TLP **17** (5–6): 819–836, 2017. © Cambridge University Press 2017
doi:10.1017/S1471068417000266 First published online 22 August 2017

Annotated defeasible logic

GUIDO GOVERNATORI
Data61, CSIRO, Australia, Brisbane
(*e-mail:* guido.governatori@data61.csiro.au)

MICHAEL J. MAHER
Reasoning Research Institute, Australia, Canberra
(*e-mail:* michael.maher@reasoning.org.au)

submitted 16 June 2017; revised 20 June 2017; accepted 23 June 2017

Abstract

Defeasible logics provide several linguistic features to support the expression of defeasible knowledge. There is also a wide variety of such logics, expressing different intuitions about defeasible reasoning. However, the logics can only combine in trivial ways. This limits their usefulness in contexts where different intuitions are at play in different aspects of a problem. In particular, in some legal settings, different actors have different burdens of proof, which might be expressed as reasoning in different defeasible logics. In this paper, we introduce annotated defeasible logic as a flexible formalism permitting multiple forms of defeasibility, and establish some properties of the formalism.

KEYWORDS: defeasible logic, non-monotonic reasoning, annotated logics, legal reasoning

1 Introduction

In some application domains, for example, legal reasoning, knowing that something holds (or it is presumed to hold) is not enough to draw further conclusions from it. One has to determine to what degree one can assert that it holds. In other words, statements in rules (here we use the term "rule" to indicate a mechanism/principle to assert conclusions from already established assertions) have an associated proof standard. Accordingly, a party wanting to assert a particular assertion has the burden to prove that assertion with the appropriate standard (or a stronger one). Consider the following rule:

$$IllegalBehaviour, \neg Justification \Rightarrow Liability$$

Suppose there is factual evidence about the illegal behaviour. The information in the rule is not enough, since it does not prescribe the burden needed to assess whether the behaviour was justified or not. According to Prakken and Sartor (2007) and Governatori and Sartor (2010), in a civil case, the lack of justification is subject to the so-called *burden of production*, i.e., there is a credible argument for it, while in a criminal case, the *burden of persuasion* applies (i.e., more sceptical reasoning must be used).

Let us consider a concrete scenario. Party A caused some injuries to B. Party A was much stronger than Party B, and thus the action causing injury is not justified. On the other hand, Party A claims that they acted in self-defence since they were under threat from Party B. The scenario can now be modelled by the following rules:

$$\begin{aligned} Injury, \neg Justification &\Rightarrow Liability \\ Threat &\Rightarrow Justification \\ Stronger &\Rightarrow \neg Justification \end{aligned}$$

Here, in case we are not able to assess whether the threat was real, we have a credible argument for $\neg Justification$ (because Party A is stronger), but we do not have a sceptical argument for it (because it might be that the threat was real, and then the outcome from the two conflicting rules is undetermined). Thus, we can establish liability in a civil case, but Party A is not criminally liable. Accordingly, we can reformulate the initial rule in the following two principles:

$$\begin{aligned} Tort, BurdenProduction(\neg Justification) &\Rightarrow CivilCaseLiability \\ Crime, BurdenPersuasion(\neg Justification) &\Rightarrow CriminalCaseLiability \end{aligned}$$

where *BurdenProduction* and *BurdenPersuasion* are annotations describing the mode in which we have to prove the lack of justification for the illegal behaviour.

Legal reasoning has developed so-called *proof standards* (e.g., scintilla of evidence, substantial evidence, preponderance of evidence, beyond reasonable doubt) according to which assertions have to be justified. Gordon and Walton (2009) proposed to encode proof standards using rule-based argumentation with salience, and Governatori (2011) shows how to represent the proof standards of Gordon and Walton (2009), where, essentially, each proof standard corresponds to a different degree of provability in some defeasible logic variant. In particular, Governatori (2011) argues that the proof standard of beyond reasonable doubt corresponds to provability in the ambiguity propagating variant of defeasible logic. However, as the following example illustrates, there are examples where more than one such proof standards must be used. This means that incompatible variants of defeasible logic have to work side-by-side.

Suppose that a piece of evidence A suggests that the defendant in a legal case is not responsible while a second piece of evidence B indicates that he/she is responsible; moreover, the sources are equally reliable. According to the underlying legal system, a defendant is presumed innocent (i.e., not guilty) unless responsibility has been proved (beyond reasonable doubt).

The above scenario is encoded by the following rules:

$$\begin{aligned} r_1 &: EvidenceA \Rightarrow \neg Responsible \\ r_2 &: EvidenceB \Rightarrow Responsible \end{aligned} \qquad \begin{aligned} r_3 &: Responsible \Rightarrow Guilty \\ r_4 &: \Rightarrow \neg Guilty \end{aligned}$$

where r_3 is stronger than r_4. Given both *EvidenceA* and *EvidenceB*, the literal *Responsible* is ambiguous. There are applicable rules (r_1 and r_2) for and against the literal, with no way to adjudicate between them. As a consequence, r_3 is not applicable, and so there is no applicable rule arguing against the presumption of

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

innocence (rule r_4). In an ambiguity blocking setting, we obtain a $\neg Guilty$ verdict; the ambiguity about responsibility is blocked from applying to *Guilty*. In contrast, in an ambiguity propagating setting, the ambiguity of *Responsible* propagates to *Guilty*, and thus the literals *Guilty* and $\neg Guilty$ are ambiguous too; hence, an undisputed conclusion cannot be drawn. When we look at the example above, is it appropriate to say that we have reached a not guilty verdict without any reasonable doubt? The evidence supporting that the defendant was responsible has not been refuted. This example supports the contention of Governatori (2011) that ambiguity propagating inference is a more appropriate representation of proof beyond a reasonable doubt.

Let us extend the scenario. Suppose that the legal system allows for compensation for wrongly accused people. A person (defendant) has been wrongly accused if the defendant is found innocent, where innocent is defined as $\neg Guilty$. In addition, by default, people are not entitled to compensation. The additional elements of this scenario are modelled by the following rules:

$$\begin{aligned} r_5 &: \neg Guilty \Rightarrow \quad Compensation \\ r_6 &: \qquad\quad \Rightarrow \neg Compensation \end{aligned}$$

where r_5 is stronger than r_6.

In the full scenario, the defendant is not found innocent, and so is not entitled to compensation.

If we take a purely ambiguity blocking stance, then, since we are not able to determine whether there was responsibility, the defendant is not guilty, and then the defendant is entitled to compensation. On the other hand, in a purely ambiguity propagating setting, *Guilty* and $\neg Guilty$ are ambiguous, and this makes *Compensation* and $\neg Compensation$ ambiguous; we are in a position where we cannot decide whether the defendant is entitled or not to compensation. Thus, both choices are unsatisfactory: either the defendant receives compensation despite not being found innocent or no decision is made about compensation.

What we want is a regime where we can reason about guilt in an ambiguity propagating way, but then reason about compensation in an ambiguity blocking way. This can be achieved by replacing rule r_5 with

$$r_5' : BeyondReasonableDoubt(\neg Guitly) \Rightarrow Compensation$$

where similarly to what we have done in the previous example, *BeyondReasonable Doubt* is an annotation to the literal $\neg Guilty$ that holds in case the literal is provable under ambiguity propagation, and the proof standard for *Compensation* can be chosen to be ambiguity blocking.

The purpose of this paper is to provide a formalism – *annotated defeasible logic* – in which such distinctions can be expressed, define its semantics, and investigate properties of the formalism.

This paper is organised as follows. In the next section, we provide brief background on defeasible logics. We then introduce annotated defeasible logic, and define its behaviour with a meta-program. In the following section, we establish some properties of annotated defeasible logic, including its relationship to existing defeasible logics and the relative inference strength of the additional inference rules we introduce.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

Finally, we show that annotated defeasible logic has the flexibility to deal with different notions of failure, corresponding to different semantics of negation-as-failure in logic programs. Due to space limitations, parts of the paper – including proof sketches – are presented in the supplementary material accompanying the paper at the TPLP archive.

2 Defeasible logics

In this section, we can only present an outline of defeasible logics. Further details can be obtained from the paper by Billington *et al.* (2010) and the references therein. We address propositional defeasible logics, but many results should extend to a first-order language.

A defeasible theory is built from a language Σ of literals (which we assume is closed under negation) and a language Λ of labels. A *defeasible theory* $D = (F, R, >)$ consists of a set of facts F, a finite set of rules R, each rule with a distinct label from Λ, and an acyclic relation $>$ on Λ called the *superiority relation*. This syntax is uniform for all the logics considered here. Facts are individual literals expressing indisputable truths. Rules relate a set of literals (the body), via an arrow, to a literal (the head), and are one of three types: a strict rule, with arrow $\rightarrow$; a defeasible rule, with arrow $\Rightarrow$; or a defeater, with arrow $\rightsquigarrow$. Strict rules represent inferences that are unequivocally sound if based on definite knowledge; defeasible rules represent inferences that are generally sound. Inferences suggested by a defeasible rule may fail, due to the presence in the theory of other rules. Defeaters do not support inferences, but may impede inferences suggested by other rules. The superiority relation provides a local priority on rules with conflicting heads. Strict or defeasible rules whose bodies are established defeasibly represent claims for the head of the rule to be concluded. When both a literal and its negation are claimed, the superiority relation contributes to the adjudication of these conflicting claims by an inference rule, leading (possibly) to a conclusion.

Defeasible logics derive conclusions that are outside the syntax of the theories. Conclusions may have the form $+dq$, which denotes that under the inference rule d, the literal q can be concluded, or $-dq$, which denotes that the logic can establish that under the inference rule d the literal q cannot be concluded. The syntactic element d is called a proof tag. In general, neither conclusion may be derivable: q cannot be concluded under d, but the logic is unable to establish that. Tags $+\Delta$ and $-\Delta$ represent monotonic provability (and unprovability) where inference is based on facts, strict rules, and modus ponens. We assume these tags and their inference rules are present in every defeasible logic. What distinguishes a logic is the inference rules for defeasible reasoning. The four logics discussed by Billington *et al.* (2010) correspond to four different pairs of inference rules, tagged ∂, δ, ∂^*, and δ^*; they produce conclusions of the form (respectively) $+\partial q$, $-\partial q$, $+\delta q$, $-\delta q$, etc., where q is a literal. These logics all abide by the Principle of Strong Negation (Antoniou *et al.* 2000), which asserts that the condition for applying a $-d$ inference rule should be the strong negation of the condition for applying $+d$. The inference rules δ and

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

δ^* require auxiliary tags and inference rules, denoted by σ_δ and σ_{δ^*}, respectively[1], expressing that there is at least (weak) support for the conclusion. These inference rules are available in the supplementary material. For each of the four principal defeasible tags d, the corresponding logic is denoted by $\mathbf{DL}(d)$. We write $D \vdash +dq$ (respectively, $D \vdash -dq$) if $+dq$ ($-dq$) can be proved by $\mathbf{DL}(d)$.

The four principal tags and corresponding inference rules represent different intuitions about defeasible reasoning, that is, define different forms of defeasibility: in ∂ and ∂^*, ambiguity is blocked, while in δ and δ^*, ambiguity is propagated; in ∂ and δ, rules for a literal act as a team to overcome competing rules, while in ∂^* and δ^*, an individual rule must overcome all competing rules. The scenario in the introduction with rules $r_1 - r_4$ exemplifies the treatments of ambiguity. For an example of team defeat, consider rules s_1 and s_2 for q and rules s_3 and s_4 for $\neg q$, with $s_1 > s_3$ and $s_2 > s_4$; then no individual rule for q can overcome the rules for $\neg q$, but s_1 and s_2 – as a team – can, because every rule for $\neg q$ is overridden by some rule in the team. A more detailed discussion of ambiguity and team defeat in the **DL** framework is given by Billington *et al.* (2010) and Maher (2012).

In the papers by Maher and Governatori (1999) and Antoniou *et al.* (2000), the inference rules in $\mathbf{DL}(d)$ were reformulated as a meta-program $\mathscr{M}_d$: a logic program that takes a representation of a defeasible theory D as input and specifies what conclusions can be drawn from the theory according to the d inference rules. (The combined meta-program and theory is denoted by $\mathscr{M}_d(D)$.) We will take this meta-programming formulation as our starting point, rather than the inference rules as presented by Billington *et al.* (2010), for example. This meta-program formulation is given in the supplementary material. We assume, initially, that the logic programming semantics in use is Kunen's semantics (Kunen 1987), which expresses the three-valued logical consequences of the Clark completion of a logic program. Equivalently, Kunen's semantics is the set of all consequences of $\Phi \uparrow n$ for any finite n, where Φ is Fitting's semantic function for logic programs (Fitting 1985). (Fitting's semantics, which is the least fixedpoint of Φ, expresses the logical consequences of three-valued Herbrand models of the Clark completion of a logic program.)

Although defeasible logics are usually founded on proofs, there are alternative semantics for these logics: a model-theoretic semantics was defined by Maher (2002), a denotational semantics for $\mathbf{DL}(\partial)$ was presented by Maher (2000), and an argumentation semantics for $\mathbf{DL}(\partial)$ was given by Governatori *et al.* (2004). Each of these approaches provides an alternative characterization of the conclusions derivable by proofs in the logic. However, in this paper, we only use the meta-programming formulation of the proof systems.

In the following, annotated defeasible logic will be defined as an integration of the four defeasible logics discussed above. However, it should be clear that the same

[1] Note that in previous works, these have been denoted by σ and σ^* or $\int$ and $\int^*$. This change of notation is made to accommodate new forms of support introduced in this paper.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

approach can be applied to any set of defeasible logics employing the same logic programming semantics.

3 Annotated defeasible logic

Annotated defeasible logic is the formalism we propose, motivated by the discussion in the introduction. We begin by addressing its syntax, which is an extension of the syntax of defeasible logics.

A tag is any one of the proof tags, or the additional tag `free`. An annotated literal has the form $t\,q$, where t is a tag and q is a literal. An *annotated defeasible rule* has the form

$$r: \quad L_1, \ldots, L_n \Rightarrow q$$

where r is a label, q is a literal, and each L_i is either an annotated literal or a fail-expression, where a *fail-expression* has the form $\texttt{fail}\ L$, where L is an annotated literal. An annotated defeater is defined similarly; strict rules are not annotated.

Roughly, the meaning of a rule

$$r: \quad t_1\, q_1, \ldots, t_n\, q_n, \texttt{fail}\, t_{n+1}\, q_{n+1}, \ldots, \texttt{fail}\, t_m\, q_m \Rightarrow q$$

is that if q_i can be proved using inference rule t_i, for $1 \leqslant i \leqslant n$, and proof of q_i can be demonstrated to fail using inference rule t_i, for $n+1 \leqslant i \leqslant m$, then we have a *prima facie* reason to infer q. As with all defeasible logics, such an inference can be overridden by another rule.

A proof tag only indicates which inference rule should be applied to resolve conflict concerning that literal. Thus, an annotated literal $t\,q$ is asking, roughly, for $+t\,q$ to be proved. A fail-expression $\texttt{fail}\,t\,q$ is asking, roughly, for $-t\,q$ to be proved. The `free` tag has a different meaning than the proof tags. A free literal $\texttt{free}\,q_i$ must be proved by the same inference rule that is intended to prove q. This provides a mechanism by which defeasible rules can be agnostic as to inference rule, which can be determined later, just as defeasible rules in current defeasible logics are.

An *annotated defeasible theory* is a defeasible theory where the defeasible rules are annotated and fail-expressions are allowed. Alternatively, we can think of an annotated defeasible theory as consisting of an unannotated defeasible theory (the *underlying theory*) D that allows fail-expressions, and an *annotation function* α that maps each body literal occurrence to its annotation. In this case, we denote the annotated defeasible theory by $\alpha(D)$. We can consider α a total function, or consider it a partial function mapping literal occurrences to proof tags. The unmapped literals are `free`.

We now turn to expressing the meaning of annotated defeasible theories using the meta-programming approach. The semantics of a theory is parameterized by a logic programming semantics, which is applied to a meta-program.

Given an annotated defeasible theory $D = (F, R, >)$, the theory is represented by facts as follows:

1. $\texttt{fact}(p)$. — if $p \in F$
2. $\texttt{strict}(r_i, p, [L_1, \ldots, L_n])$. — if $r_i : L_1, \ldots, L_n \rightarrow p \in R$

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

3. `defeasible`$(r_i, p, [L_1, \ldots, L_n])$. if $r_i : L_1, \ldots, L_n \Rightarrow p \in R$
4. `defeater`$(r_i, p, [L_1, \ldots, L_n])$. if $r_i : L_1, \ldots, L_n \rightsquigarrow p \in R$
5. `sup`(r_i, r_j). for each pair of rules such that $r_i > r_j$

where the L_i are annotated literals or fail-expressions.

The meta-program to which these facts are input is denoted by $\mathscr{M}$, while the combination of $\mathscr{M}$ and the representation of D is denoted by $\mathscr{M}(D)$. In what follows, we permit ourselves some syntactic flexibility in presenting the meta-program. (For example, we enumerate a list instead of explicitly iterating over it, and express the complementation operation $\sim$ as a function[2]. Furthermore, tags and `fail` are unary functors.) However, there is no technical difficulty in using conventional logic programming syntax to represent this program.

Before we get to the predicates that define the meaning of theories, we define some auxiliary predicates.

As discussed in the introduction to defeasible logics, the different proof tags represent different forms of defeasibility. In particular, some forms block ambiguity, while others propagate ambiguity; some use team defeat, while others require an individual rule to overcome all conflicting rules. The following facts are used to specify, for each proof tag: that it is a proof tag, whether it expresses team defeat or individual defeat, and whether the inference rule blocks or propagates ambiguity. Strictly speaking, we should distinguish the proof tags appearing syntactically in $\mathscr{M}$ from the tags appearing in conclusions (which are not part of the syntax of defeasible logics, but part of its meta-theory). However, because there is a clear correspondence between the two, we find it clearer to use the same symbol for both.

`team`(∂).	`ambiguity_blocking`(∂^*).	`proof_tag`(∂^*).
`team`(δ).	`ambiguity_blocking`(∂).	`proof_tag`(∂).
`indiv`(∂^*).	`ambiguity_propagating`(δ^*).	`proof_tag`(δ^*).
`indiv`(δ^*).	`ambiguity_propagating`(δ).	`proof_tag`(δ).

The following clauses define the class of all rules and the class of supportive rules. Defeaters are not supportive rules because they can only be used to prevent other conclusions; they cannot support any conclusion.

`supportive_rule`$(Label, Head, Body)$:-
  `strict`$(Label, Head, Body)$.

`supportive_rule`$(Label, Head, Body)$:-
  `defeasible`$(Label, Head, Body)$.

`rule`$(Label, Head, Body)$:-
  `supportive_rule`$(Label, Head, Body)$.

`rule`$(Label, Head, Body)$:-
  `defeater`$(Label, Head, Body)$.

[2] The complement of p is $\neg p$ and the complement of $\neg p$ is p. $\sim$ is unrelated to `fail`, since it is the complement of classical negation.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

The next clauses express monotonic provability.

$c1$ `definitely`(X) :-
  `fact`(X).

$c2$ `definitely`(X) :-
  `strict`$(R, X, [Y_1, \ldots, Y_n])$,
  `definitely`(Y_1),. . . ,`definitely`(Y_n).

In the predicate expressing defeasible inference, `defeasibly`, one argument is written as a subscript `Z` in the following clauses. That argument takes as its value one of the four proof tags and represents the inference rule that should be applied to resolve conflict for the literal in the other argument, unless the literal has a proof annotation. All clauses for predicates with a subscript `Z` implicitly contain `proof_tag(Z)` in their body. In clause $c3$, we see that `free`-annotated literals are to be proved according to `Z`.

In clause $c4$, fail-expressions are defined: failure is implemented by negation. This is valid because the logics involved satisfy the Principle of Strong Negation. For such logics, the conditions for $-d$ inference rules are a negation of the conditions for $+d$ inference rules. In both defeasible logics and logic programming, failure-to-prove is a primitive notion, available in defeasible logics through negative tags and in logic programming through negation. Hence, it is not surprising that failure is implemented by negation in the meta-program.

The remaining two clauses are reflective of the basic structure of defeasible reasoning. Clause $c5$ expresses that any literally that is definitely true (proved monotonically from facts and strict rules) is also defeasibly true. Clause $c6$ handles an annotated literal by using the tag Y as the subscript argument in subsidiary computations. This clause says that a literal X, annotated by Y, is proved if the negation of X is not proved monotonically and there is a supportive rule R that is not overruled, each of whose body literals are proved defeasibly according to Y.

$c3$ `defeasibly`$_{\mathtt{Z}}$(`free` X) :-
  `proof_tag`(Z),
  `defeasibly`$_{\mathtt{Z}}(Z\ X)$.

$c4$ `defeasibly`$_{\mathtt{Z}}$(`fail` X) :-
  `not defeasibly`$_{\mathtt{Z}}(X)$.

$c5$ `defeasibly`$_{\mathtt{Z}}(X)$:-
  `definitely`(X).

$c6$ `defeasibly`$_{\mathtt{Z}}(Y\ X)$:-
  `proof_tag`(Y),
  `not definitely`$(\sim X)$,
  `supportive_rule`$(R, X, [W_1, \ldots, W_n])$,
  `defeasibly`$_{\mathtt{Y}}(W_1)$,. . . ,`defeasibly`$_{\mathtt{Y}}(W_n)$,
  `not overruled`$_{\mathtt{Y}}(R, X)$.

The basic structure of overruling a rule is similar for all defeasible logics: the body of the overruling rule must be proved and the rule not "defeated". However, it varies depending on whether the logic blocks or propagates ambiguity. In an ambiguity

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

blocking logic, the body of the overruling rule must be established defeasibly, whereas in an ambiguity propagating logic, the body of the overruling rule need only be supported.

$c7$ $\texttt{overruled}_{\texttt{Z}}(R, X)$:-
$\texttt{ambiguity\_blocking}(Z)$,
$\texttt{rule}(S, \sim X, [U_1, \ldots, U_n])$,
$\texttt{defeasibly}_{\texttt{Z}}(U_1), \ldots, \texttt{defeasibly}_{\texttt{Z}}(U_n)$,
not $\texttt{defeated}_{\texttt{Z}}(R, S, \sim X)$.

$c8$ $\texttt{overruled}_{\texttt{Z}}(R, X)$:-
$\texttt{ambiguity\_propagating}(Z)$,
$\texttt{rule}(S, \sim X, [U_1, \ldots, U_n])$,
$\texttt{supported}_{\texttt{Z}}(U_1), \ldots, \texttt{supported}_{\texttt{Z}}(U_n)$,
not $\texttt{defeated}_{\texttt{Z}}(R, S, \sim X)$.

The notion of defeat varies, depending on whether a logic involves team defeat or individual defeat. In individual defeat, the overruling rule S is defeated if the rule R it tries to overrule is superior to S. In team defeat, S is defeated if there is a rule T (possibly the same as R) that is superior to S and whose body can be proved.

$c9$ $\texttt{defeated}_{\texttt{Z}}(R, S, \sim X)$:-
$\texttt{team}(Z)$,
$\texttt{sup}(T, S)$,
$\texttt{supportive\_rule}(T, X, [V_1, \ldots, V_n])$,
$\texttt{defeasibly}_{\texttt{Z}}(V_1), \ldots, \texttt{defeasibly}_{\texttt{Z}}(V_n)$.

$c10$ $\texttt{defeated}_{\texttt{Z}}(R, S, \sim X)$:-
$\texttt{indiv}(Z)$,
$\texttt{sup}(R, S)$.

The structure of this meta-program makes one point clear that was less readily apparent in papers by Antoniou *et al.* (2000) and Billington *et al.* (2010): treatment of ambiguity concerns how the body of an overruling rule is proved, while the choice of team/individual defeat concerns how an overruling rule can be defeated.

For the ambiguity propagating logics, we must define the notion of "supported". The intuition is that a literal is supported if there is a chain of supportive rules that form a proof tree for the literal, and each supportive rule is not beaten (i.e., overruled) by a rule that is proved defeasbily. In ordinary defeasible logics, support is only needed for the ambiguity propagating logics but, for annotated defeasible theories, we also need to have support for ambiguity blocking logics. This is because we might wish to use, as part of the support, a rule that contains an annotated literal such as ∂q. Hence, the `supported` predicate is defined uniformly, with a parameter `Z` specifying the form of defeasibility underlying the support. Thus, we are introducing new forms of support: σ_{∂} and σ_{∂^*}.

As with `defeasibly`, the clauses for `supported` address-free literals, fail-expressions, literals that are proved definitely, and proof-annotated literals. Note how the parameter `Z` to `supported` is used by `beaten` to select the form of defeasibility for which the body of an overruling rule must be proved.

$c11$ $\texttt{supported}_{\texttt{Z}}(\texttt{free}\ X)$:-
$\texttt{supported}_{\texttt{Z}}(Z\ X)$.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

$c12$ $\texttt{supported}_{\texttt{Z}}(\texttt{fail}\ X)$:-
$\texttt{not supported}_{\texttt{Z}}(X)$.

$c13$ $\texttt{supported}_{\texttt{Z}}(X)$:-
$\texttt{definitely}(X)$.

$c14$ $\texttt{supported}_{\texttt{Z}}(Y\ X)$:-
$\texttt{proof\_tag}(Y)$,
$\texttt{supportive\_rule}(R, X, [W_1, \ldots, W_n])$,
$\texttt{supported}_{\texttt{Y}}(W_1), \ldots, \texttt{supported}_{\texttt{Y}}(W_n)$,
$\texttt{not beaten}_{\texttt{Y}}(R, X)$.

$c15$ $\texttt{beaten}_{\texttt{Z}}(R, X)$:-
$\texttt{rule}(S, \sim X, [W_1, \ldots, W_n])$,
$\texttt{defeasibly}_{\texttt{Z}}(W_1), \ldots, \texttt{defeasibly}_{\texttt{Z}}(W_n)$,
$\texttt{sup}(S, R)$.

Let us now examine how to put annotated defeasible logic to work by revisiting the compensation example presented in the introduction. As we have already discussed, *Guilty* must be proven with the "beyond reasonable doubt" proof standard to derive that the defendant is entitled to receive a compensation.

As we have alluded to in the introduction, Gordon and Walton (2009) proposed to model proof standards such as scintilla of evidence, preponderance of evidence, clear and convincing case, beyond reasonable doubts, and dialectical validity using rule-based argumentation. For example, they define that the proof standard of preponderance of evident for a literal p is satisfied if and only if the maximum weight of applicable arguments for p exceeds some threshold α, and the difference between the maximum weight of the applicable arguments for p and the maximum weight of the applicable arguments against p exceeds some threshold β. Governatori (2011) shows how the weights and thresholds can be modelled by a preference relation (superiority) over arguments (rules), and it establishes the following relationships between the proof standards and proof tags:

Proof standard(s)	Proof tag
scintilla of evidence	σ
preponderance of evidence, clear and convincing case	∂^*
beyond reasonable doubt, dialectic validity	δ^*

where the distinction between preponderance of evidence and clear and convincing case, and beyond reasonable doubt and dialectic validity depends on how the weights associated to the arguments and thresholds are translated in instances of the superiority relation in the resulting theories. Furthermore, Governatori (2011) provides examples where the definitions of proof standards given by Gordon and Walton (2009) exhibit some counter-intuitive conclusions. To obviate such limitations, he proposes an alternative correspondence between proof tags in defeasible logic variants and proof standards, including the following:

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

Proof standard(s)	Proof tag
substantial evidence	σ
preponderance of evidence	∂
beyond reasonable doubt	δ
dialectic validity	δ (when the superiority relation is ignored)

Thus, the proof standard of beyond reasonable doubt corresponds to defeasible provability using ambiguity propagation. Accordingly, we can replace *BeyondReasonableDoubt* in rule r'_5 with $+\delta$. All the other literals appearing in the body of the rules do not require special proof standards, and thus we can annotate them with `free`. Consequently, the formalization of this scenario in annotated defeasible logic is

$$\begin{aligned}
r_1 &: \quad \texttt{free}\ \mathit{EvidenceA} \Rightarrow \neg \mathit{Responsible}\\
r_2 &: \quad \texttt{free}\ \mathit{EvidenceB} \Rightarrow \quad \mathit{Responsible}\\
r_3 &: \texttt{free}\ \mathit{Responsible} \Rightarrow \quad \mathit{Guilty}\\
r_4 &: \qquad\qquad\qquad\quad \Rightarrow \neg \mathit{Guilty}\\
r_5 &: \qquad +\delta \neg\ \mathit{Guilty} \Rightarrow \quad \mathit{Compensation}\\
r_6 &: \qquad\qquad\qquad\quad \Rightarrow \neg \mathit{Compensation}
\end{aligned}$$

It is easy to verify that we now derive $+\partial \neg \mathit{Compensation}$, that the defendant is not entitled to compensation, as the scenario requires.

4 Properties of annotated defeasible theories

We now investigate properties of annotated defeasible logic, exploiting its logic programming underpinnings.

The first theorem relates the meta-program for annotated defeasible logic to the meta-programs for existing defeasible logics **DL**(d). Those logics do not contain fail-expressions. We write $\models_K$ for logical consequence under Kunen's semantics (Kunen 1987). Recall that $\mathscr{M}_d(D)$ is the meta-programming representation for D in **DL**(d), while $\mathscr{M}(\alpha(D))$ is the meta-programming representation for D annotated by α.

Theorem 1
Let $D = (F, R, >)$ be a defeasible theory, and α be an annotation function for that theory. Let $d \in \{\delta^*, \delta, \partial^*, \partial\}$.

Suppose $\alpha(R)$ contains only annotations `free` and d, and there is no fail-expression in R. Then, for every literal q

- $\mathscr{M}(\alpha(D)) \models_K \texttt{defeasibly}_d(d\ q)$ iff $\mathscr{M}_d(D) \models_K \texttt{defeasibly}(q)$
- $\mathscr{M}(\alpha(D)) \models_K \neg\texttt{defeasibly}_d(d\ q)$ iff $\mathscr{M}_d(D) \models_K \neg\texttt{defeasibly}(q)$

Furthermore, if $d \in \{\delta^*, \delta\}$,

- $\mathscr{M}(\alpha(D)) \models_K \texttt{supported}_d(d\ q)$ iff $\mathscr{M}_d(D) \models_K \texttt{supported}(q)$
- $\mathscr{M}(\alpha(D)) \models_K \neg\texttt{supported}_d(d\ q)$ iff $\mathscr{M}_d(D) \models_K \neg\texttt{supported}(q)$

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

The proof is based on separately unfolding $\mathscr{M}(\alpha(D))$ and $\mathscr{M}_d(D)$ until they have essentially the same form.

As an immediate corollary to this theorem, we see that annotated defeasible theories are a conservative extension of defeasible theories. Let the *free annotation function* be the annotation function that maps every body literal occurrence in D to `free`. For any defeasible theory D, the unannotated theory behaves exactly the same as the theory annotated by the free annotation function.

Corollary 2
Suppose that α_F is the free annotation function for D. Let $d \in \{\delta^*, \delta, \partial^*, \partial\}$. Then, for every literal q:

- $\mathscr{M}(\alpha_F(D)) \models_K \texttt{defeasibly}_\texttt{d}(q)$ iff $D \vdash +dq$
- $\mathscr{M}(\alpha_F(D)) \models_K \neg\texttt{defeasibly}_\texttt{d}(q)$ iff $D \vdash -dq$

Furthermore, if $d \in \{\delta^*, \delta\}$,

- $\mathscr{M}(\alpha_F(D)) \models_K \texttt{supported}_\texttt{d}(q)$ iff $D \vdash +\sigma_d q$
- $\mathscr{M}(\alpha_F(D)) \models_K \neg\texttt{supported}_\texttt{d}(q)$ iff $D \vdash -\sigma_d q$

For any tag d and an annotated defeasible theory D, we define $+d(D) = \{q \mid D \vdash +dq\} = \{q \mid \mathscr{M}(D) \models_K \texttt{defeasibly}_\texttt{d}(q)\}$ and $-d(D) = \{q \mid D \vdash -dq\} = \{q \mid \mathscr{M}(D) \models_K \neg\texttt{defeasibly}_\texttt{d}(q)\}$. Similarly, we define $+\sigma_d(D)$ as $\{q \mid \mathscr{M}(D) \models_K \texttt{supported}_\texttt{d}(q)\}$ and $-\sigma_d(D)$ as $\{q \mid \mathscr{M}(D) \models_K \neg\texttt{supported}_\texttt{d}(q)\}$.

We can now extend the inclusion theorem of Billington *et al.* (2010) to the new tags and annotated defeasible logic. This theorem shows the relative inference strength of the different forms of defeasibility.

Theorem 3 (*Inclusion theorem*)
Let D be an annotated defeasible theory.

(a) $+\Delta(D) \subseteq +\delta^*(D) \subseteq +\delta(D) \subseteq +\partial(D) \subseteq +\sigma_\delta(D) \subseteq +\sigma_{\delta^*}(D)$
(b) $-\sigma_{\delta^*}(D) \subseteq -\sigma_\delta(D) \subseteq -\partial(D) \subseteq -\delta(D) \subseteq -\delta^*(D) \subseteq -\Delta(D)$
(c) $+\partial(D) \subseteq +\sigma_\partial(D) \subseteq +\sigma_\delta(D)$
(d) $-\sigma_\delta(D) \subseteq -\sigma_\partial(D) \subseteq -\partial(D)$
(e) $+\delta^*(D) \subseteq +\partial^*(D) \subseteq +\sigma_{\partial^*}(D) \subseteq +\sigma_{\delta^*}(D)$
(f) $-\sigma_{\delta^*}(D) \subseteq -\sigma_{\partial^*}(D) \subseteq -\partial^*(D) \subseteq -\delta^*(D)$

The proof is by induction on the iteration stages of Fitting's $\Phi_{\mathscr{M}(D)}$ function.

The inclusions in this theorem are presented graphically in Figure 1. The relation $t_1 \subset t_2$ expresses that, for all defeasible theories D, $+t_1(D) \subseteq +t_2(D)$ and $-t_1(D) \supseteq -t_2(D)$, and, for some defeasible theory D, $+t_1(D) \subset +t_2(D)$. The containments come from the theorem, while their strictness is demonstrated by simple examples. Examples also show that there are no containments that can be added to the figure.

This ordering on tags can be extended to annotation functions. Let α_1 and α_2 be annotation functions for a defeasible theory D. We define $\alpha_1 \sqsubseteq \alpha_2$ iff for every body occurrence o of every literal in D, $\alpha_1(o) \subset \alpha_2(o)$. If such an ordering had implications for the conclusions of the annotated theories, it would provide a useful basis from which to reason about annotated defeasible theories. Unfortunately, the

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

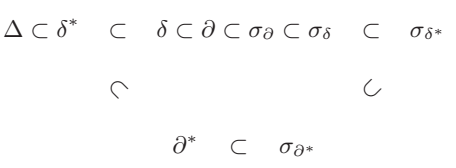

Fig. 1. Ordering of inference rules by relative inference strength.

most obvious possibility – a kind of monotonicity – does not hold, as the following example shows.

Example 4
Let D consist of the rules

$$\begin{array}{llll} r_1: & \Rightarrow p & r_5: q \Rightarrow & s \\ r_2: & \Rightarrow \neg p & r_6: & \Rightarrow \neg s \\ r_3: & \Rightarrow q & & \\ r_4: \neg p & \Rightarrow \neg q & & \end{array}$$

with $r_5 > r_6$.

Let α_1 map q in r_5 to δ, and α_2 map q in r_5 to ∂ (with all other occurrences mapped to `free`). Then $\alpha_1 \sqsubseteq \alpha_2$. Rules r_1–r_4 are a standard example distinguishing ambiguity blocking and propagating behaviours. $+\partial q$ and $-\delta q$ can be concluded. Consequently, in $\alpha_1(D)$, we conclude $+\partial \neg s$ and $-\partial s$, while in $\alpha_2(D)$, we conclude $-\partial \neg s$ and $+\partial s$.

Thus, we see that a strengthening of the annotation function (in the $\sqsubseteq$ ordering) does not necessarily lead to a strengthening of the conclusions of the annotated defeasible theory.

For the defeasible logics we address, the consequences of a defeasible theory can be computed in linear time, with respect to the size of the theory (Maher 2001; Billington *et al.* 2010), but these logics only support one form of defeasibility. Annotated defeasible logic allows the interaction between the different inference rules but, nevertheless, we expect its consequences can also be computed in linear time, although with a larger constant factor. (Certainly, it is straightforward to show we can compute consequences in quadratic time. See the supplementary material.) Let

$$\begin{aligned} \mathscr{C}(D) = \{&+dq \mid \mathscr{M}(D) \models_K \texttt{defeasibly}_d(q), d \in T\} \cup \\ \{&-dq \mid \mathscr{M}(D) \models_K \neg\texttt{defeasibly}_d(q), d \in T\} \cup \\ \{&+\sigma_d q \mid \mathscr{M}(D) \models_K \texttt{supported}_d(q), d \in T\} \cup \\ \{&-\sigma_d q \mid \mathscr{M}(D) \models_K \neg\texttt{supported}_d(q), d \in T\} \end{aligned}$$

where D is an annotated defeasible theory, $T = \{\partial, \partial^*, \delta, \delta^*\}$ refers to the four main forms of defeasibility, and q ranges over annotated literals.

Conjecture 5
Let D be an annotated defeasible theory, and $|D|$ be the number of symbols in D. Then the set of consequences $\mathscr{C}(D)$ can be computed in time $O(|D|)$.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

5 Different forms of failure

One advantage of the framework of Maher and Governatori (1999) and Antoniou *et al.* (2000) is that different notions of failure can be obtained by different semantics for logic programs. In this section, we demonstrate that annotated defeasible logic is a conservative extension of those logics for many such semantics.

Many of the logic programming semantics we will focus on can be seen to be derived from the three-valued stable models (Przymusinski 1990) (also known as *partial stable models*, but distinct from partial stable models by Saccà and Zaniolo (1990)). In addition to the semantics based on all partial stable models, there is the well-founded model (Gelder *et al.* 1991), which is the least partial stable model under the information ordering (Przymusinski 1990) (called F-least by Przymusinski (1990)); the (two-valued) stable models (Gelfond and Lifschitz 1988); the regular models (You and Yuan 1994), which are the maximal partial stable models under set inclusion on the positive literals; and the L-stable models (Eiter *et al.* 1997), which are the maximal partial stable models under set inclusion on positive and negative literals or, equivalently, the minimal partial stable models under set inclusion on the undefined literals. The interest in these semantics derives from the use of their counterparts in abstract argumentation (Caminada *et al.* 2015).

Let $\mathcal{S}$ denotes the collection of semantics mentioned above, with the exception of the stable semantics. That is, $\mathcal{S} = \{partial\ stable, well\text{-}founded, regular, L\text{-}stable, Kunen, Fitting\}$. These semantics (and the stable semantics) are preserved by unfolding (see Aravindan and Dung (1995) and Maher (2017)). Consequently, Theorem 1 extends to the semantics in $\mathcal{S}$:

Theorem 6
Let $D = (F, R, >)$ be a defeasible theory, and α be an annotation for that theory. Let $d \in \{\delta^*, \delta, \partial^*, \partial\}$. Suppose $\alpha(R)$ contains only annotations `free` and d, and there is no fail-expression in R. Let $S \in \mathcal{S}$. Then

- $\mathcal{M}(\alpha(D)) \models_S \texttt{defeasibly}_\texttt{d}(q)$ iff $\mathcal{M}_d(D) \models_S \texttt{defeasibly}(q)$
- $\mathcal{M}(\alpha(D)) \models_S \neg\texttt{defeasibly}_\texttt{d}(q)$ iff $\mathcal{M}_d(D) \models_S \neg\texttt{defeasibly}(q)$

and, if $d \in \{\delta^*, \delta\}$,

- $\mathcal{M}(\alpha(D)) \models_S \texttt{supported}_\texttt{d}(q)$ iff $\mathcal{M}_d(D) \models_S \texttt{supported}(q)$
- $\mathcal{M}(\alpha(D)) \models_S \neg\texttt{supported}_\texttt{d}(q)$ iff $\mathcal{M}_d(D) \models_S \neg\texttt{supported}(q)$

More generally, the S-models of $\mathcal{M}(\alpha(D))$ restricted to $\texttt{defeasibly}_\texttt{d}$ are identical (up to predicate renaming) to the S-models of $\mathcal{M}_d(D)$ restricted to `defeasibly`.

In particular, annotated defeasible logic under the well-founded semantics extends the well-founded defeasible logics (Maher and Governatori 1999; Maher *et al.* 2011).

This theorem does *not* apply to the stable model semantics, because of the possibility that $\mathcal{M}_d(D)$ has stable models but $\mathcal{M}(D)$ does not. This, in turn, occurs because $\mathcal{M}(D)$ represents all the inference rules, while $\mathcal{M}_d(D)$ does not. Technically, the proof fails because the deletion of irrelevant clauses is not sound under the stable model semantics. To see what can go wrong, consider the following example.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

Example 7
Let D consist of the rules

$$\begin{array}{llll} r_1: & \Rightarrow & p & \qquad r_3: \Rightarrow q \\ r_2: p, q \Rightarrow & \neg p & & \qquad r_4: \Rightarrow q \\ & & & \qquad r_5: \Rightarrow \neg q \\ & & & \qquad r_6: \Rightarrow \neg q \end{array}$$

with $r_3 > r_5$ and $r_4 > r_6$.

After unfoldings and simplifications, $\mathscr{M}(D)$ contains

$c16$ $\texttt{defeasibly}_\partial(\partial\ p)$:-
    $\texttt{not overruled}_\partial(r_1, p)$.

$c17$ $\texttt{overruled}_\partial(r_1, p)$:-
    $\texttt{defeasibly}_\partial(\partial\ p)$,
    $\texttt{defeasibly}_\partial(\partial\ q)$.

and similar clauses for ∂^* (as well as other clauses).

It is clear that if $\texttt{defeasibly}_\partial(\partial\ q)$ holds, then the structure of these two clauses prevents the existence of a stable model, while if $\neg\texttt{defeasibly}_\partial(\partial\ q)$, then $\texttt{defeasibly}_\partial(\partial\ p)$ holds in every stable model, assuming there is nothing else preventing the formation of stable models. The same applies for ∂^*.

Now, $\texttt{defeasibly}_\partial(\partial\ q)$ holds, but $\texttt{defeasibly}_{\partial^*}(\partial^*\ q)$ does not. It follows, from the proof of Theorem 1, that $\mathscr{M}_{\partial^*}(D)$ has stable models but $\mathscr{M}(D)$ does not.

Thus, Theorem 6 holds for stable models only when *all* forms of defeasibility and supportedness have stable models.

6 Related work

Among the features of annotated defeasible theories are (1) the language supports multiple forms of defeasibility within a single defeasible theory, indeed within a single rule; (2) the language provides explicit fail-expressions; (3) the framework has the ability to incorporate different notions of failure-to-prove, corresponding to different semantics of negation-as-failure. No other formalism for defeasible reasoning has all these features.

Courteous logic programs (Grosof 1997) (and later developments (Wan *et al.* 2009; Wan *et al.* 2015)) permit negation-as-failure expressions in defeasible rules, which are essentially the same as fail-expressions. Antoniou *et al.* (2000) discussed a specific transformation for eliminating these expressions from courteous logic programs; that transformation is not sound for ambiguity propagating logics. Our meta-programming approach to fail-expressions was discussed by Maher and Governatori (1999), for a language with a single form of defeasibility, and our Theorem 1 extends to languages with such fail-expressions.

Within proof-theoretic treatments of defeasible logics (see, for example, Maier and Nute (2010) and Billington *et al.* (2010)), the logics can incorporate multiple forms of defeasibility, but they do not interact. For example, the proof of $+\partial q$ cannot depend on the proof of $+\delta p$: it can only depend on proofs of ∂ conclusions.

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

Within the meta-programming framework of Maher and Governatori (1999) and Antoniou *et al.* (1999), a logic has only a single form of defeasibility, although this can be easily remedied by the use of multiple variants of the `defeasibly` predicate. Still, the multiple forms do not interact. Structured argumentation approaches, such as ASPIC+ (Prakken 2010), use unannotated rules without an inference rule (in the sense above) and hence define a single form of defeasibility. A meta-program component of the languages LPDA and ASPDA by Wan *et al.* (2009; 2015), called an argumentation theory, is capable of specifying a different inference rule for each literal, but not for each *occurrence* of each literal. Thus, although they provide more interaction than the defeasible logics, they do not provide the ability to apply different inference rules to the same atom.

It should be noted that the logics of Billington *et al.* (2010) are able to simulate each other as proved by Maher (2012; 2013) (and ASPIC+ appears expressive enough to simulate these logics), but such an approach to incorporating multiple forms of defeasibility leads to an unnatural representation and has computational penalties. It also fails to represent free-expressions, since the top level form of defeasibility must be fixed before simulations can be coded.

Annotated logic programs (Kifer and Subrahmanian 1992) are an extension of logic programs to multi-valued logics, where the truth values are assumed to form an upper semi-lattice. Atoms in the body are annotated by truth values and the head is annotated by a function of those truth values. Thus, there are some similarities to annotated defeasible logic, in the use of annotations, including a similarity of variable annotations and free-expressions. However, annotated defeasible logic uses proof tags – not truth values – as annotations, and does not assume any ordering on the annotations. Further, the semantics of annotated logic programs is essentially a disjunction of the conclusions of rules, so this formalism is unable to represent the overriding of a rule by a competing rule.

Most defeasible logics support a single semantics of failure: Kunen's (Billington *et al.* 2010), well-founded (Grosof 1997; Maher and Governatori 1999; Wan *et al.* 2009; Maier and Nute 2010; Maher *et al.* 2011), or stable (Verheij 2003; Maier 2013; Wan *et al.* 2015). Apart from the framework of Antoniou *et al.* (2000), the only defeasible formalisms supporting multiple semantics are structured argumentation languages like ASPIC+ (Prakken 2010). But such languages do not support multiple forms of defeasibility.

The annotation mechanism we presented is closely related to the introduction of modal literals in modal defeasible logic (Governatori *et al.* 2016), where each rule is labelled with the mode ($\Box$) its conclusion can be proved and the literals $\Box q$ and $\Diamond q$ correspond to $+\partial_{\Box} q$ and $-\partial_{\Box} \neg q$. While each modality has its own inference rule, each supports a single form of defeasibility. This raised the question whether different forms of defeasibility could be combined: the present paper offers a positive answer.

Conclusion

We have argued that we need a formalism that supports different kinds of defeasible reasoning, and introduced annotated defeasible logic to fulfil that requirement. The

https://doi.org/10.1017/S1471068417000266 Published online by Cambridge University Press

semantics of the annotated logic is defined through a logic program, and we are able to exploit that medium to prove properties of the logic.

Supplementary materials

For supplementary material for this article, please visit https://doi.org/10.1017/S1471068417000266

References

Antoniou, G., Billington, D., Governatori, G. and Maher, M. J. 2000. A flexible framework for defeasible logics. In *Proc. of AAAI National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, 405–410.

Antoniou, G., Maher, M. J. and Billington, D. 2000. Defeasible logic versus logic programming without negation as failure. *Journal of Logical and Algebraic Methods in Programming 42,* 1, 47–57.

Aravindan, C. and Dung, P. M. 1995. On the correctness of unfold/fold transformation of normal and extended logic programs. *Journal of Logical and Algebraic Methods in Programming 24,* 3, 201–217.

Billington, D., Antoniou, G., Governatori, G. and Maher, M. J. 2010. An inclusion theorem for defeasible logics. *ACM Transactions on Computational Logic 12,* 1, 6.

Caminada, M., Sá, S., Alcântara, J. and Dvorák, W. 2015. On the equivalence between logic programming semantics and argumentation semantics. *International Journal of Approximate Reasoning 58*, 87–111.

Eiter, T., Leone, N. and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence 19,* 1–2, 59–96.

Fitting, M. 1985. A Kripke–Kleene semantics for logic programs. *Journal of Logical and Algebraic Methods in Programming 2,* 4, 295–312.

Gelder, A. V., Ross, K. A. and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *Journal of The Academy of Clinical Microbiologists 38,* 3, 620–650.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of JICSLP*, 1070–1080.

Gordon, T. F. and Walton, D. 2009. Proof burdens and standards. In *Argumentation in Artificial Intelligence*, I. Rahwan and G. Simari, Eds. Springer, Berlin, 239–260.

Governatori, G. 2011. On the relationship between Carneades and defeasible logic. In *Proc. of 13th International Conference on Artificial Intelligence and Law*, K. D. Ashley and T. M. van Engers, Eds. ACM, 31–40.

Governatori, G., Maher, M. J., Antoniou, G. and Billington, D. 2004. Argumentation semantics for defeasible logic. *Journal of Logic and Computation 14,* 5, 675–702.

Governatori, G., Olivieri, F., Scannapieco, S., Rotolo, A. and Cristani, M. 2016. The rational behind the concept of goal. *Theory and Practice of Logic Programming, 16,* 3, 296–324.

Governatori, G. and Sartor, G. 2010. Burdens of proof in monological argumentation. In *Proc. of 23rd Annual Conference on Legal Knowledge and Information Systems*, R. Winkels, Ed. Frontiers in Artificial Intelligence and Applications, vol. 223. IOS Press, Amsterdam, 57–66.

Grosof, B. N. 1997. Prioritized conflict handling for logic programs. In *Proc. of ILPS*, 197–211.

Kifer, M. and Subrahmanian, V. S. 1992. Theory of generalized annotated logic programming and its applications. *Journal of Logic and Computation, 12,* 3&4, 335–367.

Kunen, K. 1987. Negation in logic programming. *Journal of Logic and Computation, 4,* 4, 289–308.

Maher, M. J. 2000. A denotational semantics of defeasible logic. In *Proc. of 1st International Conference on Computational Logic – CL 2000*, 24–28 July, 2000, London, UK, 209–222.

Maher, M. J. 2001. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming 1,* 6, 691–711.

Maher, M. J. 2002. A model-theoretic semantics for defeasible logic. In *Proc. of ICLP 2002 Workshop Paraconsistent Computational Logic*, 67–80.

Maher, M. J. 2012. Relative expressiveness of defeasible logics. *Theory and Practice of Logic Programming, 12,* 4–5, 793–810.

Maher, M. J. 2013. Relative expressiveness of defeasible logics II. *Theory and Practice of Logic Programming, 13,* 4–5, 579–592.

Maher, M. J. 2017. Relating concrete defeasible reasoning formalisms and abstract argumentation. *Fundamenta Informaticae, 153,* 1, 1–28.

Maher, M. J. and Governatori, G. 1999. A semantic decomposition of defeasible logics. In *Proc. of AAAI National Conference on Artificial Intelligence*, AAAI Press, 299–305.

Maher, M. J., Governatori, G. and Lam, H. P. 2011. Well-founded defeasible logics, Reasoning Research Institute. Technical report.

Maier, F. 2013. Interdefinability of defeasible logic and logic programming under the well-founded semantics. *Theory and Practice of Logic Programming, 13,* 1, 107–142.

Maier, F. and Nute, D. 2010. Well-founded semantics for defeasible logic. *Synthese, 176,* 2, 243–274.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument and Computation*, *1*, 93–124.

Prakken, H. and Sartor, G. 2007. Formalising arguments about the burden of persuasion. In *Proc. of 11th International Conference on Artificial Intelligence and Law*, A. Gardner and R. Winkels, Eds. ACM, 97–106.

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

Saccà, D. and Zaniolo, C. 1990. Stable models and non-determinism in logic programs with negation. In *Proc. of PODS*, 205–217.

Verheij, B. 2003. DefLog: On the logical interpretation of prima facie justified assumptions. *Journal of Computer and System Sciences, 13,* 3, 319–346.

Wan, H., Grosof, B. N., Kifer, M., Fodor, P. and Liang, S. 2009. Logic programming with defaults and argumentation theories. In *Proc. of ICLP*, 432–448.

Wan, H., Kifer, M. and Grosof, B. N. 2015. Defeasibility in answer set programs with defaults and argumentation rules. *Semantic Web, 6,* 1, 81–98.

You, J. and Yuan, L. 1994. A three-valued semantics for deductive databases and logic programs. *Journal of Computer and System Sciences, 49,* 2, 334–361.