Annotated Defeasible Logic

Guido Governatori  
_Data61, CSIRO, Australia_  
E-mail: guido.governatori@data61.csiro.au

Michael J. Maher  
_Reasoning Research Institute, Australia_  
E-mail: michael.maher@reasoning.org.au

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

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.

This paper is under consideration for acceptance in _Theory and Practice of Logic Programming._

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

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:

\[
\text{IllegalBehaviour}, \neg \text{Justification} \Rightarrow \text{Liability}
\]

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 rules:

\[
\begin{align*}
\text{Injury}, \neg \text{Justification} & \Rightarrow \text{Liability} \\
\text{Threat} & \Rightarrow \text{Justification} \\
\text{Stronger} & \Rightarrow \neg \text{Justification}
\end{align*}
\]

Here, in case we are not able to assess whether the threat was real, we have a credible argument for \(\neg \text{Justification}\) (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:

\[
\begin{align*}
\text{Tort}, \text{BurdenProduction}(\neg \text{Justification}) & \Rightarrow \text{CivilCaseLiability} \\
\text{Crime}, \text{BurdenPersuasion}(\neg \text{Justification}) & \Rightarrow \text{CriminalCaseLiability}
\end{align*}
\]

where \text{BurdenProduction} and \text{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 \textit{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:

\[
\begin{align*}
r_1 : \text{EvidenceA} & \Rightarrow \neg \text{Responsible} \\
r_2 : \text{EvidenceB} & \Rightarrow \text{Responsible} \\
r_3 : \text{Responsible} & \Rightarrow \text{Guilty} \\
r_4 : & \Rightarrow \neg \text{Guilty}
\end{align*}
\]

where \(r_3\) is stronger than \(r_4\). Given both \text{EvidenceA} and \text{EvidenceB}, the literal \text{Responsible} is ambiguous. There are applicable rules \((r_1\text{ and }r_2)\) 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 innocence (rule \(r_4\)). In an ambiguity blocking setting we obtain a \(\neg \text{Guilty}\) verdict; the ambiguity about responsibility is blocked from applying to \text{Guilty}.

In contrast, in an ambiguity propagating setting, the ambiguity of \text{Responsible} propagates to \text{Guilty}, and thus the literals \text{Guilty} and \(\neg \text{Guilty}\) 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 \( \neg \text{Guilty} \). In addition, by default, people are not entitled to compensation. The additional elements of this scenario are modelled by the rules:

\[
\begin{align*}
\text{r}_5 & : \quad \neg \text{Guilty} \Rightarrow \text{Compensation} \\
\text{r}_6 & : \quad \Rightarrow \neg \text{Compensation}
\end{align*}
\]

where \( \text{r}_5 \) is stronger than \( \text{r}_6 \).

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, \text{Guilty} and \( \neg \text{Guilty} \) are ambiguous, and this makes \text{Compensation} and \( \neg \text{Compensation} \) 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 achieve by replacing rule \( \text{r}_5 \) with

\[
\text{r}'_5 : \quad \text{BeyondReasonableDoubt}(\neg \text{Guilty}) \Rightarrow \text{Compensation}
\]

where, similarly to what we have done in the previous example, \text{BeyondReasonableDoubt} is an annotation to the literal \( \neg \text{Guilty} \) that holds in case the literal is provable under ambiguity propagation, and the proof standard for \text{Compensation} can be chosen to be ambiguity blocking.

The purpose of this paper is to provide a formalism – \emph{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. 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.

**Defeasible Logics**

In this section we can only present an outline of defeasible logics. Further details can be obtained from (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 \( \Sigma \) of literals (which we assume is closed under negation) and a language \( \Lambda \) of labels. A defeasible theory \( D = (\mathcal{F}, R, >) \) consists of a set of
facts $F$, a finite set of rules $R$, each rule with a distinct label from $\Lambda$, and an acyclic relation $>_{s}$ on $\Lambda$ called the \textit{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 $\rightarrow_{s}$; a defeasible rule, with arrow $\Rightarrow_{s}$; or a defeater, with arrow $\sim_{s}$. 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 in (Billington et al. 2010) correspond to four different pairs of inference rules, tagged $\partial$, $\delta$, $\partial^{*}$, and $\delta^{*}$; 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 $\sim d$ inference rule should be the strong negation of the condition for applying $+d$. The inference rules $\delta$ and $\delta^{*}$ require auxiliary tags and inference rules, denoted by $\sigma_{3}$ and $\sigma_{3}^{*}$, respectively, 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 $\text{DL}(d)$. We write $D \vdash dq$ (respectively, $D \vdash -dq$) if $+dq$ (respectively, $-dq$) can be proved by $\text{DL}(d)$.

The four principal tags and corresponding inference rules represent different intuitions about defeasible reasoning, that is, define different forms of defeasibility: in $\partial$ and $\partial^{*}$ ambiguity is blocked, while in $\delta$ and $\delta^{*}$ ambiguity is propagated; in $\delta$ and $\delta^{*}$ rules for a literal act as a team to overcome competing rules, while in $\partial^{*}$ and $\delta^{*}$ 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 $\sim q$, with $s_{1} > s_{3}$ and $s_{2} > s_{4}$; then no individual rule for $q$ can overcome the rules for $\sim q$, but $s_{1}$ and $s_{2}$ as a team – can, because every rule for $\sim q$ is overidden by some rule in the team. A more detailed discussion of ambiguity and team defeat in the DL framework is given in (Billington et al. 2010; Maher 2012).

In (Maher and Governatori 1999; Antoniou et al. 2000), the inference rules in $\text{DL}(d)$ were reformulated as a meta-program $\mathcal{M}_{d}$: a logic program that takes a representation of a defeasible

\footnote{Note that in previous works these have been denoted by $\sigma$ and $\sigma^{*}$ or $\int$ and $\int^{*}$. This change of notation is made to accommodate new forms of support introduced in this paper.}
Annotated Defeasible Logic

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 $M_d(D)$.) We will take this meta-programming formulation as our starting point, rather than the inference rules as presented in (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 3-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 $\Phi$ is Fitting’s semantic function for logic programs (Fitting 1985). (Fitting’s semantics, which is the least fixedpoint of $\Phi$, expresses the logical consequences of 3-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 in (Maher 2002), a denotational semantics for $DL(\partial)$ was presented in (Maher 2000), and an argumentation semantics for $DL(\partial)$ was given in (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 approach can be applied to any set of defeasible logics employing the same logic programming semantics.

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

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

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 $\text{fail} \, L$, where $L$ is an annotated literal. An annotated defeater is defined similarly; strict rules are not annotated.

Roughly, the meaning of a rule

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

is that if $q_i$ can be proved using inference rule $t_i$, for $1 \leq i \leq n$, and proof of $q_i$ can be demonstrated to fail using inference rule $t_i$, for $n + 1 \leq i \leq m$, 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 $\text{fail} \, t \, q$ is asking, roughly, for $-t \, q$ to be proved. The free tag has a different meaning than the proof tags. A free literal $\text{free} \, q_i$ 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 $\alpha$ that maps each body literal occurrence to its annotation. In this case we denote the annotated defeasible theory by $\alpha(D)$. We can consider $\alpha$ 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 $D = (F, R, \succ)$, the theory is represented by facts as follows:

1. $\text{fact}(p)$.
2. $\text{strict}(r_i, p, [L_1, \ldots, L_n])$.
3. $\text{defeasible}(r_i, p, [L_1, \ldots, L_n])$.
4. $\text{defeater}(r_i, p, [L_1, \ldots, L_n])$.
5. $\text{sup}(r_i, r_j)$.

where the $L_i$ are annotated literals or fail-expressions.

The meta-program to which these facts are input is denoted by $\mathcal{M}$, while the combination of $\mathcal{M}$ and the representation of $D$ is denoted by $\mathcal{M}(D)$. 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.) 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 $\mathcal{M}$ 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.

$\text{team}(\partial)$.
$\text{ambiguity\_blocking}(\partial^*)$.
$\text{proof\_tag}(\partial^*)$.

$\text{team}(\delta)$.
$\text{ambiguity\_blocking}(\delta)$.
$\text{proof\_tag}(\delta)$.

$\text{indiv}(\partial^*)$.
$\text{ambiguity\_propagating}(\delta^*)$.
$\text{proof\_tag}(\delta^*)$.

$\text{indiv}(\delta^*)$.
$\text{ambiguity\_propagating}(\delta)$.
$\text{proof\_tag}(\delta)$.

$\text{sup}(r_i, r_j)$.

---

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.
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).
```

The next clauses express monotonic provability.

```
c1  definitely(X) :-
    fact(X).
c2  definitely(X) :-
    strict(R, X, [Y1, ..., Yn]),
    definitely(Y1),...,definitely(Yn).
```

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  defeasiblyZ2(free X) :-
    proof_tag(Z),
    defeasiblyZ2(Z X).
c4  defeasiblyZ2(fail X) :-
    not defeasiblyZ2(X).
c5  defeasiblyZ2(X) :-
    definitely(X).
```
defeasibly_{2}(Y, X) :-
proof_tag(Y),
not_definitely(\neg X),
supportive_rule(R, X, [W_1, \ldots, W_n]),
defeasibly_{1}(W_1), \ldots, defeasibly_{1}(W_n),
not_overruled_{2}(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 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.

defeated_{2}(R, S, \neg X) :-
team(Z),
sup(T, S),
supportive_rule(T, X, [V_1, \ldots, V_n]),
defeasibly_{2}(V_1), \ldots, defeasibly_{2}(V_n),
not_defeated_{2}(R, S, \neg X).

defeated_{2}(R, S, \neg X) :-
indiv(Z),
sup(R, S).

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.

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 defeasibly. 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 \text{\partial 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: \sigma_{\partial} and \sigma_{\partial^*}.
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.

\[
c_{11} \quad \text{supported}_2(\text{free } X) :- \\
\quad \text{supported}_2(Z X).
\]

\[
c_{12} \quad \text{supported}_2(\text{fail } X) :- \\
\quad \text{not supported}_2(X).
\]

\[
c_{13} \quad \text{supported}_2(X) :- \\
\quad \text{definitely}(X).
\]

\[
c_{14} \quad \text{supported}_2(Y X) :- \\
\quad \text{proof} _\text{tag}(Y), \\
\quad \text{supportive_rule}(R, X, [W_1, \ldots, W_n]), \\
\quad \text{supported}_2(W_1), \ldots, \text{supported}_2(W_n), \\
\quad \text{not beaten}_2(R, X).
\]

\[
c_{15} \quad \text{beaten}_2(R, X) :- \\
\quad \text{rule}(S, \sim X, [W_1, \ldots, W_n]), \\
\quad \text{defeasibly}_2(W_1), \ldots, \text{defeasibly}_2(W_n), \\
\quad \text{sup}(S, R).
\]

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, \textit{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 evidence for a literal $p$ is satisfied if and only if the maximum weight of applicable arguments for $p$ exceeds some threshold $\alpha$, 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 $\beta$. (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     | $\sigma$  |
| preponderance of evidence, clear and convincing case | $\partial^*$ |
| beyond reasonable doubt, dialectical validity | $\delta^*$ |

where the distinction between preponderance of evidence and clear and convincing case, and beyond reasonable doubt and dialectical 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 in (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:
Thus, the proof standard of beyond reasonable doubt corresponds to defeasible provability using ambiguity propagation. Accordingly, we can replace \textit{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 \texttt{free}. Consequently, the formalization of this scenario in annotated defeasible logic is:

\[
\begin{align*}
& r_1: \quad \texttt{free} \ EvidenceA \Rightarrow \neg \textit{Responsible} \\
& r_2: \quad \texttt{free} \ EvidenceB \Rightarrow \textit{Responsible} \\
& r_3: \quad \texttt{free} \ \textit{Responsible} \Rightarrow \textit{Guilty} \\
& r_4: \quad \Rightarrow \neg \textit{Guilty} \\
& r_5: \quad +\delta \neg \textit{Guilty} \Rightarrow \textit{Compensation} \\
& r_6: \quad \Rightarrow \neg \textit{Compensation}
\end{align*}
\]

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

\section*{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 \( \text{DL}(d) \). Those logics do not contain fail-expressions. We write \( \models_K \) for logical consequence under Kunen’s semantics (Kunen 1987). Recall that \( \mathcal{M}_d(D) \) is the meta-programming representation for \( D \) in \( \text{DL}(d) \), while \( \mathcal{M}(\alpha(D)) \) is the meta-programming representation for \( D \) annotated by \( \alpha \).

\textbf{Theorem 1}

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

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

- \( \mathcal{M}(\alpha(D)) \models_K \text{ defeasibly}_d(d \ q) \iff \mathcal{M}_d(D) \models_K \text{ defeasibly}(q) \)
- \( \mathcal{M}(\alpha(D)) \models_K \neg\text{ defeasibly}_d(d \ q) \iff \mathcal{M}_d(D) \models_K \neg\text{ defeasibly}(q) \)

Furthermore, if \( d \in \{\delta^*, \delta\} \):

- \( \mathcal{M}(\alpha(D)) \models_K \text{ supported}_d(d \ q) \iff \mathcal{M}_d(D) \models_K \text{ supported}(q) \)
- \( \mathcal{M}(\alpha(D)) \models_K \neg\text{ supported}_d(d \ q) \iff \mathcal{M}_d(D) \models_K \neg\text{ supported}(q) \)

The proof is based on separately unfolding \( \mathcal{M}(\alpha(D)) \) and \( \mathcal{M}_d(D) \) 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 \textit{free annotation function} be the annotation
functions for a defeasible theory \( D \) containments that can be added to the figure. While their strictness is demonstrated by simple examples. Examples also show that there are no containments for some defeasible theory \( D \) expresses that, for all defeasible theories \( D \)

\[
\Delta \subset \delta^* \subset \delta \subset \sigma_\delta \subset \sigma_\delta^* \subset \sigma_\delta^*
\]

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

function that maps every body literal occurrence in \( D \) to \text{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 \( \alpha_F \) is the free annotation function for \( D \). Let \( d \in \{ \delta^*, \delta, \partial^*, \partial \} \). Then, for every literal \( q \),

- \( \mathcal{M}(\alpha_F(D)) \models K \text{ defeasibly}_d(q) \) iff \( D \vdash dq \)
- \( \mathcal{M}(\alpha_F(D)) \models K \neg \text{ defeasibly}_d(q) \) iff \( D \vdash dq \)

Furthermore, if \( d \in \{ \delta^*, \delta \} \),

- \( \mathcal{M}(\alpha_F(D)) \models K \text{ supported}_d(q) \) iff \( D \vdash +dq \)
- \( \mathcal{M}(\alpha_F(D)) \models K \neg \text{ supported}_d(q) \) iff \( D \vdash -dq \)

For any tag \( d \) and an annotated defeasible theory \( D \) we define \( +d(D) = \{ q \mid D \vdash dq \} = \{ q \mid \mathcal{M}(D) \models K \text{ defeasibly}_d(q) \} \) and \( -d(D) = \{ q \mid D \vdash dq \} = \{ q \mid \mathcal{M}(D) \models K \neg \text{ defeasibly}_d(q) \} \). Similarly, we define \( +\sigma_d(D) \) as \( \{ q \mid \mathcal{M}(D) \models K \text{ supported}_d(q) \} \) and \( -\sigma_d(D) \) as \( \{ q \mid \mathcal{M}(D) \models K \neg \text{ supported}_d(q) \} \).

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) \( +\Delta(D) \subseteq +\delta^*(D) \subseteq +\delta(D) \subseteq +\sigma_\delta(D) \subseteq +\sigma_\delta^* (D) \)
(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) \( +\delta^*(D) \subseteq +\sigma_\delta^*(D) \subseteq +\sigma_\partial^*(D) \subseteq +\sigma_\delta^*(D) \)
(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 \( \Phi_{\mathcal{M}(D)} \) 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 \( \alpha_1 \) and \( \alpha_2 \) be annotation functions for a defeasible theory \( D \). We define \( \alpha_1 \subseteq \alpha_2 \) 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 most obvious possibility – a kind of monotonicity – does not hold, as the following example shows.

**Example 4**

Let \( D \) consist of the rules

\[
\begin{align*}
r_1 &: \quad \Rightarrow p \\
r_2 &: \quad \Rightarrow \neg p \\
r_3 &: \quad \Rightarrow q \\
r_4 &: \quad \neg p \Rightarrow \neg q \\
r_5 &: \quad q \Rightarrow s \\
r_6 &: \quad \Rightarrow \neg s
\end{align*}
\]

with \( r_5 > r_6 \).

Let \( \alpha_1 \) map \( q \) in \( r_5 \) to \( \delta \), and \( \alpha_2 \) map \( q \) in \( r_5 \) to \( \partial \) (with all other occurrences mapped to \text{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

\[
C(D) = \{ +dq \mid \mathcal{M}(D) \models_K \text{defeasibly}_d(q), d \in T \} \cup \\
\{ -dq \mid \mathcal{M}(D) \models_K \text{defeasibly}_d(q), d \in T \} \cup \\
\{ +\sigma dq \mid \mathcal{M}(D) \models_K \text{supported}_d(q), d \in T \} \cup \\
\{ -\sigma dq \mid \mathcal{M}(D) \models_K \text{\neg supported}_d(q), d \in T \}
\]

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 \( C(D) \) can be computed in time \( O(|D|) \).

**Different Forms of Failure**

One advantage of the framework of (Maher and Governatori 1999; 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 3-valued stable models (Przymusinski 1990) (also known as partial stable models, but distinct from partial stable models in (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 in (Przymusinski 1990)); the (2-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 \( S \) denote the collection of semantics mentioned above, with the exception of the stable semantics. That is, \( S = \{ \text{partial stable}, \text{well-founded}, \text{regular}, \text{L-stable}, \text{Kunen}, \text{Fitting} \} \).

These semantics (and the stable semantics) are preserved by unfolding (see (Aravindan and Dung 1995; Maher 2017)). Consequently, Theorem 1 extends to the semantics in \( S \):

**Theorem 6**

Let \( D = (F, R, >) \) be a defeasible theory, and \( \alpha \) 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 \( S \in S \). Then

- \( M(\alpha(D)) \models_S \text{defeasibly}_d(q) \) iff \( M_d(D) \models_S \text{defeasibly}_d(q) \)
- \( M(\alpha(D)) \models_S \neg \text{defeasibly}_d(q) \) iff \( M_d(D) \models_S \neg \text{defeasibly}_d(q) \)

and, if \( d \in \{ \delta^*, \delta \} \),

- \( M(\alpha(D)) \models_S \text{supported}_d(q) \) iff \( M_d(D) \models_S \text{supported}_d(q) \)
- \( M(\alpha(D)) \models_S \neg \text{supported}_d(q) \) iff \( M_d(D) \models_S \neg \text{supported}_d(q) \)

More generally, the S-models of \( M(\alpha(D)) \) restricted to defeasibly \( d \) are identical (up to predicate renaming) to the S-models of \( M_d(D) \) 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 \( M_d(D) \) has stable models but \( M(D) \) does not. This, in turn, occurs because \( M(D) \) represents all the inference rules, while \( M_d(D) \) 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.

**Example 7**

Let \( D \) consist of the rules

\[
\begin{align*}
  r_1 : & \quad \Rightarrow \quad p \\
  r_2 : & \quad p, q \Rightarrow \neg p \\
  r_3 : & \quad \Rightarrow \quad q \\
  r_4 : & \quad \Rightarrow \quad q \\
  r_5 : & \quad \Rightarrow \quad \neg q \\
  r_6 : & \quad \Rightarrow \quad \neg q
\end{align*}
\]

with \( r_3 > r_5 \) and \( r_4 > r_6 \).

After unfoldings and simplifications, \( M(D) \) contains

\[
\begin{align*}
  c16 \quad \text{defeasibly}_0(\partial p) : & \quad \text{not overruled}_0(r_1, p). \\
  c17 \quad \text{overruled}_0(r_1, p) : & \quad \text{defeasibly}_0(\partial p), \\
  & \quad \text{defeasibly}_0(\partial q).
\end{align*}
\]
and similar clauses for $\partial^*$ (as well as other clauses).

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

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

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

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 in (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 don’t interact. For example, the proof of $+\delta q$ cannot depend on the proof of $+\delta p$: it can only depend on proofs of $\delta$ conclusions. Within the meta-programming framework of (Maher and Governatori 1999) (Antoniou et al. 2000), 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 don’t 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 (Wan et al. 2009; Wan et al. 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 (Maher 2012; Maher 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 (Maher and Governatori 1999, Maier and Nute 2010, Maher et al. 2011, Grosof 1997, Wan et al. 2009), 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 modality (∇) its conclusion can be proved and the literals ∇q and ⊇q correspond to +∂q and −∂νq. 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 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.

References

Antoniou, G., Billinton, D., Governatori, G., and Maher, M. J. 2000. A flexible framework for defeasible logics. In AAAI/IJAI. AAAI Press / The MIT Press, 405–410.

Antoniou, G., Maher, M. J., and Billinton, D. 2000. Defeasible logic versus logic programming without negation as failure. J. Log. Program. 42, 1, 47–57.

Araujo, C. and Dung, P. M. 1995. On the correctness of unfold/fold transformation of normal and extended logic programs. J. Log. Program. 24, 3, 201–217.

Billinton, D., Antoniou, G., Governatori, G., and Maher, M. J. 2010. An inclusion theorem for defeasible logics. ACM Trans. Comput. Log. 12, 1, 6.

Caminada, M., Sá, S., Alcântara, J., and Dvorák, W. 2015. On the equivalence between logic programming semantics and argumentation semantics. Int. J. Approx. Reasoning 58, 87–111.

Eiter, T., Leone, N., and Saccà, D. 1997. On the partial semantics for disjunctive deductive databases. Ann. Math. Artif. Intell. 19, 1-2, 59–96.

Fitting, M. 1985. A Kripke-Kleene semantics for logic programs. J. Log. Program. 2, 4, 295–312.

Gelder, A. V., Ross, K. A., and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. J. ACM 38, 3, 620–650.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In Proc. 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 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. J. Log. Comput. 14, 5, 675–702.

GOVERNATORI, G., OLIVIERI, F., SCANNAPIECO, S., ROTOLE, 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 The Twenty-Third Annual Conference on Legal Knowledge and Information Systems, R. Winkels, Ed. Frontiers in Artificial Intelligence and Applications, vol. 223. IOS Press, Amsterdam, 57–66.

GROSF, B. N. 1997. Prioritized conflict handling for logic programs. In ILPS. 197–211.

KIFER, M. AND SUBRAHMANIAN, V. S. 1992. Theory of generalized annotated logic programming and its applications. J. Log. Program. 12, 3&4, 335–367.

KUNEN, K. 1987. Negation in logic programming. J. Log. Program. 4, 4, 289–308.

MAHER, M. J. 1988. Correctness of a logic program transformation system. Tech. Rep. RC 13496, IBM T.J. Watson Research Center.

MAHER, M. J. 1990. Reasoning about stable models (and other unstable semantics). Tech. rep.

MAHER, M. J. 2000. A denotational semantics of defeasible logic. In Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings. 209–222.

MAHER, M. J. 2001. Propositional defeasible logic has linear complexity. TPLP 1, 6, 691–711.

MAHER, M. J. 2002. A model-theoretic semantics for defeasible logic. In Paraconsistent Computational Logic, Proceedings of the ICLP 2002 workshop. 67–80.

MAHER, M. J. 2012. Relative expressiveness of defeasible logics. TPLP 12, 4-5, 793–810.

MAHER, M. J. 2013. Relative expressiveness of defeasible logics II. TPLP 13, 4-5, 579–592.

MAHER, M. J. 2017. Relating concrete defeasible reasoning formalisms and abstract argumentation. Fundam. Inform. 153, 1, 1–28.

MAHER, M. J. AND GOVERNATORI, G. 1999. A semantic decomposition of defeasible logics. In AAAI/IAAI. AAAI Press, 299–305.

MAHER, M. J., GOVERNATORI, G., AND LAM, H. P. 2011. Well-founded defeasible logics. Tech. rep.

MAIER, F. 2013. Interdefinability of defeasible logic and logic programming under the well-founded semantics. TPLP 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 The Eleventh 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. Fundam. Inform. 13, 4, 445–463.

SACCA, D. AND ZANIOLO, C. 1990. Stable models and non-determinism in logic programs with negation. In PODS. 205–217.

SEKI, H. 1993. Unfold/fold transformation of general logic programs for the well-founded semantics. J. Log. Program. 16, 1, 5–23.

VERHEIJ, B. 2003. DefLog: on the logical interpretation of prima facie justified assumptions. J. Log. Comput. 13, 3, 319–346.

WAN, H., GROSF, B. N., KIFER, M., FODOR, P., AND LIANG, S. 2009. Logic programming with defaults and argumentation theories. In ICLP. 432–448.

WAN, H., KIFER, M., AND GROSF, 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. J. Comput. Syst. Sci. 49, 2, 334–361.
Inference Rules

Defeasible logics are usually defined via their proof mechanism. Here we present the inference rules for the four defeasible logics we integrate within annotated defeasible logic. Each inference rule is labelled by the kind of conclusions it infers. The presentation is adapted from (Billington et al. 2010). A defeasible logic is determined by the inference rules it allows. For example, $\text{DL}(\partial)$ allows $+\partial$ and $-\partial$, while $\text{DL}(\delta)$ allows $+\delta$, $-\delta$, $+\sigma_s$, and $-\sigma_s$.

A proof $P$ is a sequence of conclusions. The conclusion at position $i$ in the sequence is denoted by $P(i)$, and a prefix of the proof of length $i$ is denoted by $P[1..i]$. The inference rules establish when a conclusion can be drawn at position $i + 1$, given the conclusions already proved ($P[1..i]$). Where $q$ is a literal, $R_{sd}[q]$ denotes the set of strict or defeasible rules with head $q$, while $R[q]$ denotes the set of all rules (including defeaters) with head $q$. For a rule $r$, $A(r)$ denotes the antecedent (or body) of $r$.

\[+\partial\] Infer $P(i + 1) = +\partial q$ if either
\[.1) +\Delta q \in P[1..i]; or\]
\[.2) The following three conditions all hold.
\[.1) \exists r \in R_{sd}[q] \forall a \in A(r), +\partial a \in P[1..i], and\]
\[.2) -\Delta\sim q \in P[1..i], and\]
\[.3) \forall s \in R[\sim q]\] either
\[.1) \exists a \in A(s), -\partial a \in P[1..i]; or\]
\[.2) \exists a \in R_{sd}[q] such that
\[.1) \forall a \in A(t), +\partial a \in P[1..i], and\]
\[.2) t > s.\]
\[+\delta\] Infer $P(i + 1) = +\delta q$ if either
\[.1) +\Delta q \in P[1..i]; or\]
\[.2) The following three conditions all hold.
\[.1) \exists r \in R_{sd}[q] \forall a \in A(r), +\delta a \in P[1..i], and\]
\[.2) -\Delta\sim q \in P[1..i], and\]
\[.3) \forall s \in R[\sim q]\] either
\[.1) \exists a \in A(s), -\delta a \in P[1..i]; or\]
\[.2) \exists a \in R_{sd}[q] such that
\[.1) \forall a \in A(t), +\delta a \in P[1..i], and\]
\[.2) t > s.\]

\[-\partial\] Infer $P(i + 1) = -\partial q$ if
\[.1) -\Delta q \in P[1..i], and\]
\[.2) either\]
\[.1) \forall r \in R_{sd}[q] \exists a \in A(r), -\partial a \in P[1..i]; or\]
\[.2) +\Delta\sim q \in P[1..i]; or\]
\[.3) \exists s \in R[\sim q] such that
\[.1) \forall a \in A(s), +\partial a \in P[1..i], and\]
\[.2) \forall t \in R_{sd}[q] either
\[.1) \exists a \in A(t), -\partial a \in P[1..i]; or\]
\[.2) not(t > s).\]

\[-\delta\] Infer $P(i + 1) = -\delta q$ if
\[.1) -\Delta q \in P[1..i], and\]
\[.2) either\]
\[.1) \forall r \in R_{sd}[q] \exists a \in A(r), -\delta a \in P[1..i]; or\]
\[.2) +\Delta\sim q \in P[1..i]; or\]
\[.3) \exists s \in R[\sim q] such that
\[.1) \forall a \in A(s), +\delta a \in P[1..i], and\]
\[.2) \forall t \in R_{sd}[q] either
\[.1) \exists a \in A(t), -\delta a \in P[1..i]; or\]
\[.2) not(t > s).\]

\[+\sigma_s\] Infer $P(i + 1) = +\sigma_s q$ if either
\[.1) +\Delta q \in P[1..i]; or\]
\[.2) \exists r \in R_{sd}[q] such that
\[.1) \forall a \in A(r), +\sigma_s a \in P[1..i], and\]
\[.2) \forall s \in R[\sim q] either
\[.1) \exists a \in A(s), -\sigma_s a \in P[1..i]; or\]
\[.2) not(s > r).\]

\[\sigma_s\] Infer $P(i + 1) = -\sigma_s q$ if
\[.1) -\Delta q \in P[1..i], and\]
\[.2) \forall r \in R_{sd}[q] either\]
\[.1) \exists a \in A(r), -\sigma_s a \in P[1..i]; or\]
\[.2) \exists s \in R[\sim q] such that
\[.1) \forall a \in A(s), +\sigma_s a \in P[1..i], and\]
\[.2) s > r.\]
The original metaprograms (Maher and Governatori 1999; Antoniou et al. 2000) for the four main forms of defeasibility are outlined below. They consist of clauses defining defeasibly rule and supportive rule (see body of the paper), and a selection of the following clauses for each form of defeasibility.

\[ +\delta^q \text{ Infer } P(i + 1) = +\delta^q \text{ if either } \\
.1) +\Delta q \in P[1..i], \text{ or} \\
.2) \exists r \in R_{sd}[q] \text{ such that } \\
.1) \forall a \in A(r), +\delta^q a \in P[1..i], \text{ and} \\
.2) -\Delta \sim q \in P[1..i], \text{ and} \\
.3) \forall s \in R[\neg q] \text{ either } \\
.1) \exists a \in A(s), -\delta^q a \in P[1..i]; \text{ or} \\
.2) r > s. \]

\[ -\delta^q \text{ Infer } P(i + 1) = -\delta^q \text{ if } \\
.1) -\Delta q \in P[1..i], \text{ and} \\
.2) \forall r \in R_{sd}[q] \text{ either } \\
.1) \exists a \in A(r), -\delta^q a \in P[1..i]; \text{ or} \\
.2) +\Delta \sim q \in P[1..i]; \text{ or} \\
.3) \exists s \in R[\neg q] \text{ such that } \\
.1) \forall a \in A(s), +\delta^q a \in P[1..i], \text{ and} \\
.2) not(r > s). \]

\[ +\delta^*_q \text{ Infer } P(i + 1) = +\delta^*_q \text{ if either } \\
.1) +\Delta q \in P[1..i], \text{ or} \\
.2) \exists r \in R_{sd}[q] \text{ such that } \\
.1) \forall a \in A(r), +\delta^*_q a \in P[1..i], \text{ and} \\
.2) -\Delta \sim q \in P[1..i], \text{ and} \\
.3) \forall s \in R[\neg q] \text{ either } \\
.1) \exists a \in A(s), -\delta^*_q a \in P[1..i]; \text{ or} \\
.2) r > s. \]

\[ -\delta^*_q \text{ Infer } P(i + 1) = -\delta^*_q \text{ if } \\
.1) -\Delta q \in P[1..i], \text{ and} \\
.2) \forall r \in R_{sd}[q] \text{ either } \\
.1) \exists a \in A(r), -\delta^*_q a \in P[1..i]; \text{ or} \\
.2) +\Delta \sim q \in P[1..i]; \text{ or} \\
.3) \exists s \in R[\neg q] \text{ such that } \\
.1) \forall a \in A(s), +\delta^*_q a \in P[1..i], \text{ and} \\
.2) not(r > s). \]

**Original Meta-programs**

The original metaprograms (Maher and Governatori 1999; Antoniou et al. 2000) for the four main forms of defeasibility are outlined below. They consist of clauses defining definitely, clauses defining rule and supportive rule (see body of the paper), and a selection of the following clauses for each form of defeasibility.

\[ \text{c21 defeasibly}(X) :: \\
\text{ definitely}(X). \]

\[ \text{c22 defeasibly}(X) :: \\
\text{ not definitely}(!X), \\
\text{ supportive_rule}(R, X, \{Y_1, \ldots, Y_n\}), \\
\text{ defeasibly}(Y_1), \ldots, \text{ defeasibly}(Y_n), \\
\text{ not overruled}(R, X). \]

\[ \text{c23 overruled}(R, X) :: \\
\text{ rule}(S, \sim X, \{U_1, \ldots, U_n\}), \\
\text{ defeasibly}(U_1), \ldots, \text{ defeasibly}(U_n), \\
\text{ not defeated}(S, \sim X). \]

\[ \text{c24 defeated}(S, \sim X) :: \\
\text{ sup}(T, S), \\
\text{ supportive_rule}(T, X, \{V_1, \ldots, V_n\}), \\
\text{ defeasibly}(V_1), \ldots, \text{ defeasibly}(V_n). \]
Annotated Defeasible Logic

\[\text{c25} \quad \text{supported}(X) : \]
\[\text{definitely}(X).\]

\[\text{c26} \quad \text{supported}(X) : \]
\[\text{supportive_rule}(R, X, [Y_1, \ldots, Y_n]),\]
\[\text{supported}(Y_1), \ldots, \text{supported}(Y_n),\]
\[\text{not beaten}(R, X).\]

\[\text{c27} \quad \text{beaten}(R, X) : \]
\[\text{rule}(S, \neg X, [W_1, \ldots, W_n]),\]
\[\text{defeasibly}(W_1), \ldots, \text{defeasibly}(W_n),\]
\[\text{sup}(S, R).\]

\[\text{c28} \quad \text{overruled}(R, X) : \]
\[\text{rule}(S, \neg X, [U_1, \ldots, U_n]),\]
\[\text{supported}(U_1), \ldots, \text{supported}(U_n),\]
\[\text{not defeated}(S, \neg X).\]

\[\text{c29} \quad \text{overruled}(R, X) : \]
\[\text{rule}(S, \neg X, [U_1, \ldots, U_n]),\]
\[\text{defeasibly}(U_1), \ldots, \text{defeasibly}(U_n),\]
\[\text{not sup}(R, S).\]

The selection of clauses for each meta-program is as follows:
\(\mathcal{M}_\partial\) contains the clauses \[\text{c21} - \text{c24}\].
\(\mathcal{M}_\delta\) contains the clauses \[\text{c21} - \text{c22}, \text{c28}, \text{c24}, \text{c25} - \text{c27}\].
\(\mathcal{M}_\partial^*\) contains the clauses \[\text{c21} - \text{c22}, \text{c29}\].
\(\mathcal{M}_\delta^*\) consists of the clauses \[\text{c21} - \text{c22}, \text{c30}, \text{c25} - \text{c27}\].

Proofs of results

We present (sketches of) proofs for the results in the paper.

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

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

- \(\mathcal{M}(\alpha(D)) \models_K \text{defeasibly}_d(q)\) if\ h\ \(\mathcal{M}_d(D) \models_K \text{defeasibly}(q)\)
- \(\mathcal{M}(\alpha(D)) \models_K \neg\text{defeasibly}_d(q)\) if\ h\ \(\mathcal{M}_d(D) \models_K \neg\text{defeasibly}(q)\)

Furthermore, if \(d \in \{\delta^*, \delta\}\),

- \(\mathcal{M}(\alpha(D)) \models_K \text{supported}_d(q)\) if\ h\ \(\mathcal{M}_d(D) \models_K \text{supported}(q)\)
- \(\mathcal{M}(\alpha(D)) \models_K \neg\text{supported}_d(q)\) if\ h\ \(\mathcal{M}_d(D) \models_K \neg\text{supported}(q)\)
Proof
(Sketch) The proof of this theorem is similar for each tag $d$. For brevity, we only provide the details for $\delta$. The proof is based on unfolding $M(\alpha(D))$ until it has essentially the same form as an unfolding of $M_\delta(D)$. The form of unfolding we use uses clauses from the current program, and may be applied as long as no clause is used to unfold an atom in its own body. Such unfolding preserves the Kunen semantics (i.e. 3-valued models of the Clark-completion) of a logic program by essentially the same argument that it preserves the 2-valued models (Maher 1988). Clauses $c1$, $c2$, and $c5$ are the same in both $M$ and $M_\delta$, so we will essentially ignore them.

In both $M(\alpha(D))$ and $M_\delta(D)$ we unfold all occurrences of the predicates used to represent the annotated defeasible theory, and rule and supportive rule. Then, in $M(\alpha(D))$, we unfold all occurrences of the predicates specifying the type of each tag: ambiguity propagating, ambiguity blocking, proof tag, team, and indiv. At this point clauses derived from $c6$, $c8$ are ground, while clauses derived from $c10$ only have a single, unused variable $X$ in their heads. Similarly, clauses derived from $c14$ and $c15$ are ground.

Then unfold all defeasibly$_Z$(free $L$) atoms. This will not result in a clause unfolding itself: in $c3$ because $Z$ is not free, and in $c5$ because $Y$ is not free. Similarly, we unfold all supported$_Z$(free $L$) atoms. As a result, free only occurs in the head of clauses derived from $c3$ and $c11$.

Finally, unfold all $M(\alpha(D))$ and defeated atoms in $M_\delta(D)$. At this stage, clauses derived from $M_\delta(D)$ are essentially the same as some of the clauses derived from $M(\alpha(D))$ with subscript $\delta$; the differences are in the name/arity of predicates (e.g., defeasibly versus defeasibly$_\delta$) and the presence of rules with heads of the form defeasibly$_\delta$(free $L$), supported$_\delta$(free $L$), defeasibly$_\delta$(fail $L$) or supported$_\delta$(fail $L$). However, no atom with subscript $\delta$ depends on a predicate with a different subscript, nor on clauses with free or fail in the head. Hence, the consequences of $M(D)$ of the form defeasibly$_\delta$(q) and supported$_\delta$(q) are unaffected by the presence or absence of such rules, and so we delete them all.

Consequently, the two transformed programs are the same (modulo predicate renaming), and hence have the same conclusions. Since the transformations preserve the semantics of the programs, the result follows. \qed

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

- $M(\alpha_F(D)) \models_K$ defeasibly$_d$(q) iff $D \vdash dq$
- $M(\alpha_F(D)) \models_K$ \neg defeasibly$_d$(q) iff $D \vdash \neg dq$

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

- $M(\alpha_F(D)) \models_K$ supported$_d$(q) iff $D \vdash +\sigma_dq$
- $M(\alpha_F(D)) \models_K$ \neg supported$_d$(q) iff $D \vdash \neg \sigma_dq$

Proof
The corollary follows from applying the previous theorem for each tag $d$ to the case where $\alpha$ is the free annotation function, and the correctness of the individual meta-programs. \qed
Theorem 3 (Inclusion Theorem)

Let $D$ be an annotated defeasible theory.

(a) $+\Delta(D) \subseteq +\delta'(D) \subseteq +\delta(D) \subseteq +\delta_\beta(D) \subseteq +\delta_\beta(D)$
(b) $-\delta_\beta(D) \subseteq -\delta_\beta(D) \subseteq -\delta(D) \subseteq -\delta'(D) \subseteq -\Delta(D)$
(c) $+\delta(D) \subseteq +\delta_\beta(D) \subseteq +\delta_\beta(D)$
(d) $-\delta_\beta(D) \subseteq -\delta_\beta(D) \subseteq -\delta(D)$
(e) $+\delta'(D) \subseteq +\delta'(D) \subseteq +\delta_\beta(D) \subseteq +\delta_\beta(D)$
(f) $-\delta_\beta(D) \subseteq -\delta_\beta(D) \subseteq -\delta'(D) \subseteq -\delta'(D)$

Proof

(Sketch) Let $\Phi = \Phi_{\mathcal{M}(D)}$ be Fitting’s semantic function for the logic program $\mathcal{M}(D)$ [Fitting 1985]. Recall that Kunen’s semantics is the set of all consequences of $\Phi \uparrow n$ for any finite $n$. We prove the containments by induction on the iteration of $\Phi$. For brevity, we omit parts of the induction hypothesis related to proving (a). We also omit the parts related to (b), (d) and (f) since, by the Principle of Strong Negation (Antoniou et al. 2000), their statements and proof are symmetric to those for the positive conclusions. The induction hypothesis contains

\[
\begin{align*}
\text{supported}_\beta \subseteq \text{supported}_\beta \\
\text{beaten}_\beta \subseteq \text{beaten}_\beta \\
\text{beaten}_\beta \subseteq \text{beaten}_\beta \\
\text{overruled}_\beta \subseteq \text{overruled}_\beta \\
\text{supported}_\beta \subseteq \text{supported}_\beta \\
\text{overruled}_\beta \subseteq \text{overruled}_\beta
\end{align*}
\]

Clearly this statement holds in the empty interpretation. It is mostly straightforward to show that if the induction hypothesis holds in $\Phi \uparrow n$ then it holds in $\Phi \uparrow n+1$. For example, consider the first two containments in the induction hypothesis. If they hold in $\Phi \uparrow n$ (and also $\text{defeasibly}_\beta \subseteq \text{defeasibly}_\beta$ holds) then, applying clause [15] the second containment holds in $\Phi \uparrow n+1$ and, applying clause [14] the first containment holds in $\Phi \uparrow n+1$. To address fail-expressions we also need the corresponding versions of these containments and arguments for negative conclusions.

One containment, $\text{defeasibly}_\beta \subseteq \text{supported}_\beta$, is not easily proved by induction, but it has a direct proof. For a set $S = \Phi \uparrow n$, if $\text{defeasibly}_\beta(x) \in \Phi(S)$ then there is a supportive rule $r$ whose body literals are defeasibly true in $S$ (i.e. $\text{defeasibly}_\beta(w_i) \in S$) and not $\text{overruled}_\beta(r, x) \in S$. Now not $\text{overruled}_\beta(r, x) \in S$ only if, for every rule $s$ for $\sim x$ whose body literals are defeasibly true in $S$, there is a supportive rule $t$ whose body literals are defeasibly true in $S$ and $t > s$. Since $D$ is finite and $>\leq$ is acyclic, for some such $t$, for every such $s$, $s \not> t$. This $t$ can now be used as $r$ in clauses [14] and [15] to show that $\text{supported}_\beta(x) \in \Phi(S)$.

As mentioned in the body of the paper, it is straightforward to compute the consequences of an annotated defeasible theory in quadratic time. We outline the proof.

Proposition 4

Let $D$ be an annotated defeasible theory, and $|D|$ be the number of symbols in $D$. Then the set of consequences $C(D)$ can be computed in time $O(|D|^2)$. 
**Proof**

(Sketch) Consider the grounding of the clauses, by unfolding with the input representation of the defeasible theory and related facts, and the worst-case (i.e. maximum) size of the result. Unfolding with facts like `proof` produces an increase in rules by a constant factor, because the number of tags is fixed. Unfolding clauses for `rule` etc. produces a set of ground instances linear in the size of rules in $D$. For clauses $c_1$ and $c_2$, the size of ground instances is proportional to the number of tags in $D$. For clauses $c_6, c_{14}, c_{15}$, the size of ground instances is proportional to the size of rules in $D$. The size of ground instances of clauses $c_{10}$ is proportional to the number of superiority statements in $D$. The size of ground instances of clauses $c_9$ is proportional to the product of the number of superiority statements and the maximum size of rules in $D$.

Thus the size of all ground clauses is bounded above by $|D|^2$. The ground rules form an essentially propositional logic program. Computing the consequences of a propositional logic program under the Kunen semantics is linear in the size of the program. Consequently, the cost of computing the conclusions is $O(|D|^2)$. □

Recall that $S = \{\text{partial stable}, \text{well–founded}, \text{regular}, \text{L–stable}, \text{Kunen, Fitting} \}$ is a set of semantics. These semantics (and the stable semantics) are preserved by unfolding (with the Kunen semantics requiring the restriction on a rule unfolding itself). This was established for the well-founded (Seki 1993, Aravindan and Dung 1995) and stable models (Maher 1990, Aravindan and Dung 1995), and in (Maher 2017) for the partial stable models and the L-stable models. For the Kunen and Fitting semantics it follows the same proof as in (Maher 1988) for the 2-valued Clark completion semantics. Consequently, Theorem 1 extends to the semantics in $S$:

**Theorem 6**

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

- $\mathcal{M}(\alpha(D)) \models_S \text{feasibly}^d(q)$ iff $\mathcal{M}_d(D) \models_S \text{feasibly}(q)$
- $\mathcal{M}(\alpha(D)) \models_S \neg\text{feasibly}^d(q)$ iff $\mathcal{M}_d(D) \models_S \neg\text{feasibly}(q)$

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

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

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

**Proof**

The proof of Theorem 1 also applies to this theorem, since unfolding (without self-unfolding) preserves models for all semantics in $S$ (see Theorem 3.2 of (Maher 2017)), as does deletion of irrelevant clauses. □
Examples

We present some counterexamples, to show that Figure 1 does not omit any containments and that all the containments are strict. For these examples we do not need to use any annotations: they equally apply to (unannotated) defeasible theories, and we present them in that form.

There are four possible containments we must show do not hold: $\delta \nsubseteq \partial^*$, $\sigma_\delta \nsubseteq \sigma_{\partial^*}$, $\partial \nsubseteq \sigma_\delta$, and $\delta \nsubseteq \sigma_{\partial^*}$. We have two examples that demonstrate these four points.

Example 7

Let the defeasible theory $D$ consist of the rules

\begin{align*}
  r_1 & : \quad \Rightarrow p \\
  r_2 & : \quad \Rightarrow \neg p \\
  r_3 & : \quad \Rightarrow q \\
  r_4 & : \quad \neg p \Rightarrow \neg q \\
  r_5 & : \quad q \Rightarrow s \\
  r_6 & : \quad \Rightarrow \neg s
\end{align*}

with $r_5 > r_6$.

Rules $r_1 \cdot r_4$ are a standard example distinguishing ambiguity blocking and propagating behaviours. $+\partial^* q$ and $-\delta q$ can be concluded. Thus, $\delta \nsubseteq \partial^*$. In addition, we conclude $-\sigma_{\partial^*} s$ and $+\sigma_\delta s$. Thus, $\sigma_\delta \nsubseteq \sigma_{\partial^*}$.

Now we show that $\partial \nsubseteq \sigma_\delta$ and $\delta \nsubseteq \sigma_{\partial^*}$.

Example 8

Consider the following defeasible theory $D$:

\begin{align*}
  r_1 & : \quad p \Rightarrow \neg q \\
  r_2 & : \quad q \Rightarrow s \\
  r_3 & : \quad \Rightarrow \neg s \\
  r_4 & : \quad \Rightarrow \neg p \\
  r_5 & : \quad \Rightarrow p \\
  r_6 & : \quad \Rightarrow p \\
  r_7 & : \quad \Rightarrow \neg p \\
  r_8 & : \quad \Rightarrow \neg p
\end{align*}

with $r_1 > r_2$, $r_5 > r_7$, $r_6 > r_8$.

Then we have $+\delta p$ and $-\partial^* p$. Consequently, we have $-\sigma_\delta q$ and $+\partial^* q$. Hence, $\partial \nsubseteq \sigma_\delta$. Furthermore, we have $+\delta s$ and $-\partial^* s$. Hence $\delta \nsubseteq \sigma_{\partial^*}$.

Hence, there are no containments missing from Figure 1.

That the containments in the top row of Figure 1 are strict was mostly established in (Billington et al. 2010). The strictness of containments between forms of support follows straightforwardly from the strictness of containment for the corresponding forms of defeasibility. For the remaining containments, consider the following example.

Example 9

Consider the following defeasible theory $D$:

\begin{align*}
  r_1 & : \quad p \Rightarrow q \\
  r_2 & : \quad \Rightarrow \neg q \\
  r_3 & : \quad \Rightarrow p \\
  r_4 & : \quad \Rightarrow \neg p
\end{align*}

We have $-\partial^* p$ (and $-\partial^* \neg p$) but $+\sigma_\delta p$. Consequently, we have $+\partial^* q$ but $-\delta^* q$, showing that $\delta^* \subset \partial^*$ on $D$. 
Note also that we have conclusions $-dp$ and $+\sigma_d p$ for any defeasible proof tag $d$. Hence $+\partial \subset \sigma_{\partial}$ and $+\partial^* \subset \sigma_{\partial^*}$.

Hence all the containments in Figure 1 are strict.