On a Flexible Representation for Defeasible Reasoning Variants

Abdelraouf Hecham, Pierre Bisquert, Madalina Croitoru

On a Flexible Representation for Defeasible Reasoning Variants

Abdelraouf Hecham
University of Montpellier
Montpellier, France
hecham@lirmm.fr

Pierre Bisquert
IATE, INRA
Montpellier, France
pierre.bisquert@inra.fr

Madalina Croitoru
University of Montpellier
Montpellier, France
croitoru@lirmm.fr

ABSTRACT

We propose *Statement Graphs* (SG), a new logical formalism for defeasible reasoning based on argumentation. Using a flexible labeling function, SGs can capture the variants of defeasible reasoning (ambiguity blocking or propagating, with or without team defeat, and circular reasoning). We evaluate our approach with respect to human reasoning and propose a working first order defeasible reasoning tool that, compared to the state of the art, has richer expressivity at no added computational cost. Such tool could be of great practical use in decision making projects such as H2020 NoAW.

KEYWORDS

Defeasible Reasoning; Defeasible Logics; Existential Rules

ACM Reference Format:
Abdelraouf Hecham, Pierre Bisquert, and Madalina Croitoru. 2018. On a Flexible Representation for Defeasible Reasoning Variants. In , , , IFAAMAS, 9 pages.

1 INTRODUCTION

Defeasible reasoning [25] is used to evaluate claims or statements in an inconsistent setting. It has been successfully applied in many multi-agent domains such as legal reasoning [3], business rules [24], contracting [17], planning [15], agent negotiations [12], inconsistency management [23], etc. Defeasible reasoning can also be used for reasoning with uncertain rules in food science. In the EU H2020 NoAW project we are interested in reasoning logically about how to manage waste from wine by products. Such rules, elicited by experts, non experts, consumers etc via online surveys have to be put together and used as a whole for decision making. Unfortunately, there is no universally valid way to reason defeasibly. An inherent characteristic of defeasible reasoning is its systematic reliance on a set of intuitions and rules of thumb, which have been long debated between logicians [1, 19, 22, 26]. For example, could an information derived from a contested claim be used to contest another claim (i.e. ambiguity handling)? Could "chains" of reasoning for the same claim be combined to defend against challenging statements (i.e. team defeat)? Is circular reasoning allowed? etc.

The main available[1] defeasible reasoning tools are ASPIC+ [27], DEFT [18], DeLP [14], DR-DEVICE [5], and Flora-2 [31]. Table 1 shows that no tools can support all features.

Existing literature has established the link between argumentation and defeasible reasoning via grounded semantics [13, 28] and

[1] Available and runnable as of November 2017.

.

Table 1: Defeasible features supported by tools.

Tool	Blocking	Propagating	Team Defeat	No Team Defeat
ASPIC+	-	✓	-	✓
DEFT	-	✓	-	✓
DeLP	-	✓	-	✓
DR-DEVICE	✓	-	✓	-
Flora-2	✓	-	✓	-

Dialectical Trees [14]. Such approaches only allow for ambiguity propagation without team defeat [16, 26].

In this paper we propose a new logical formalism called *Statement Graph* (SGs) that *captures all features showed in Table 1* via a flexible labelling function. The SG can be seen as a generalisation of Abstract Dialectical Frameworks (ADF) [8] that enrich ADF acceptance condition.

After introducing SGs in Section 3 we show in Section 4 how the flexible labelling function of SG can capture ambiguity blocking (Section 4.1), ambiguity propagating (Section 4.2), team defeat (Section 4.3) and circular reasoning (Section 4.4). In Section 5 we evaluate the practical applicability of SGs. We demonstrate (Section 5.2) certain features of human reasoning empirically demonstrated by psychologists. Furthermore, we provide a tool (Section 5.1) that implements SGs and, despite its higher expressivity than the tools in Table 1, performs better or equally good.

2 BACKGROUND NOTIONS

Language. We consider a propositional language of literals and the connectives ($\wedge, \rightarrow, \Rightarrow, \rightsquigarrow$). A literal is either an atom (an atomic formula) or the complement of an atom. The complement of atom f is denoted $\overline{f}$. $\top$ and $\bot$ are considered atoms. Given Φ a finite non-empty conjunction of literals and a literal ψ, a rule r is a formula of the form $\Phi \Rrightarrow \psi$ such that $\Rrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$. We call Φ the body of r denoted $\mathcal{B}(r)$ and ψ the head of r denoted $\mathcal{H}(r)$. The set $\mathcal{R}$ of rules is composed of:

1. $\mathcal{R}_{\rightarrow}$ the set of strict rules (of the form $\Phi \rightarrow \psi$) expressing undeniable implications i.e. if Φ then definitely ψ.
2. $\mathcal{R}_{\Rightarrow}$ the set of defeasible rules (of the form $\Phi \Rightarrow \psi$) expressing a weaker implication i.e. if Φ then generally ψ.
3. $\mathcal{R}_{\rightsquigarrow}$ the set of defeater rules (of the form $\Phi \rightsquigarrow \psi$) used to prevent a complement of a conclusion i.e. if Φ then $\overline{\psi}$ should not be concluded. It does not imply that ψ is concluded.

Given a literal f, the rule of the form $\top \rightarrow f$ or $\top \Rightarrow f$ is called a fact rule. A derivation for f is a sequence of strict and defeasible rules that starts from a fact rule and end with a rule $r \in \mathcal{R}$ s.t. $\mathcal{H}(r) = f$. A strict derivation contains only strict rules.

A defeasible knowledge base is a tuple $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, \succ)$ where $\mathcal{F}$ is a set of fact rules, $\mathcal{R}$ is a set of strict, defeasible and defeater rules, and $\succ$ is a superiority relation on $\mathcal{R}$. A superiority relation is an acyclic relation $>$ (i.e. the transitive closure of $>$ is irreflexive). If $r_1 > r_2$, then r_1 is called superior to r_2, and r_2 inferior to r_1. This expresses that r_1 may override r_2. A query (also called *claim*) on a knowledge base $\mathcal{KB}$ is a conjunction of literals.

Defeasible reasoning. To reason defeasibly about a conclusion, all chains of rule applications (also called arguments) reaching that conclusion must be evaluated along with any conflict (i.e. attack) that arises from other chains of rules. This can be achieved using two kinds of approaches: (1) approaches based on the evaluation of arguments during their construction, such as defeasible logics [6, 25] and (2) approaches using the idea of extensions, where arguments are built then evaluated at a later stage; these encapsulate argumentation-based techniques [13, 14].

An argument is a minimal sequence of rule applications from a fact rule to a rule with a literal in its head called the conclusion of that argument. An argument arg attacks another argument arg' on a literal f if its conclusion is $\overline{f}$ and f is one of the literals that appear in arg'. We say that arg defeats arg' if the rule in arg generating $\overline{f}$ is not inferior to the rule in arg' generating f.

Ambiguity Handling. A literal f is **ambiguous** if there is an undefeated argument for f and another undefeated argument for $\overline{f}$ and the superiority relation does not state which argument is superior, as shown in Example 1.

Example 1. *The following defeasible knowledge base $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, \varnothing)$ describes a situation where a piece of evidence 'A' suggests that a defendant is responsible while an evidence 'B' indicates that he is not responsible; Both evidences are equally reliable. A defendant is presumed not guilty unless responsibility has been proven:*

- $\mathcal{F} = \{\top \Rightarrow \overline{guilty}, \top \rightarrow evidA, \top \rightarrow evidB\}$
- $\mathcal{R} = \{r_{d1} : evidA \Rightarrow responsible,$
  $r_{d2} : evidB \Rightarrow \overline{responsible},$
  $r_{d3} : responsible \Rightarrow guilty\}.$

Evaluating the query $q = \overline{guilty}$ (i.e. is the defendant not guilty?) requires the construction of all arguments for and against this literal.

- $arg_1 = \langle \top \Rightarrow \overline{guilty} \rangle$.
- $arg_2 = \langle \top \rightarrow evidA,\ evidA \Rightarrow responsible \rangle$.
- $arg_3 = \langle \top \rightarrow evidB,\ evidB \Rightarrow \overline{responsible} \rangle$.
- $arg_4 = \langle \top \rightarrow evidA,\ r_{d1},\ responsible \Rightarrow guilty \rangle$.

arg_2 and arg_3 attack and defeat each other as no argument is superior to the other, therefore their conclusions "*responsible*" and "$\overline{responsible}$" are said to be *ambiguous*. In an **ambiguity blocking** setting (such as Nute's defeasible logic [25]), the ambiguity of "*responsible*" blocks (forbids) any ambiguity derived from it, meaning that all arguments containing "*responsible*" cannot be used to attack other arguments (they are considered as defeated). Therefore arg_1 is uncontested and the answer to q is 'true' (i.e. $\mathcal{KB} \vDash_{block} \overline{guilty}$, where $\vDash_{block}$ denotes entailment in ambiguity blocking).

On the other hand, in an **ambiguity propagating** setting (such as grounded semantics and dialectical trees [16]), the ambiguity of "*responsible*" is propagated to "$\overline{guilty}$" because "*guilty*" can be derived (arg_4 is allowed to defeat arg_1), hence, the answer to the query q is 'false' (i.e. $\mathcal{KB} \nvDash_{prop} \overline{guilty}$, where $\vDash_{prop}$ denotes entailment in ambiguity propagating).

Ambiguity propagation results in fewer conclusions (as more ambiguities are allowed) and may be preferable when the cost of an incorrect conclusion is high. *Ambiguity blocking* may be appropriate in situations where contested claims cannot be used to contest other claims (e.g. in the legal domain) [19].

Team Defeat. The absence of team defeat means that for an argument to be undefeated it has to single-handedly defeat all its attackers, as shown in Example 2.

Example 2. *Generally, animals do not fly unless they are birds. Also, penguins do not fly except magical ones. 'Tweety' is an animal, a bird, and a magical penguin. Can 'Tweety' fly? $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, >)$:*

- $\mathcal{F} = \{\top \Rightarrow animal, \top \Rightarrow bird, \top \Rightarrow penguin, \top \Rightarrow magical\}$.
- $\mathcal{R} = \{r_{d1} : animal \Rightarrow \overline{fly},\ r_{d2} : bird \Rightarrow fly,\ r_{d3} : penguin \Rightarrow \overline{fly},\ r_{d4} : magical \wedge penguin \Rightarrow fly\}$.
- $(r_{d2} > r_{d1})$, $(r_{d4} > r_{d3})$.

The query is $q = fly$. **In the absence of team defeat**, the answer to q is **'false'** (i.e. $\mathcal{KB} \nvDash^{noTD} fly$, where $\vDash^{noTD}$ denotes entailment in defeasible reasoning without team defeat) because there is no chain of reasoning for "fly" that can defend itself from all attacks: even if r_{d2} defends itself from r_{d1} (because $r_{d2} > r_{d1}$), it does not defend against r_{d3} (since $r_{d2} \not> r_{d3}$), and the same applies for r_{d4}: it defends against r_{d3} but fails against r_{d1} because $r_{d4} \not> r_{d1}$. **If team defeat is allowed** then the answer to q is **'true'** (i.e. $\mathcal{KB} \vDash^{TD} fly$, where $\vDash^{TD}$ denotes entailment in defeasible reasoning with team defeat) because all attacks are defeated: r_{d1} is defeated by r_{d2} ($r_{d2} > r_{d1}$) and r_{d3} is defeated by r_{d4} ($r_{d4} > r_{d3}$). Argumentation-based techniques for defeasible reasoning do not allow for team defeat, whereas defeasible logics do [2, 26].

3 THE STATEMENT GRAPH

A Statement Graph (SG) can be seen as a graph representation of the reasoning process happening inside a knowledge base. It is built using logical building blocks (called statements) that describe a situation (premises) and a rule that can be applied on that situation.

Definition 1 (Statement). *A statement is a tuple $\mathbb{s} = (\Phi, r)$ where Φ is a (possibly empty) set of literals (called premises) and $r \in \mathcal{R} \cup \{\top\} \cup \{\emptyset\}$ is either a rule, the Top literal or an empty set. A statement can be of three types:*

(1) A 'claim statement' for a claim C of the form $(C, \emptyset)$.
(2) A 'Top statement' of the form $(\emptyset, \top)$.
(3) A 'rule application statement' of the form (Φ, r) such that $\mathcal{B}(r) = \Phi$.

We denote by $Rule(\mathbb{s}) = r$ and $Premise(\mathbb{s}) = \Phi$ the rule and premises of a statement $\mathbb{s}$ respectively.

A statement $\mathbb{s}_1$ can *attack* (or *support*) a statement $\mathbb{s}_2$ if it provides a justification against (or for) the premises of $\mathbb{s}_2$.

Definition 2 (Attack and Support). *Given two statements $\mathbb{s}_1 = (\Phi_1, r_1)$ and $\mathbb{s}_2 = (\Phi_2, r_2)$:*

- *$\mathbb{s}_1$ supports $\mathbb{s}_2$ iff: $\exists f \in \Phi_2$ s.t. $\mathcal{H}(r_1) = f$ and $r_1 \notin \mathcal{R}_{\rightsquigarrow}$. (we say that $\mathbb{s}_1$ supports $\mathbb{s}_2$ on f).*
- *$\mathbb{s}_1$ attacks $\mathbb{s}_2$ iff:*

(1) *Either* $\exists f \in \Phi_2$ *s.t.* $\mathcal{H}(r_1) = \overline{f}$ *and* $r_1 \notin \mathcal{R}_{\rightsquigarrow}$. *(we say that* $\mathbb{s}_1$ *undercuts* $\mathbb{s}_2$ *on* f*).*

(2) *Or* $r_1 \in \mathcal{R}_{\rightsquigarrow}$ *and* $\mathcal{H}(r_1) = \overline{\mathcal{H}(r_2)}$ *(we say that* $\mathbb{s}_1$ *attacks the rule application of* $\mathbb{s}_2$*).*

Statements are generated from a knowledge base, they can be structured in a graph according to the support and attack relations they have between each other.

Definition 3 (Statement Graph). *A Statement Graph of the knowledge base* $\mathcal{KB}$ *is a directed graph* $\mathbb{SG}_{\mathcal{KB}} = (\mathcal{V}, \mathcal{E}_A, \mathcal{E}_S)$*:*

- $\mathcal{V}$ *is the set of statements generated from* $\mathcal{KB}$.
- $\mathcal{E}_S \subseteq \mathcal{V} \times \mathcal{V}$ *is the set of support edges. There is a support edge* $e = (\mathbb{s}_1, \mathbb{s}_2) \in \mathcal{E}_S$ *iff* $\mathbb{s}_1$ *supports* $\mathbb{s}_2$.
- $\mathcal{E}_A \subseteq \mathcal{V} \times \mathcal{V}$ *is the set of attack edges. There is an attack edge* $e = (\mathbb{s}_1, \mathbb{s}_2) \in \mathcal{E}_A$ *iff the statement* $\mathbb{s}_1$ *attacks* $\mathbb{s}_2$.

For an edge $e = (\mathbb{s}_1, \mathbb{s}_2)$*, we denote* $\mathbb{s}_1$ *by* $Source(e)$ *and* $\mathbb{s}_2$ *by* $Target(e)$.

For a statement $\mathbb{s}$ *we denote its incoming attack edges by* $\mathcal{E}_A^-(\mathbb{s})=\{e \in \mathcal{E}_A | Target(e) = \mathbb{s}\}$ *and its incoming support edges by* $\mathcal{E}_S^-(\mathbb{s})=\{e \in \mathcal{E}_S | Target(e) = \mathbb{s}\}$*. We also denote its outgoing attack edges by* $\mathcal{E}_A^+(\mathbb{s})=\{e \in \mathcal{E}_A | Source(e) = \mathbb{s}\}$ *and outgoing support edges by* $\mathcal{E}_S^+(\mathbb{s})=\{e \in \mathcal{E}_S | Source(e)=\mathbb{s}\}$.

An SG can be constructed in two ways, either by generating all possible statements in a knowledge base then adding the attack and support edges (in order to have a general overview of the knowledge base), or by starting from a specific statement and generating recursively all statements that support or attack it until no other statement can be generated, as shown in Example 3.

Example 3. *Consider the knowledge base in Example 1. A SG for the claim statements* $(\{guilty\}, \emptyset)$ *and* $(\{\overline{guilty}\}, \emptyset)$ *is shown in Figure 1 (support edges depicted by dashed arrows).*

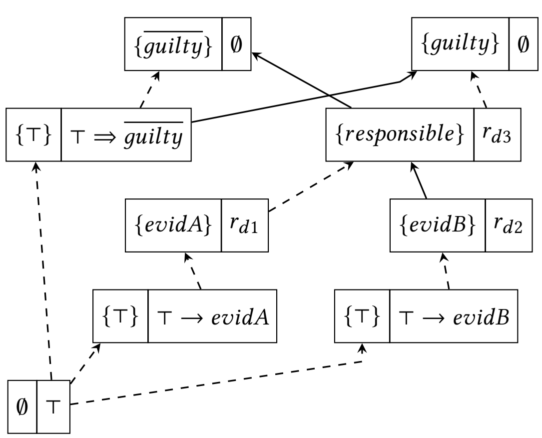

Figure 1: SG generated from $\mathcal{KB}$ in Example 1.

An SG provides statements and edges with a label using a *labeling function* that starts from the Top statement and propagates labels to the other statements. Query answering can then be determined based on the label of the claim statement for a query. This can be seen as a logic-based instantiation of ADFs (Abstract Dialectical Frameworks) [8] but rather than using a boolean acceptance condition, SG uses a labeling function.

Definition 4 (Labeling Function). *A labeling function applied to a statement graph is a function* $St : \mathcal{V} \cup \mathcal{E}_A \cup \mathcal{E}_S \rightarrow Label$ *that takes as input a statement* $\mathbb{s} \in \mathcal{V}$ *or an edge* $e \in \mathcal{E}_A \cup \mathcal{E}_S$ *and returns a label in* $Label = \{IN_{str}, IN_{def}, OUT_{str}, OUT_{def}, AMBIG, UNSUP\}$.

The intuition behind these labels is as follows:

- IN_{str} indicates that the statement is accepted and its rule can be strictly applied based on strictly accepted premises.
- IN_{def} indicates that the statement is accepted and its rule can be defeasibly applied based on strictly or defeasibly accepted premises.
- OUT_{str} and OUT_{def} indicate that the statement is not accepted because its rule or premises have been strictly or defeasibly defeated respectively.
- AMBIG indicates that the statement's rule or premises are challenged and the superiority relation cannot be used to determine if it is accepted or not.
- UNSUP indicates that the statement's premises are not supported by facts.

A statement is given a label based on its incoming edges and their labels. The notion of *complete support* describes the situation where a statement has a support edge for each one of its premises.

Definition 5 (Complete Support). *A complete support for a statement* $\mathbb{s}$ *is a set of support edges denoted* $\mathcal{E}_{CS}^{\mathbb{s}}$ *such that:*

- $\forall f \in Premise(\mathbb{s}), \exists e \in \mathcal{E}_{CS}^{\mathbb{s}}$ *s.t.* $Source(e)$ *supports* $\mathbb{s}$ *on* f.
- $\nexists S'$ *s.t.* $S' \subset \mathcal{E}_{CS}^{\mathbb{s}}$ *and* S' *is a complete support for* $\mathbb{s}$*. (minimality w.r.t. set inclusion).*

Given a complete support $\mathcal{E}_{CS}^{\mathbb{s}}$:

- $\mathcal{E}_{CS}^{\mathbb{s}}$ is called "**IN_{str} complete support**" iff $\forall e \in \mathcal{E}_{CS}^{\mathbb{s}}$, $St(e) = \mathrm{IN}_{str}$.
- $\mathcal{E}_{CS}^{\mathbb{s}}$ is called "**IN_{def} complete support**" iff it is not a IN_{str} complete support, and $\forall e \in \mathcal{E}_{CS}^{\mathbb{s}}$, $St(e) \in \{\mathrm{IN}_{str}, \mathrm{IN}_{def}\}$.
- $\mathcal{E}_{CS}^{\mathbb{s}}$ is called "**AMBIG complete support**" iff it is not a IN_{str} or IN_{def} complete support, and $\forall e \in \mathcal{E}_{CS}^{\mathbb{s}}$, $St(e) \in \{\mathrm{IN}_{str}, \mathrm{IN}_{def}, \mathrm{AMBIG}\}$.

We say that an edge $\mathbf{e}$ is superior to another edge $\mathbf{e'}$ and that $\mathbf{e'}$ is inferior to $\mathbf{e}$ iff $Rule(Source(\mathbf{e})) \succ Rule(Source(\mathbf{e'}))$, and we say that a support edge e_{sup} **defends against** an attack edge e_{att} iff e_{sup} is supporting the literal attacked by e_{att} and:

(1) Either e_{sup} is labeled IN_{str}.

(2) Or e_{sup} is labeled IN_{def} and e_{sup} is superior to e_{att} (i.e. $Rule(Source(e_{sup})) \succ Rule(Source(e_{att}))$).

Let us conclude this section by explaining how the SG is built from a propositional knowledge base. The nodes correspond to each of the rules in the knowledge base. The edges are constructed in a bottom up manner starting from the fact rules. The next section presents reasoning and labeling functions and how cycles are prevented.

4 STATEMENT GRAPH REASONING

Statement Graphs are flexible enough to represent all variants of defeasible reasoning depicted in Table 1. This flexibility is due to the labeling function that evaluates all supports and attacks for a specific rule application step. In the next section we first explain how SGs capture basic defeasible reasoning with ambiguity blocking, team defeat, and without cycles.

4.1 Labeling for Ambiguity Blocking

In SGs ambiguity blocking means that all ambiguous attack edges can be discarded and not taken into account. Team defeat means that a statement survives as long as the edges attacking it are defeated by its support edges. We use the labeling function '**BDL**' (Blocking Defeasible Logic) to obtain entailment results equivalent to Billington's defeasible logic [6] (i.e. defeasible reasoning with ambiguity blocking, team defeat and without cycles). BDL is defined as follows: edges are given the same label as their source statements (i.e. given an edge e, $\mathrm{BDL}(e) = \mathrm{BDL}(Source(e))$. Given a statement $\mathbb{s}$, *if* $\mathbb{s}$ *is the Top statement* $(\emptyset, \top)$ *then* $BDL(\mathbb{s}) = IN_{str}$. Otherwise:

(a) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{IN}_{str}$ *if* $\mathbb{s}$ *has a* IN_{str} *complete support and* $Rule(\mathbb{s})$ *is either* $\emptyset$ *or a strict rule, and* $\nexists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $BDL(\boldsymbol{e}) = IN_{str}$.

A statement is labeled IN_{str} iff it is the top statement or if it has a complete strict support (i.e. there is a strict derivation for each of its premises), an empty or strict rule, and is not strictly attacked.

(b) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{OUT}_{str}$ *iff* $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $BDL(\boldsymbol{e}) = IN_{str}$.

A statement is labeled OUT_{str} iff it is strictly attacked (i.e there is a strict derivation against its premises or rule).

(c) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{IN}_{def}$ *iff* $BDL(\mathbb{s}) \notin \{IN_{str}, OUT_{str}\}$ *and* $\mathbb{s}$ *has a* IN_{str} *or* IN_{def} *complete support* $\mathcal{E}_{CS}^{\mathbb{s}}$

1. *and* $\forall \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t* $BDL(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *undercuts* $\mathbb{s}$, $\exists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *s.t.* $\boldsymbol{e_s}$ *defends against* $\boldsymbol{e}$.
2. *and* $\forall \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $BDL(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *attacks the rule application of* $\mathbb{s}$, $Rule(\mathbb{s})$ *is either a strict rule or* $Rule(\mathbb{s}) \succ Rule(Source(\boldsymbol{e}))$.

A statement is labeled IN_{def} iff it is not strictly accepted or strictly defeated and it has a strict or defeasibly accepted complete support (i.e. there is a strict or defeasibly accepted derivation for each of its premises) and (c.1.) for all defeasibly accepted attacks it receives, it has a superior edge that defeats it (this condition allows for team defeat since a support edge does not have to defeat all attacks by itself) and (c.2.) the statement rule is either a strict rule or is superior to any defeasibly applicable rule attacking it.

(d) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{OUT}_{def}$ *iff* $BDL(\mathbb{s}) \neq OUT_{str}$ *and* $\mathbb{s}$ *has an* IN_{str} *or* IN_{def} *complete support* $\mathcal{E}_{CS}^{\mathbb{s}}$ *and*

1. *either* $\exists f \in Premise(\mathbb{s})$ *where* $\nexists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $BDL(\boldsymbol{e_s}) = IN_{str}$ *and* $\forall \boldsymbol{e'_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $BDL(\boldsymbol{e'_s}) \in \{IN_{def}, AMBIG\}$, $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t.* $BDL(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *is superior to* $\boldsymbol{e'_s}$.
2. *or* $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $BDL(\boldsymbol{e}) = IN_{def}$ *attacking the rule of* $\mathbb{s}$ *and* $Rule(\mathbb{s})$ *is not a strict rule and* $Rule(Source(\boldsymbol{e})) \succ Rule(\mathbb{s})$.

A statement is labeled OUT_{def} iff it is not strictly defeated and it has a strict or defeasibly accepted complete support and either (d.1.) one of its premises is not strictly supported and for all its defeasibly accepted or ambiguous support edges, there exists a defeasibly accepted attack edge that is superior to it (this condition allows for team defeat as an attack edge does not have to defeat all supports by itself). Or (d.2.) the statement's rule is not strict and there is a defeasibly accepted edge with a superior rule attacking it.

(e) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{AMBIG}$ *if* $BDL(\mathbb{s}) \notin \{IN_{str}, OUT_{str}, OUT_{def}\}$ *and*

1. *either* $\mathbb{s}$ *has an AMBIG complete support and no* IN_{str} *or* IN_{def} *complete support.*
2. *or* $\mathbb{s}$ *has an* IN_{str} *or* IN_{def} *complete support* $\mathcal{E}_{CS}^{\mathbb{s}}$ *and*
   1. *either* $\exists f \in Premise(\mathbb{s})$ *s.t.* $\nexists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $BDL(\boldsymbol{e_s}) = IN_{str}$ *and* $\forall \boldsymbol{e'_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $BDL(\boldsymbol{e'_s}) = IN_{def}$, $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t.* $BDL(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *is neither superior nor inferior to* $\boldsymbol{e'_s}$.
   2. *or* $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $BDL(\boldsymbol{e}) = IN_{def}$ *attacking the rule of* $\mathbb{s}$ *and* $Rule(\mathbb{s})$ *is not a strict rule and* $Rule(Source(\boldsymbol{e})) \not\succ Rule(\mathbb{s})$ *and* $Rule(\mathbb{s}) \not\succ Rule(Source(\boldsymbol{e}))$.

A statement is labeled AMBIG if it is not strictly accepted or strictly or defeasibly defeated and it either (e.1.) has an ambiguous complete support and no strict or defeasibly accepted complete support, or (e.2.) has a strict or defeasibly accepted complete support and either (e.2.1.) one of its premises is not strictly supported and for all its defeasibly accepted support edges, there exists a defeasibly accepted attack edge that is neither superior nor inferior to it, or (e.2.2) the statement's rule is not strict and is defeasibly attacked by an edge with neither a superior nor an inferior rule.

(f) $\boldsymbol{BDL}(\mathbb{s}) = \boldsymbol{UNSUP}$ *if* $BDL(\mathbb{s}) \neq OUT_{str}$ *and* $\exists f \in Premise(\mathbb{s})$ *s.t.* $\nexists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *where* $BDL(\boldsymbol{e_s}) \in \{IN_{str}, IN_{def}, AMBIG\}$.

A statement is labeled UNSUP iff it is not strictly defeated and it has a premise that is not supported by a strictly accepted, defeasibly accepted, or ambiguous edge.

EXAMPLE 4. *Consider the SG in Example 3. Applying BDL labeling function results in Figure 2. In particular, the statement* $(\{\top\}, \top \rightarrow evidA)$ *is labeled* IN_{str} *because it has a complete* IN_{str} *support and a strict rule. The statement* $(\{\overline{guilty}\}, \emptyset)$ *is labeled* IN_{def} *because it has a defeasibly accepted complete support that is not challenged by a strictly or defeasibly accepted edge.*

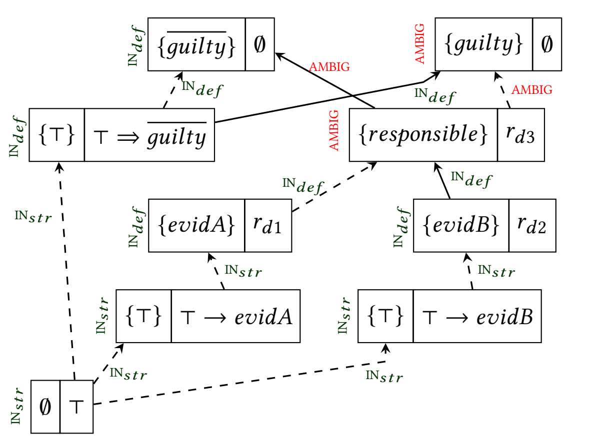

Figure 2: Applying BDL on SG of Example 3.

$\mathbb{SG}_{\mathcal{KB}}^{\mathrm{BDL}}$ denotes an SG that uses the BDL labeling function, and $\mathbb{SG}_{\mathcal{KB}}^{\mathrm{BDL}}\langle\mathbb{s}\rangle$ denotes the label of a statement $\mathbb{s}$.

LEMMA 4.1 (BDL IS A FUNCTION). [2] *All statement in a knowledge base* $\mathcal{KB}$ *have exactly one label in* $\mathbb{SG}_{\mathcal{KB}}^{BDL}$.

[2]Full proofs available at https://www.dropbox.com/s/n6k79oduzu5mzvf/proofs_sg.pdf

The equivalence between BDL and reasoning with ambiguity blocking and team defeat is shown in Proposition 1.

Proposition 1. *Let f be a literal in a defeasible $\mathcal{KB}$:*
(1) $\mathcal{KB} \vDash^{TD}_{block} f$ iff $\mathbb{SG}^{BDL}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{IN_{str}, IN_{def}\}$.
(2) $\mathcal{KB} \nvDash^{TD}_{block} f$ iff $\mathbb{SG}^{BDL}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{OUT_{str}, OUT_{def}, UNSUP\}$.

Sketch. Proof using the formalization of ambiguity blocking with team defeat shown in [2, 6]. (See Footnote 2) □

4.2 Labeling for Ambiguity Propagation

Defeasible reasoning via structured argumentation such as ASPIC+ with grounded semantics [27] yields the same entailment results as defeasible reasoning with ambiguity propagation and no team defeat [16]. The intuition behind ambiguity propagation is to reject a literal if there is an argument attacking it (whether it relies on ambiguous literals or not) and is not inferior to it.

From an SG point of view, ambiguity propagating means that ambiguous attack edges are considered valid attacks that make the statement ambiguous if it cannot defend against them.

We use the labeling function '**PDL**' (Propagating Defeasible Logic) to obtain entailment results equivalent to defeasible reasoning with ambiguity propagating, team defeat and without cycles [2]. PDL is defined the same as BDL except for the definition of IN_{def} and AMBIG labels. Edges are given the same label as their source statements (i.e. given an edge e, PDL(e) = PDL($Source(e)$) and given a statement $\mathbb{s}$:

(a) ***PDL***($\mathbb{s}$) = $\boldsymbol{IN_{str}}$ *iff BDL*($\mathbb{s}$) = IN_{str}.
(b) ***PDL***($\mathbb{s}$) = $\boldsymbol{OUT_{str}}$ *iff BDL*($\mathbb{s}$) = OUT_{str}.
(d) ***PDL***($\mathbb{s}$) = $\boldsymbol{OUT_{def}}$ *iff BDL*($\mathbb{s}$) = OUT_{def}.
(f) ***PDL***($\mathbb{s}$) = ***UNSUP*** *iff BDL*($\mathbb{s}$) = *UNSUP*.

The only difference between ambiguity blocking (BDL) and ambiguity propagating (PDL) is that in the latter ambiguous attacks are taken into account and can make the statement ambiguous. This change only affects the labeling of IN_{def} and AMBIG.

(c) ***PDL***($\mathbb{s}$) = $\boldsymbol{IN_{def}}$ *iff PDL*($\mathbb{s}$) $\notin \{IN_{str}, OUT_{str}\}$ *and* $\mathbb{s}$ *has a* IN_{str} *or* IN_{def} *complete support* $\mathcal{E}^{\mathbb{s}}_{CS}$
   1. *and* $\forall \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *s.t PDL*($\boldsymbol{e}$) $\in \{IN_{def}, AMBIG\}$ *and* $\boldsymbol{e}$ *undercuts* $\mathbb{s}$, $\exists \boldsymbol{e_s} \in \mathcal{E}^-_S(\mathbb{s})$ *s.t.* $\boldsymbol{e_s}$ *defends against* $\boldsymbol{e}$.
   2. *and* $\forall \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *s.t. BDL*($\boldsymbol{e}$) $=\in \{IN_{def}, AMBIG\}$ *and* $\boldsymbol{e}$ *attacks the rule application of* $\mathbb{s}$, *Rule*($\mathbb{s}$) *is either a strict rule or Rule*($\mathbb{s}$) $\succ$ *Rule*($Source(\boldsymbol{e})$).

In BDL, a IN_{def} statement has to defend against defeasibly accepted attacks; in PDL it also has to defend against ambiguous ones.

(e) ***PDL***($\mathbb{s}$) = ***AMBIG*** *iff PDL*($\mathbb{s}$)$\notin\{IN_{str}, OUT_{str}, OUT_{def}\}$ *and*
   1. *either* $\mathbb{s}$ *has an AMBIG complete support and no* IN_{str} *or* IN_{def} *complete support.*
   2. *or* $\mathbb{s}$ *has a* IN_{str} *or* IN_{def} *complete support* $\mathcal{E}^{\mathbb{s}}_{CS}$ *and*
      1. *either* $\exists f \in Premise(\mathbb{s})$ *s.t.* $\nexists \boldsymbol{e_s} \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t. PDL*($\boldsymbol{e_s}$) $= IN_{str}$ *and* $\forall \boldsymbol{e'_s} \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t. PDL*($\boldsymbol{e'_s}$) $= IN_{def}$, $\exists \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t. either (PDL*($\boldsymbol{e}$) $= IN_{def}$ *and* $\boldsymbol{e}$ *is neither superior nor inferior to* $\boldsymbol{e'_s}$) *or (PDL*($\boldsymbol{e}$) *= AMBIG and* $\boldsymbol{e}$ *is not inferior to* $\boldsymbol{e'_s}$).
      2. *or Rule*($\mathbb{s}$) *is not a strict rule and* $\exists \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *s.t. either (PDL*($\boldsymbol{e}$) $= IN_{def}$ *attacking the rule of* $\mathbb{s}$ *and Rule*($Source(\boldsymbol{e})$) $\nsucc$ *Rule*($\mathbb{s}$) *and Rule*($\mathbb{s}$) $\nsucc$ *Rule*($Source(\boldsymbol{e})$)) *or (PDL*($\boldsymbol{e}$) *= AMBIG attacking the rule of* $\mathbb{s}$ *and Rule*($\mathbb{s}$) $\nsucc$ *Rule*($Source(\boldsymbol{e})$)).

In PDL, a statement is also labeled AMBIG if it is attacked on its premises or rule by an ambiguous edge that is not inferior.

Example 5. *Consider the SG in Example 3. Applying PDL labeling function results in Figure 3. In particular, the statement* $(\{\overline{guilty}\},\emptyset)$ *is labeled AMBIG because it has a defeasibly accepted support edge that is not superior to the ambiguous attack edge.*

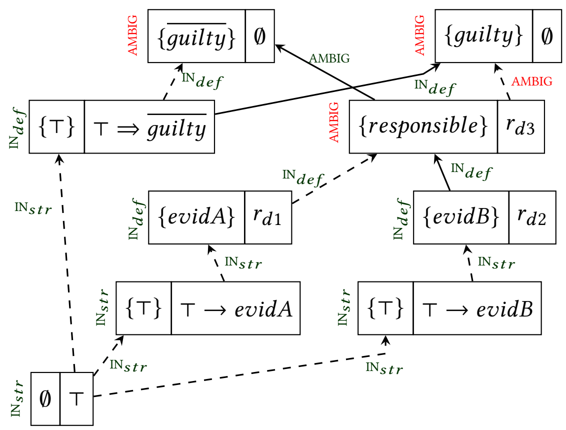

Figure 3: Applying PDL on SG of Example 3.

The equivalence between PDL and reasoning with ambiguity propagating and team defeat is shown in Proposition 2.

Proposition 2. *Let f be a literal in a defeasible $\mathcal{KB}$:*
(1) $\mathcal{KB} \vDash^{TD}_{prop} f$ iff $\mathbb{SG}^{PDL}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{IN_{str}, IN_{def}\}$.
(2) $\mathcal{KB} \nvDash^{TD}_{prop} f$ iff $\mathbb{SG}^{PDL}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{OUT_{str}, OUT_{def}, UNSUP\}$.

4.3 Labeling without Team Defeat

An inherent characteristic of defeasible reasoning is its systematic reliance on a set of intuitions and rules of thumb, which have been longly debated between logicians [1, 19, 22, 26]. Team defeat (also called direct reinstatement) is not an exception. Some defeasible reasoning techniques -such as argumentation-based ones- do not allow for team defeat, they consider that for an argument to be accepted, it has to defend itself, alone, against all its direct surviving attacks.

From SG's perspective, forbidding team defeat means that a support edge has to defend itself from all attacks. We denote the labeling function for ambiguity blocking without team defeat by BDL_{noTD} which is almost the same as BDL except that rather than considering all support edges, we only consider those in the complete support. For IN_{def}, condition (c.1) is changed to:

(c).1 *and* $\forall \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *s.t* $BDL_{noTD}(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *undercuts* $\mathbb{s}$, $\exists \boldsymbol{e_s} \in \mathcal{E}^{\mathbb{s}}_{CS}$ *s.t.* $\boldsymbol{e_s}$ *defends against* $\boldsymbol{e}$.

For OUT_{def}, an attack has to be superior to all supports:

(d).1 *either* $\exists f \in Premise(\mathbb{s})$ *where* $\nexists \boldsymbol{e_s} \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t.* $BDL_{noTD}(\boldsymbol{e_s}) = IN_{str}$ *and* $\exists \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t.* $BDL_{noTD}(\boldsymbol{e})$

$= IN_{def}$ *and* $\forall \boldsymbol{e}'_s \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t.* $BDL_{noTD}(\boldsymbol{e}'_s) \in \{IN_{def},$ *AMBIG*$\}$, $\boldsymbol{e}$ *is superior to* $\boldsymbol{e}'_s$.

For AMBIG, an attack has to be neither superior nor inferior to all supports:

(e).1.1 *either* $\exists f \in Premise(\mathbb{s})$ *s.t.* $\nexists \boldsymbol{e_s} \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t.* $BDL_{noTD}(\boldsymbol{e_s}) = IN_{str}$ *and* $\exists \boldsymbol{e} \in \mathcal{E}^-_A(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t.* $BDL_{noTD}(\boldsymbol{e}) = IN_{def}$ *and* $\forall \boldsymbol{e}'_s \in \mathcal{E}^-_S(\mathbb{s})$ *for* f *s.t.* $BDL_{noTD}(\boldsymbol{e}'_s) = IN_{def}$, *and* $\boldsymbol{e}$ *is neither superior nor inferior to* $\boldsymbol{e}'_s$.

As for ambiguity propagating without team defeat (denoted by PDL_{noTD}) the same changes are done accordingly. The entailment equivalence is described in Proposition 3.

Proposition 3. *Let* f *be a literal in a defeasible* $\mathcal{KB}$*:*

(1) $\mathcal{KB} \models^{noTD}_{block} f$ *iff* $\mathbb{SG}^{BDL_{noTD}}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{IN_{str}, IN_{def}\}$.
(2) $\mathcal{KB} \not\models^{noTD}_{block} f$ *iff* $\mathbb{SG}^{BDL_{noTD}}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{OUT_{str}, OUT_{def},$ *UNSUP*$\}$.
(3) $\mathcal{KB} \models^{noTD}_{prop} f$ *iff* $\mathbb{SG}^{PDL_{noTD}}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{IN_{str}, IN_{def}\}$.
(4) $\mathcal{KB} \not\models^{noTD}_{prop} f$ *iff* $\mathbb{SG}^{PDL_{noTD}}_{\mathcal{KB}}\langle(\{f\},\emptyset)\rangle \in \{OUT_{str}, OUT_{def},$ *UNSUP*$\}$.

4.4 Circular Reasoning and Attack Cycles

The first formalisms of defeasible reasoning [2, 6, 25] did not take cycles into account and would loop infinitely and fail to draw reasonable conclusions in some cases [21]. There are two types of cycles, *circular reasoning* (a.k.a. positive loops [7]) where cycles are due to rule applications (in SGs the cycle would only contain support edges), and cyclic attacks (a.k.a. negative loops [7]) where the cycles are due to conflicting rules (these cycles contain attack and possibly support edges). *Failure-by-looping* is a mechanism to avoid drawing unreasonable conclusions in presence of cycles [21].

Circular reasoning cycle is a sequence of unlabeled edges $\langle e_0, \ldots, e_n\rangle$ where $e_i \in \mathcal{E}_S$ and $Source(e_0) = Target(e_i)$. If all statements in the cycle cannot be labeled by taking into account other edges outside this cycle then these statements are labeled UNSUP, as described in the following Example 6. Formally, (for all labeling functions, not only BDL):

(g) ***BDL***$(\mathbb{s})$ = ***UNSUP*** *if* $BDL(\mathbb{s}) \notin \{IN_{str}, OUT_{str}, IN_{def}, OUT_{def},$ *AMBIG*$\}$ *and* $\mathbb{s}$ *is part of a support cycle.*

Example 6. *Consider the following* $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, \emptyset)$ *(and* $\mathbb{SG}^{BDL}_{\mathcal{KB}}$ *in Figure 4) representing the knowledge that a defendant is responsible iff he is guilty, he is presumed not guilty unless responsibility is proven, and there is no proof for or against his responsibility.*

- $\mathcal{F} = \{\top \Rightarrow \overline{guilty}\}$
- $\mathcal{R} = \{r_{d1} : responsible \Rightarrow guilty, r_{d2} : guilty \Rightarrow responsible\}$.

The query 'is the defendant not guilty?' cannot be answered without failure-by-looping. The defendant is not guilty (i.e. $\mathbb{SG}^{BDL}_{\mathcal{KB}}\langle(\overline{guilty},\emptyset)\rangle = IN_{def}$ *therefore* $\mathcal{KB} \models_{block} \overline{guilty}$*).*

Attack cycle is a sequence of unlabeled edges $\langle e_0, \ldots, e_n\rangle$ where $e_i \in \mathcal{E}_A \cup \mathcal{E}_S$ and $Source(e_0) = Target(e_i)$. If all statements in the cycle cannot be labeled using edges outside this cycle then these statements are labeled AMBIG, as described in Example 7. Formally:

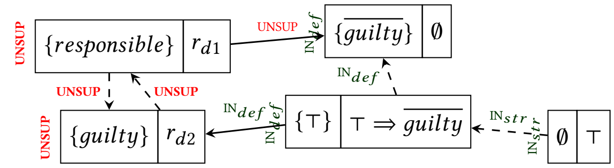

Figure 4: $\mathbb{SG}^{\mathbf{BDL}}_{\mathcal{KB}}$ of Example 6.

(h) ***BDL***$(\mathbb{s})$ = ***AMBIG*** *if* $BDL(\mathbb{s}) \notin \{IN_{str}, OUT_{str}, IN_{def}, OUT_{def},$ *AMBIG*$\}$ *and* $\mathbb{s}$ *is not part of a support cycle and is part of an attack cycle.*

Example 7. *Consider the following* $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, \emptyset)$ *(and* $\mathbb{SG}^{PDL}_{\mathcal{KB}}$ *in Figure 5) that represents a process of deciding if the* ***Platypus*** *is a reptile. The rules are (all defeasible): If it lays eggs and does not have wings then it is a reptile. If it is a reptile then it does not have fur. If it has fur then it is a mammal. If it is a mammal then it does not lay eggs. The Platypus lays eggs, has fur, and does not have wings.*

- $\mathcal{F} = \{\top \Rightarrow layEggs, \top \Rightarrow \overline{wings}, \top \Rightarrow fur\}$
- $\mathcal{R} = \{r_{d1} : layEggs \wedge \overline{wings} \Rightarrow reptile, r_{d2} : reptile \Rightarrow \overline{fur}, r_{d3} : fur \Rightarrow mammal, r_{d4} : mammal \Rightarrow \overline{layEggs}\}$.

The answer to the query ***'is the Platypus a reptile?' is 'false'*** *(i.e.* $\mathbb{SG}^{PDL}_{\mathcal{KB}}\langle(reptile,\emptyset)\rangle = AMBIG$ *therefore* $\mathcal{KB} \not\models_{prop} reptile$*).*

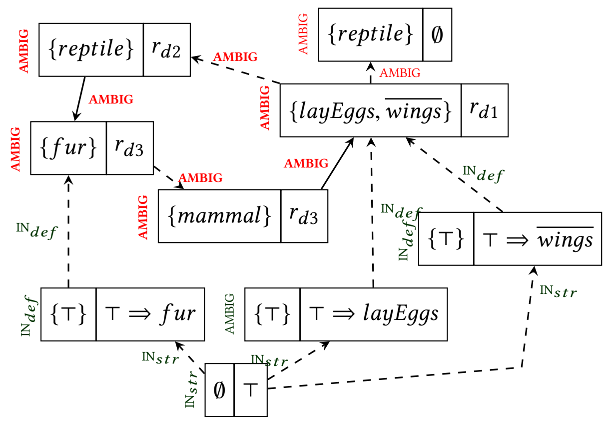

Figure 5: $\mathbb{SG}^{\mathbf{PDL}}_{\mathcal{KB}}$ of Example 7.

Circular reasoning is avoided by argumentation approaches as all constructed arguments start from $\top$ while attack cycles are handled inherently by grounded semantics and dialectical trees. Propositions 1, 2, and 3 still hold with *failure-by-looping*.

5 EVALUATION

5.1 Defeasible Reasoning Tool

While defeasible reasoning has been applied in various domains, available tools lack many functionalities. For example there are no first order tool that provide ambiguity propagating with team defeat (cf. Table 1). In order to expand the usability and appeal of defeasible reasoning, we propose an implementation of Statement Graph called **ELDR** (Existential Logic for Defeasible Reasoning) that provides defeasible reasoning with first order existential rules[3],

[3]The minimal superset of the languages used in first order logic defeasible reasoning.

ambiguity blocking or propagating, with or without team defeat, and with failure-by-looping.

Existential rules ($\exists$-rules) [10] are built with ($\exists, \forall$) quantifiers, the connectors ($\rightarrow, \Rightarrow, \rightsquigarrow$) and conjunction ($\wedge$). An *atom* is of the form $p(t_1, \cdots, t_k)$ and its complement is of the form $\overline{p(t_1, \cdots, t_k)}$, where p is a predicate and t_i are variables (denoted by uppercase) or constants (denoted by lowercase or nulls). A *rule r* is a formula of the form $\forall \vec{X}, \vec{Y}\ \left(\mathcal{B}(\vec{X}, \vec{Y}) \Rrightarrow \exists \vec{Z}\ \mathcal{H}(\vec{X}, \vec{Z})\right)$ such that $\Rrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, where $\vec{X}, \vec{Y}$ are tuples of variables, $\vec{Z}$ is a tuple of *existential variables*, and $\mathcal{B}, \mathcal{H}$ are finite non-empty conjunctions of atoms. To generate the statements we restrict rules to the FES (Finite Expansion Set) fragment [4] which are guaranteed to stop in forward chaining. We use skolemisation in order to ground the rule applications with existentials. A statement is composed of ground atoms (atoms without variables) and a rule as shown in Example 8.

Example 8. *The following $\mathcal{KB} = (\mathcal{F}, \mathcal{R}, \emptyset)$ (and $\mathbb{SG}^{BDL}_{\mathcal{KB}}$ in Figure 6) of an* ***animal shelter*** *describes the process of deciding if a found animal is a stray or not. An animal is assumed to be a stray unless proven otherwise. Generally, if an animal has a collar then it has an owner and if it has an owner then it is not a stray. An animal called 'dogo' with a collar is found alone, is it a stray?*

- $\mathcal{F} = \{\top \Rightarrow stray(dogo), \top \rightarrow hasCollar(dogo)\}$
- $\mathcal{R} = \{r_{d1} : \forall X\ hasCollar(X) \Rightarrow \exists Y\ hasOwner(X, Y),$ $r_{d2} : \forall X\ hasOwner(X, Y) \Rightarrow \overline{stray(X)}\}$.

*The answer to 'is 'dogo' a stray is **'false'** (i.e. $\mathcal{KB} \nvDash_{block}\ stray(dogo)$ since $\mathbb{SG}^{BDL}_{\mathcal{KB}}\langle(stray(dogo), \emptyset)\rangle$ = AMBIG).*

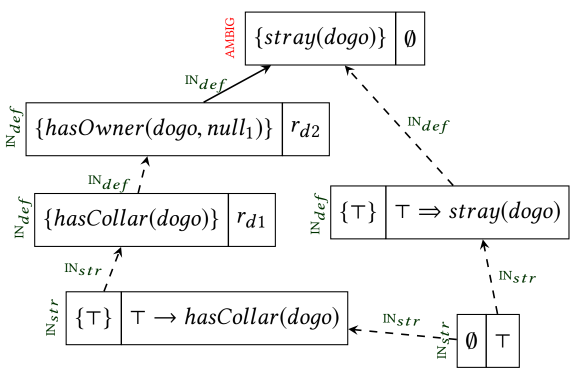

Figure 6: $\mathbb{SG}^{\mathbf{BDL}}_{\mathcal{KB}}$ of Example 8.

Tools and Performance. The main reasoning tools considered are ASPIC+ [27], DEFT [18], DeLP [14] and Flora-2 [31]. Table 1 (cf. Section 1) shows which variants of defeasible reasoning these tools support compared to ELDR that handles them all. Except for DEFT none supports FES rules.

Handling more variants of defeasible reasoning is a desirable feature as long as performance does not significantly suffer. We conducted an empirical evaluation of ELDR in order to measure its performance w.r.t. the available implementations of first order defeasible reasoning tools[4]. The experiments are built upon a pre-established defeasible reasoning benchmark proposed in [20]. The benchmark we consider is composed of 5 parameterized knowledge bases (also known as theories): **Chain Theory** tests performance when faced with a simple chain of rules; **Circle Theory** tests infinite loops (cycles); **Trees Theories** test a large number of arguments with small derivations; **Levels Theory** tests performance for ambiguity blocking or propagating; and **Teams Theory** tests performance w.r.t. a sizeable number of conflicts when team defeat is allowed or not. Table 2 presents the time (in CPU seconds) for each tool to answer the query of each theory. ∞ denotes a stack overflow, $T.O.$ denotes a timeout (set to 300 seconds).

Table 2: Execution time in seconds (selected results)

Theory		Propagating without Team Defeat				Blocking with TD	
		ELDR	ASPIC+	DEFT	DeLP	ELDR	Flora-2
$chain(n)$	$n = 500$	0.39	58.52	**0.33**	64.26	**0.39**	2.33
	$n = 2000$	1.93	∞	**1.86**	$T.O.$	**1.92**	7.77
$circle(n)$	$n = 500$	1.15	∞	**0.31**	99.85	**1.14**	2.42
	$n = 2000$	3.92	∞	**1.88**	$T.O.$	**3.93**	7.84
$levels(n)$	$n = 500$	2.16	**0.83**	3.65	$T.O.$	**2.41**	53.94
	$n = 2000$	72.09	**51.23**	64.94	$T.O.$	**75.14**	$T.O.$
$trees(n)$	$n = 2$	0.06	0.04	**0.01**	253.38	**0.06**	0.81
	$n = 5$	**0.07**	∞	1.32	$T.O.$	**0.07**	5.07
$teams(n)$	$n = 2$	0.93	0.54	**0.31**	27.87	0.66	**0.59**
	$n = 5$	54.22	120.73	**30.04**	$T.O.$	**2.36**	3.05

Tools can only be compared on situations where they compute the same results, we used PDL_{noTD} to compare against ASPIC+, DEFT, and DeLP, and BDL to compare against Flora-2. Results indicate that ELDR outperforms existing tools on certain tests and gives the same results as the best tools on all other tests. These results are due to various factors: first, some tools were made as proofs of concept with no performance in mind, second, the used inference mechanism (resolution vs forward chaining), third, the number of generated arguments (e.g. if there is a rule application used in different arguments, SG evaluates only one statement for it, however other formalisms might evaluate it for each argument).

5.2 Labeling for Human Reasoning

We conclude the paper by a discussion on how the flexibility of SGs can allow to capture human reasoning. We focus in this paper on the suppression task [9], a psychological study showing that people tend to change (suppress) previously drawn conclusions when additional information becomes available. The "**modus-ponens suppression task**" is explained in Example 9.

Example 9. *Consider the following situation 1 [9]:*

(1) "If Lisa has an essay to write, she will study late in the library".
(2) "Lisa has an essay to write".
- Will Lisa study late in the library?

Most subjects (96%) conclude that she will study late in the library. However, if subjects receive an additional information (situation 2):

(3) "If the library stays open, she will study late in the library".

Only a minority (38%) concludes she will study late in the library.

This study shows that, much like non-monotonic reasoning, conclusions can be suppressed in human reasoning in presence of additional information. To represent the situation in a logical form, a background knowledge rule stronger than (1) is added: *"if the*

[4]For ASPIC+ we used an implementation provided by the authors of the tool. For DeLP we used the implementation in Tweety1.7 libraries. For DEFT and Flora-2 we used their open source version.

library does not stay open then Lisa will not study late in the library" as shown in Example 10.

Example 10. *Consider a representation of Example 9 s.t.:*

- *"essay" denotes "She has an essay to write".*
- *"library" denotes "She will study late in the library".*
- *"open" denotes "Library stays open".*

Let $\mathcal{KB}_1 = (\mathcal{F}, \mathcal{R}_1, \emptyset)$ *be the representation of situation 1:*

- $\mathcal{R}_1 = \{r_{d1} : essay \Rightarrow library\}$. $\mathcal{F} = \{\top \Rightarrow essay\}$.

Let $\mathcal{KB}_2 = (\mathcal{F}, \mathcal{R}_2, \succ)$ *be the representation of situation 2:*

- $\mathcal{R}_2 = \mathcal{R}_1 \cup \{r_{d2} : open \Rightarrow library, r_{d3} : \overline{open} \Rightarrow \overline{library}\}$.
- $r_{d3} \succ r_{d1}$.

Defeasible reasoning (in all its variants) **cannot represent the suppression task** as "*library*" is derivable and not defeasibly contested (i.e $\mathcal{KB}_1 \vDash library$ and $\mathcal{KB}_2 \vDash library$). However, three-valued logics could model human reasoning [11, 29, 30].

A possible explanation for the modus-ponens suppression effect is that humans consider as possibly valid unsupported counter-arguments (attacks), they think that the library might be closed and therefore cannot conclude that Lisa will study in the library. Let us show in the reminder of the section how such reasoning behavior could be captured by the labeling function of the SGs. More precisely, this can be represented by a labeling function (denoted SUP) that considers UNSUP attack edges valid if superior to the support edges to make the statement AMBIG. Given an edge e, $\text{SUP}(e) = \text{SUP}(Source(e))$. Given a statement $\mathbb{s}$:

(a) $\textbf{SUP}(\mathbb{s}) = \textbf{IN}_{str}$ *iff* $BDL(\mathbb{s}) = IN_{str}$.
(b) $\textbf{SUP}(\mathbb{s}) = \textbf{OUT}_{str}$ *iff* $BDL(\mathbb{s}) = OUT_{str}$.
(d) $\textbf{SUP}(\mathbb{s}) = \textbf{OUT}_{def}$ *iff* $BDL(\mathbb{s}) = OUT_{def}$.
(f) $\textbf{SUP}(\mathbb{s}) = \textbf{UNSUP}$ *iff* $BDL(\mathbb{s}) = UNSUP$.

For IN_{def} we add extend (c) with the conditions (c.3) and (c.4) that for all attack edge $\mathbf{e}$ s.t. $\text{SUP}(\mathbf{e}) = \text{UNSUP}$, if $\mathbf{e}$ attacks a premise then there must exist a support edge for that premise that is not inferior to $\mathbf{e}$, and if $\mathbf{e}$ attacks the rule, then the rule must not be inferior to $\mathbf{e}$.

(c).3 *and* $\forall \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t* $SUP(\boldsymbol{e}) = UNSUP$ *and* $\boldsymbol{e}$ *undercuts* $\mathbb{s}$ *on* f, $\exists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $\boldsymbol{e_s}$ *is not inferior to* $\boldsymbol{e}$.
(c).4 *and* $\forall \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t.* $SUP(\boldsymbol{e}) = UNSUP$ *and* $\boldsymbol{e}$ *attacks the rule application of* $\mathbb{s}$, $Rule(\mathbb{s})$ *is either a strict rule or* $Rule(Source(\boldsymbol{e})) \not\succ Rule(\mathbb{s})$.

A statement is also AMBIG if there is a UNSUP attack on the premise that is superior to the support edge, or if it attacks the rule and is superior to it.

(e).2.1 *either* $\exists f \in Premise(\mathbb{s})$ *s.t.* $\nexists \boldsymbol{e_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $SUP(\boldsymbol{e_s}) = IN_{str}$ *and* $\forall \boldsymbol{e'_s} \in \mathcal{E}_S^-(\mathbb{s})$ *for* f *s.t.* $SUP(\boldsymbol{e'_s}) = IN_{def}$, $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *attacking* $\mathbb{s}$ *on* f *s.t. either* ($SUP(\boldsymbol{e}) = IN_{def}$ *and* $\boldsymbol{e}$ *is neither superior nor inferior to* $\boldsymbol{e'_s}$) *or* ($SUP(\boldsymbol{e}) = UNSUP$ *and* $\boldsymbol{e}$ *is superior to* $\boldsymbol{e'_s}$).
(e).2.2 *or* $Rule(\mathbb{s})$ *is not a strict rule and* $\exists \boldsymbol{e} \in \mathcal{E}_A^-(\mathbb{s})$ *s.t. either* ($SUP(\boldsymbol{e}) = IN_{def}$ *attacking the rule of* $\mathbb{s}$ *and* ($Rule(Source(\boldsymbol{e})) \not\succ Rule(\mathbb{s})$ *and* $Rule(\mathbb{s}) \not\succ Rule(Source(\boldsymbol{e}))$) *or* ($SUP(\boldsymbol{e}) = UNSUP$ *attacking the rule of* $\mathbb{s}$ *and* $Rule(Source(\boldsymbol{e})) \not\succ Rule(\mathbb{s})$).

The SUP labeling function gives $\mathbb{SG}^{\text{SUP}}_{\mathcal{KB}_1}\langle(library, \emptyset)\rangle = \text{IN}_{def}$ and $\mathbb{SG}^{\text{SUP}}_{\mathcal{KB}_2}\langle(library, \emptyset)\rangle = \text{AMBIG}$ (as shown in Figures 7 and 8) which correctly models the modus ponens suppression effect.

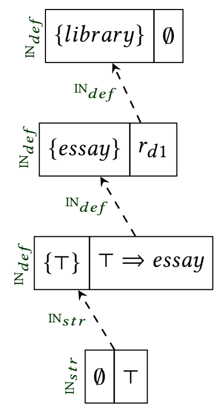

Figure 7: $\mathbb{SG}^{\text{SUP}}_{\mathcal{KB}_1}$ of Example 10.

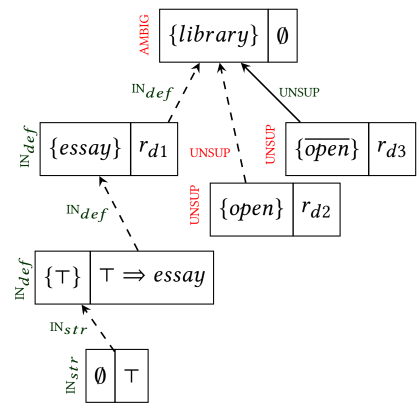

Figure 8: $\mathbb{SG}^{\text{SUP}}_{\mathcal{KB}_2}$ of Example 10.

Please note that we defined SUP using BDL but there is no proof that human use ambiguity blocking with team defeat. However we make this assumption for simplicity. A labeling function based on the same intuition as SUP but using ambiguity propagating (with or without team defeat) would also effectively model the modus-ponens suppression effect. Empirical data and more testing are needed to justify one or the other.

Last, let us note that while the suppression effect can occur either as a consequence of a suitable reasoning mechanism or due to specific logical representation of the situation [29], we made sure to use the "plain" representation where only the background knowledge is added [29]. Other representations can be used such as the "necessary condition" by using the rule "$essay \land open \Rightarrow library$" or the weak completion and adding an abnormality predicate [11].

6 DISCUSSION

In this paper we introduced a new argumentation-based formalism called Statement Graph that represents rule applications as "statements" with attack and support relations (i.e. edges) between them. By applying a flexible labeling function on edges, different variants of defeasible reasoning can be obtained (ambiguity propagating or blocking with or without team defeat). We evaluated our work by proposing the ELDR tool that implements SGs and not only covers gaps not addressed by the existing tools but it also has the same and sometimes better performance results. In future work we plan on studying if SGs can be used to represent other non-monotonic reasoning such as the selection task and other classes of defeasible reasoning logics ([7, 21]).

ACKNOWLEDGEMENTS

The authors acknowledge the support of the H2020 CORDIS NoAW project (project ID 688338).

REFERENCES

[1] Grigoris Antoniou. 2006. Defeasible reasoning: A discussion of some intuitions. *International Journal of Intelligent Systems* 21, 6 (2006), 545–558. https://doi.org/10.1002/int.20147

[2] Grigoris Antoniou, David Billington, Guido Governatori, Michael J Maher, and Andrew Rock. 2000. A Family of Defeasible Reasoning Logics and its Implementation. In *Proceedings of the 14th European Conference on Artificial Intelligence*. 459—-463.

[3] Grigoris Antoniou, David Billington, and Michael J Maher. 1999. On the analysis of regulations using defeasible rules. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*. IEEE, 7—-pp.

[4] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. 2011. On rules with existential variables: Walking the decidability line. *Artificial Intelligence* 175, 9-10 (2011), 1620–1654.

[5] Nick Bassiliades, Grigoris Antoniou, and Ioannis Vlahavas. 2006. A defeasible logic reasoner for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)* 2, 1 (2006), 1–41.

[6] David Billington. 1993. Defeasible Logic is Stable. *Journal of logic and computation* 3, 4 (1993), 379–400.

[7] David Billington. 2008. Propositional clausal defeasible logic. *Logics in Artificial Intelligence* (2008), 34–47.

[8] Gerhard Brewka and Stefan Woltran. 2010. Abstract dialectical frameworks. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*.

[9] Ruth M J Byrne. 1989. Suppressing valid inferences with conditionals. *Cognition* 31, 1 (1989), 61–83.

[10] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2012. A general datalog-based framework for tractable query answering over ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web* 14 (2012), 57–83.

[11] Emmanuelle-Anna Dietz, Steffen Hölldobler, and Christoph Wernhard. 2014. Modeling the suppression task under weak completion and well-founded semantics. *Journal of Applied Non-Classical Logics* 24, 1-2 (2014), 61–85.

[12] Marlon Dumas, Guido Governatori, Arthur H M Hofstede, and Phillipa Oaks. 2002. A Formal Approach to Negotiating Agents. *Electronic commerce research and applications* 1, 2 (2002), 193–207.

[13] Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77, 2 (1995), 321–357.

[14] Alejandro J García and Guillermo R Simari. 2004. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming* 4, 1+ 2 (2004), 95–138.

[15] Diego R Garcia, Alejandro J Garcia, and Guillermo R Simari. 2007. Planning and defeasible reasoning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 856—-858.

[16] Guido Governatori, Micheal J Maher, Grigoris Antoniou, and David Billington. 2004. Argumentation Semantics for Defeasible Logic. *Journal of Logic and Computation* 14, 5 (2004), 675–702. https://doi.org/10.1093/logcom/14.5.675.1

[17] Benjamin N Grosof, Yannis Labrou, and Hoi Y Chan. 1999. A declarative approach to business rules in contracts: courteous logic programs in XML. In *Proceedings of the 1st ACM conference on Electronic commerce*. ACM, 68–77.

[18] Abdelraouf Hecham, Madalina Croitoru, and Pierre Bisquert. 2017. Argumentation-Based Defeasible Reasoning For Existential Rules. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. 1568—-1569.

[19] John F Horty, D S Touretzky, and R H Thomason. 1987. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. 476–482.

[20] Michael J Maher, Andrew Rock, Grigoris Antoniou, David Billington, and Tristan Miller. 2001. Efficient defeasible reasoning systems. *International Journal on Artificial Intelligence Tools* 10, 04 (2001), 483–501.

[21] Frederick Maier and Donald Nute. 2010. Well-founded semantics for defeasible logic. November 2008 (2010), 243–274. https://doi.org/10.1007/s11229-009-9492-1

[22] David Makinson and Karl Schlechta. 1991. Floating conclusions and zombie paths: two deep difficulties in the “directly skeptical” approach to defeasible inheritance nets. *Artificial intelligence* 48, 2 (1991), 199–209.

[23] Maria Vanina Martinez, Ariel Cristhian David Deagustini, Marcelo A Falappa, and Guillermo Ricardo Simari. 2014. Inconsistency-Tolerant Reasoning in Datalog +- Ontologies via an Argumentative Semantics. In *IBERAMIA*, Vol. 14. 15–27. https://doi.org/10.1007/978-3-319-12027-0

[24] Leora Morgenstern. 1998. Artificial Intelligence Inheritance comes of age : applying nonmonotonic techniques to problems in industry. *Artificial Intelligence* 103 (1998), 237–271.

[25] Donald Nute. 1988. *Defeasible reasoning: a philosophical analysis in prolog*. Springer.

[26] Henry Prakken. 2002. Intuitions and the modelling of defeasible reasoning: some case studies. In *Ninth Int Workshop on Nonmonotonic Reasoning*. Toulouse, 91–99. arXiv:cs/0207031

[27] Henry Prakken. 2010. An abstract framework for argumentation with structured arguments. *Argument and Computation* 1, 2 (2010), 93–124.

[28] Henry Prakken and Giovanni Sartor. 1997. Argument-based extended logic programming with defeasible priorities. *Journal of applied non-classical logics* 7, 1-2 (1997), 25–75.

[29] Marco Ragni, Christian Eichhorn, Tanja Bock, Gabriele Kern-Isberner, and Alice Ping Ping Tse. 2017. Formal Nonmonotonic Theories and Properties of Human Defeasible Reasoning. *Minds and Machines* 27, 1 (2017), 79–117. https://doi.org/10.1007/s11023-016-9414-1

[30] Keith Stenning and Michiel Van Lambalgen. 2012. *Human reasoning and cognitive science*. MIT Press.

[31] Guizhen Yang, Michael Kifer, and Chang Zhao. 2003. Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE* (2003), 671–688.