either consistent or equal to the set of all literals. For the computation of the closure we simply neglect default negated literals.

The Gelfond/Lifschitz operator γ_P now is defined as follows:

$$\gamma_P(X) = Cn(P^X)$$

For normal logic programs (that is programs without strong negation) the atoms true according to well-founded semantics are just the least fixed point of the twofold application of γ. It was argued in [6] that for the extension of well-founded semantics to extended logic programs with two kinds of negation it is favourable to slightly modify the fixed point operator: rather than computing the least fixed point of γ^2 Brewka proposed to compute the least fixed point of $\gamma\gamma^*$ where γ^* rather than yielding the consequences $Cn(P^X)$ yields the closure $Cl(P^X)$. This leads to a larger set of well-founded conclusions without violating correctness wrt answer set semantics.

The intuition behind well-founded semantics can be described as follows: given a set of literals S already known to be derivable, $\gamma^*(S)$ produces a set of potential conclusions which still might defeat rules in P. The conclusions of rules not defeated by any of the potential defeaters are clearly derivable. Starting with the empty set, we thus generate larger and larger sets S until a fixed point is reached. The following terminology reflects this intuition:

Definition 1. *Let P be an extended logic program.*

– *A literal l is an S-potential defeater iff l is in the closure of the rules in P not defeated by S.*

– *A rule r is S-undefeatable iff r is not defeated by any S-potential defeater.*
– *A literal l is S-derivable iff l is a consequence of S-undefeatable rules in P.*

It is obvious that l is S-derivable iff $l \in \gamma(\gamma^*(S))$. The least fixed point of $\gamma\gamma*$ is called $WFS(P)$, or simply WFS if P is clear from context.

To take preferences into account we first introduce a notion of dominance. Intuitively, a rule r dominates a rule r' in the context of a set of literals S if r has higher priority and if the application of r in context S actually defeats r'. As pointed out in [6] the second condition is necessary to guarantee that prioritized well-founded semantics is an extension of well-founded semantics. Here is the formal definition

Definition 2. *Let r and r' be rules, S a set of literals. We say r S-dominates r' iff*

1. *$r > r'$, and*
2. *$Cl(\{r\} \cup \{s : s \text{ is } S\text{-undefeatable}\})$ defeats r'.*

For the case of prioritized programs Def. 1 becomes

Definition 3. *Let $(P, >)$ be a prioritized logic program.*

– *A literal l is an S-potential r-defeater iff l is in the closure of rules in P which are (1) not defeated by S and (2) not S-dominated by r.*
– *A rule r is S-safe iff r is not defeated by any S-potential r-defeater.*
– *A literal l is S-derivable iff l is a consequence of the set of S-safe rules in P.*

The definition for prioritized logic programs is different from the one for non-prioritized programs in two respects. Firstly, there is not a single set of potential defeaters for all rules but each rule r has its own set of potential defeaters. Secondly, the rules which are used to derive potential defeaters must satisfy an additional condition: to potentially defeat r a rule must not be dominated by r in context S. Since fewer rules can be used to derive potential defeaters for a rule the safe rules are a superset of the undefeatable rules. We thus obtain more derivable literals. For the special case where $>$ is empty the two definitions of S-derivable clearly coincide.

The set of S-derivable literals grows monotonically with S. We thus can start as usual with the empty set of literals and iterate the computation of S-derivable formulas until a fixed point is reached.

Here is a small example illustrating the definition.

Example 3:
1) $c \leftarrow not\ \neg c, a$
2) $\neg c \leftarrow not\ c$
3) a

Let $1 > 2$. Clearly, rule 3 is $\emptyset$-safe since there is no way of defeating a rule without default negation. But also 1) is $\emptyset$-safe since the closure of 1 together with the $\emptyset$-undefeatable rule 3 defeats 2 and thus 1 $\emptyset$-dominates 2. Therefore the set of $\emptyset$-derivable literals is $\{c, a\}$. This set is already the least fixed point.

4 The translation

We use a straightforward modular translation $Trans$ from defeasible theories $T = (R, >)$ to extended logic programs $Trans(T) = (Trans(R), >')$ where $Trans(R) = \{Trans(r) : r \in R\}$ and the translation of each rule is defined as follows:

$\{a_1, \ldots, a_n\} \rightarrow b$	becomes	$b \leftarrow a_1, \ldots, a_n$
$\{a_1, \ldots, a_n\} \Rightarrow b$	becomes	$b \leftarrow not\ -b, a_1, \ldots, a_n$

Furthermore, we require that $Trans(r) >' Trans(r')$ iff $r > r'$. (in the rest of the paper we use the same symbol for the two preference relations because we don't expect this to cause any confusion).

The prioritized logic programs obtained this way are a proper subset of prioritized extended logic programs which we call defeasible logic programs. Defeasible logic programs use default negation in a highly restricted way (corresponding to normal defaults in default logic).

5 (In)correctness

We first investigate correctness of defeasible logic wrt prioritized well-founded semantics, that is, the question whether for each defeasible conclusion $+\delta p$ of a defeasible theory T we have $p \in WFS(Trans(T))$. The answer for the general case will be no, but for a somewhat restricted case correctness can be established.

The negative answer for the general case can be demonstrated by the following counterexample (we put the defeasible logic rules and their translation into the same line):

Example 4:

1)	$\Rightarrow p$	$p \leftarrow not\ \neg p$
2)	$p \rightarrow q$	$q \leftarrow p$
3)	$\Rightarrow \neg q$	$\neg q \leftarrow not\ q$

Assume $3 > 2$. Now $+\delta\neg q$ is a conclusion which can be established through the following derivation:

$$-\Delta p, -\Delta q, +\delta\neg q$$

Well-founded semantics, on the other hand, concludes q but not $\neg q$: although 3 has higher priority than 2 it does not dominate 2 since a strict rule can never be defeated. From this we have the following proposition:

Proposition 1. *Defeasible logic is incorrect wrt prioritized well-founded semantics: there is a defeasible theory $T = (R, >)$ and a formula q such that $+\delta q$ is a consequence of T but $q \notin WFS(Trans(T))$.*

The example already hints at the source of the incorrectness. In defeasible logic a strict rule can be overridden by a defeasible rule with higher priority. This can never happen in well-founded semantics where the conclusion of a strict rule is accepted whenever its antecedents are accepted no matter what the preferences are. Indeed, a restriction on the admissable preferences turns out to be sufficient for obtaining correctness.

Proposition 2. *Let $T = (R, >)$ be a defeasible theory such that $>$ is defined on defeasible rules only, $Trans(T) = (P, >)$ its translation. If $+\delta q$ is a conclusion of T then $q \in WFS(Trans(T))$.*

Proof. For the proof we show the following lemmata: Let $S = S_0 \cup S_1 \cup \ldots$ be the least fixed point reached for $Trans(T)$, that is, $S_0 = \emptyset$ and for all $i > 0$, S_i is the set of S_{i-1}-derivable literals. Then the following results hold:

Lemma 1. *If $+\Delta q$ is a conclusion of T, then $q \in S_1$ (and henceforth in all S_j for $j \geq 1$).*

Lemma 2. *If $+\delta q$ is a conclusion of T, then $q \in S_i$ for some i (and henceforth in all S_j for $j \geq i$).*

Lemma 3. *If $-\sigma q$ is a conclusion of T, then q is not an S-potential defeater, that is there is an i such that q is not an S_i- potential defeater (and henceforth not an S_j-potential defeater for all $j \geq i$).*

Note that our proposition is equivalent to Lemma 2. Lemma 1 is immediate since strict rules can never be defeated.

The proof for Lemmas 2 and 3 is by joint induction on the length n of the shortest proof of the corresponding tagged literals. The base case can be checked easily. For the inductive step we assume that Lemmas 2 and 3 hold for tagged literals whose shortest proofs are of length at most $n-1$. We have to distinguish 2 cases representing the possible tagged literals appearing in the lemmata:

case $+\delta q$:
there are two alternatives
a) $+\Delta q$ appears in the proof before $+\delta q$, then according to Lemma 1, q is already in S_1 and thus in S, or
b) there is a rule r with head q whose antecedents are, by induction hypothesis, in S_j for some j, and for all conflicting rules r': an antecedent is, by induction hypothesis, not an S_k-potential defeater, for some k, or r' is S_j dominated by r. Hence r is S_m-safe for sufficiently large m and thus $q \in S$. Note that for domination to hold we need the fact that r' is a defeasible rule, otherwise r could not dominate r'.

case $-\sigma q$:
we know that for each rule r with head q one of the following 2 alternatives holds:
a) there is an antecedent which is, by induction hypothesis, not an S_j-potential defeater for some j, or

b) there is a conflicting rule s whose antecedents are, by induction hypothesis, already in S_i, for some i, and which S_i-dominates r (since $s > r$ rule r must be defeasible and thus domination follows from the fact that the rules have complementary heads).

Let k be the smallest integer such that for each rule r with head q an antecedent of r is not an S_k-potential defeater (case a) or r is S_k-dominated by a rule with complementary head (case b). Let D be the set of rules with head q for which case b) holds but not case a). There are two possibilities: if D is empty then q is not a potential S_k-defeater. If D is not empty then, since $>$ is acyclic, there must be a rule r' among the rules S_k-dominating elements of D which is itself not S_k-dominated and thus S_k-safe. S_{k+1} therefore contains $-q$, all rules for which case b) holds are thus S_{k+1}-defeated and q is not an S_{k+1}-potential conclusion.

6 Incompleteness

We next discuss completeness. It turns out that defeasible logic is incomplete wrt prioritized well-founded semantics:

Proposition 3. *Let $T = (R, >)$ be a defeasible theory, and $Trans(T) = (P, >)$ its translation. $q \in WFS(Trans(T))$ does not imply that $+\delta q$ is a conclusion of T.*

To prove the proposition we will discuss some counterexamples which also illustrate the sources of the incompleteness.

Here is a first counterexample:

Example 5:

1)	$\Rightarrow \neg p$	$\neg p \leftarrow not\ p$
2)	$p \Rightarrow p$	$p \leftarrow p$

Assume there are no preferences. There is no proof for $+\delta\neg p$. Although the conflicting rule 2 can never be used to derive p, the mere existence of the rule is regarded as sufficient reason not to conclude $\neg p$. Well-founded semantics, on the other hand, concludes $\neg p$: p is not a potential $\emptyset$-defeater, 1 is thus $\emptyset$-undefeatable and used to derive $\neg p$. Well-founded semantics thus implicitly performs the kind of loop checking which is lacking in defeasible logic.

For the next counterexample consider again the rules of Example 4:

Example 6:

1)	$\Rightarrow p$	$p \leftarrow not\ \neg p$
2)	$p \rightarrow q$	$q \leftarrow p$
3)	$\Rightarrow \neg q$	$\neg q \leftarrow not\ q$

This time we assume no priorities. Clearly $q \in WFS(Trans(T))$ but $+\delta q$ is not a conclusion of T. This illustrates a major difference in the way strict rules are treated. In well-founded semantics a strict rule is applied whenever its antecedents are accepted, independently of whether the antecedents are derived using strict or defeasible rules. In defeasible logic strict rules have a different role, depending on whether all antecedents have strict proofs or not. If this is the case, then the rule is applied. If one of the antecedents is only defeasibly derivable then the strict rule is treated like a defeasible rule and may be blocked by a conflicting defeasible rule, as in our example.

Is this ambivalent role of rules adequate? In other words, is well-founded semantics sometimes not cautious enough? We do not think so. As the authors of [3] point out "strict rules are intended to define relationships that are definitional in nature". The example they give is $emu(X) \rightarrow bird(X)$. If there is a definitional relationship between emus and birds it seems fully adequate to accept, say $bird(Tweety)$ if $emu(Tweety)$ is accepted. The additional conclusions obtained by well-founded semantics in examples like the one above seem perfectly reasonable.

The next example shows that well-founded semantics takes more preferences into account than defeasible logic.

Example 7:

1)	$\Rightarrow p$	$p \leftarrow not \neg p$
2)	$p \rightarrow \neg q$	$\neg q \leftarrow p$
3)	$\Rightarrow q$	$q \leftarrow not \neg q$
4)	$q \rightarrow \neg p$	$\neg p \leftarrow q$

Assume $1 > 3$. Defeasible logic does not conclude $+\delta p$. The reason is that only preferences of rules with complementary heads play a role in the proof theory of defeasible logic. Preferences among other rules are simply disregarded. Well-founded semantics, on the other hand, concludes p in the example. Rule 1 is $\emptyset$-safe since the closure of 1 together with the $\emptyset$-undefeatable rule 2 defeats 3, that is 1 $\emptyset$-dominates 3.

Again we believe that the additional conclusions obtained by well-founded semantics are perfectly reasonable.

7 Conclusions

In this paper we have analyzed the relationship between defeasible logic and well-founded semantics for prioritized extended logic programs with two types of negation. For the comparison we used a straightforward modular translation of defeasible theories to extended logic programs. The analysis was based on the arguably most attractive variant of defeasible logic, the ambiguity propagating defeasible logic presented in [3]. The prioritized well-founded semantics we used is a considerably simplified version of a semantics proposed in [6]. The simplification was possible since for the purpose of this paper the ability to represent preference information in the logical language was not essential.

It turned out that, although correctness does not hold in general, a minor restriction is sufficient to guarantee correctness: if we admit preferences between defeasible rules only, then all defeasibly provable literals are true in prioritized well-founded semantics. It should be mentioned that the use of the ambiguity propagating variant of defeasible logic without team defeat clearly is essential for this result. Nute's original version is obviously incorrect (see Example 1) as any other variant without ambiguity propagation.

We also analyzed the sources of the incompleteness of defeasible logic. It turned out that three factors contribute to the incompleteness: 1) the lack of loop checking, 2) the somewhat ambivalent role of strict rules which - so to speak - turn into defeasible rules if not all antecedents are strictly provable, and 3) the preference handling which completely neglects preferences between rules which do not have complementary literals.

From a semantical point of view well-founded semantics seems to have clear advantages: the additional conclusions obtained seem perfectly reasonable. Moreover, in comparison with the complex rules of defeasible logic the definition of well-founded semantics is quite simple and elegant. Finally, the semantics is defined for a much larger class of programs than those obtained by translating defeasible theories.

What remains is the computational aspect. In both approaches the computation of conclusions is polynomial in the size of the rule base. In [6] the variant of well-founded semantics where preference information is expressed in the language is reported to be of cubic complexity. In [12] Nute's defeasible logic is reported to be of linear complexity. It remains an issue for further study how these results transfer to the variants discussed in this paper, and whether there are applications where a possible computational advantage of defeasible logic can outweigh its semantical disadvantages.

Acknowledgements

I would like to thank G. Antoniou for several helpful discussions on defeasible logic.

References

1. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., A Flexible Framework for Defeasible Logics, Proc. 17th American National Conference on Artificial Intelligence, AAAI-2000, 405-410
2. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Representation Results for Defeasible Logics, *ACM Transactions on Computational Logic*, in print
3. Antoniou, G., Billington, D., Governatori, G., Maher, M.J., Rock, A., A Family of Defeasible Reasoning Logics and its Implementation, Proc. European Conference on Artificial Intelligence ECAI-2000, 459-463
4. Baral, C., Gelfond, M., Logic Programming and Knowledge Representation,*Journal of Logic Programming*, 19,20:73-148, 1994
5. Baral, C., Subrahmanian, V.S., Duality between Alternative Semantics of Logic Programs and Nonmonotonic Formalisms, Intl. Workshop on Logic Programming and Nonmonotonic Reasoning, 1991

6. Brewka, G., Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996
7. Brewka, G., Eiter, T., Preferred Answer Sets for Extended Logic Programs. *Artificial Intelligence* 109, 297–356, 1999
8. Gelfond, M., Lifschitz, V., Logic Programs with Classical Negation, Proc. 7th Intl. Conference on Logic Programming, 1990
9. van Gelder, A., Ross, K., Schlipf, J., The Well-Founded Semantics for General Logic Programs, *Journal of ACM*, 221-230, 1990
10. Lifschitz, V., Foundations of Declarative Logic Programming, in: G. Brewka (ed.), Principles of Knowledge Representation, Studies in Logic, Language and Information, CSLI publications, 1996
11. McCarthy, J., Circumscription – A Form of Nonmonotonic Reasoning, *Artificial Intelligence* 13, 1980
12. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T., Efficient Defeasible Reasoning Systems, Proc. 12th IEEE International Conference on Tools with Artificial Intelligence, ICTAI-2000, 384-392
13. Nute, D., Defeasible Reasoning, in Proc. 20th Hawaii International Conference on on Systems Science, IEEE Press, 1987, 470-477
14. Nute, D., Defeasible Logic, in D.M. Gabbay, C.J. Hogger, J.A. Robinson (eds.) Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Oxford University Press, 1994, 353-395
15. Przymusinski, T., The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics, *Fundamenta Informaticae*, 1989
16. Przymusinski, T., Stable Semantics for Disjunctive Programs, *New Generation Computing*, 9:401-424, 1991
17. Reiter, R., A Logic for Default Reasoning, *Artificial Intelligence* 13 (1980) 81-132