A First Order Logic Benchmark for Defeasible Reasoning Tool Profiling

Abdelraouf Hecham, Madalina Croitoru, Pierre Bisquert

A First Order Logic Benchmark for Defeasible Reasoning Tool Profiling

Abdelraouf Hecham[1], Madalina Croitoru[1], Pierre Bisquert[2]

[1] LIRMM, University of Montpellier, France
[2] INRA, France

Abstract. In this paper we are interested in the task of a data engineer choosing what tool to use to perform defeasible reasoning with a first order logic knowledge base. To this end we propose the first benchmark in the literature that allows one to classify first order defeasible reasoning tools based on their semantics, expressiveness and performance.

1 Introduction

Used in many practical domains [21, 3, 14], defeasible reasoning allows to reason with incomplete or inconsistent knowledge where conclusions can be challenged by additional information. 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 [17, 20, 23, 1]. For example, should an information that is derived from a contested claim be used to contest another claim (i.e. ambiguity handling)? Or, can different 'chains' of reasoning for the same claim be combined to defend against challenging statements (i.e. reinstatement)? It appears that no single approach is appropriate in all situations, or for all purposes. When it comes to the defeasible reasoning tools in the literature, confusingly, each follows different subsets not clearly stated in the tool description or companion papers.

We are interested in the task of a data engineer *looking to select what tool to use* to perform defeasible reasoning. To facilitate this task we propose **the first benchmark in the literature for first order logic defeasible reasoning tools profiling**. We show how to use the proposed benchmark **in order to categorize existing tools** based on their *semantics* (e.g. ambiguity handling), logical language (e.g. existential rules) and *expressiveness* (e.g. priorities). We stress that we do *not want to compare* the tools amongst themselves but to be able to provide *an informative benchmark* to allow understanding the *strengths and intuitions of available tools*.

After introducing in Section 2 the logical language and defeasible reasoning, in Section 3 we enumerate the various criteria for analysing defeasible reasoning tools. In Section 4 we present the proposed benchmark and show how the structure of the benchmark corresponds to the various tools criteria. Finally, in Section 5, we run a set of available tools on the benchmark and discuss the results.

2 Background Notions

Language. We consider[1] a first order language Ł composed of formulas built with the usual quantifiers $(\exists, \forall)$ and the connectors: implication $(\rightarrow, \Rightarrow, \rightsquigarrow)$ and conjunction $(\wedge)$. An *atom* is of the form $p(t_1, \cdots, t_k)$ and its complement is of the form $\neg p(t_1, \cdots, t_k)$, where p is a predicate and t_i are variables or constants [2]. $\top, \bot$ are also allowed and considered themselves atoms. A *rule* r is a formula of the form $\forall X, Y \ \big(\mathscr{B}(X,Y) \Rrightarrow \exists Z \ \mathscr{H}(X,Z)\big)$ such that $\Rrightarrow \in \{\rightarrow, \Rightarrow, \rightsquigarrow\}$, where X, Y are tuples of variables, Z is a tuple of *existential variables*, and $\mathscr{B}$, $\mathscr{H}$ are finite non-empty conjunctions of atoms respectively called *body* and *head* of r and denoted $Body(r)$ and $Head(r)$. A rule r can be of three types: (1) strict rules $(\rightarrow)$ express undeniable implications i.e. if $Body(r)$ then definitely $Head(r)$; (2) defeasible rules $(\Rightarrow)$ express a weaker implication i.e. if $Body(r)$ then generally $Head(r)$ and (3) defeater rule $(\rightsquigarrow)$ are used to prevent the complement of a conclusion from being drawn i.e. if $Body(r)$ then the complement of any atom in $Head(r)$ should not be concluded. It does not imply that $Head(r)$ is concluded.

A defeasible knowledge base is a tuple $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \succ)$ where $\mathscr{F}$ is a set of 'fact' rules of the form $\top \rightarrow \mathscr{B}(a)$ or $\top \Rightarrow \mathscr{B}(a)$, $\mathscr{R}$ is a finite set of rules (strict rules are denoted by $\mathscr{R}_{\rightarrow}$, defeasible rules by $\mathscr{R}_{\Rightarrow}$, and defeater rules by $\mathscr{R}_{\rightsquigarrow}$), and $\succ$ is an acyclic superiority relation over rules. A rule r_1 is said to be *superior* to another rule r_2 iff $r_1 \succ r_2$ and $r_2 \not\succ r_1$ (r_2 is said to be inferior to r_1). This expresses that r_1 may override r_2. Since defeasible reasoning is mostly concerned by whether a conclusion is entailed or not, we only consider boolean queries. A boolean query is an existentially closed conjunction of atoms of the form $Q = \exists X \Phi(X,a)$? (where Φ is a conjunction of atoms, X is a set of existential variables, and a is a set of constants). A ground boolean query is a conjunction of ground atoms of the form $Q = \Phi(a)$?.

Inference Mechanisms. Reasoning can be achieved using inference mechanisms that fall under two main approaches. *Forward chaining* (also called *chase*) is the exhaustive application of the set of rules over the set of facts. *Backward chaining* consists in using the rules to rewrite the query. Reasoning with existential rules is not always guaranteed to stop. Decidable abstract classes of rules have been defined [6]: (1.) Finite Expansion Set (FES) where forward chaining is guaranteed to stop, (2.) Finite Unification Set (FUS) where backward chaining is guaranteed to stop, and (3.) Bounded Treewidth Set (BTS) which ensures decidability although no algorithm is currently available for this class. Greedy BTS (GBTS) is an expressive subclass of BTS that is provided with a forward-chaining-like algorithm. Concrete classes that may specialize one or several of the above abstract classes are recognizable by syntactic properties (e.g. transitive rules for FES class) [5], online tools such as KIABORA[3] can output if the knowledge base is decidable (i.e FES, FUS, GBTS or their combination). Cyclicity on rules poses additional decidability problems [5]. The Graph of Rule Dependencies (GRD) is a directed graph that encodes the interactions between rules: nodes represent rules and there is an

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

[2] Variables are denoted by uppercase letters $X, Y, Z, etc.$, constants by lowercase letters $a, b, c, etc.$, and unknown constants (nulls) by $null_1, null_2, etc.$

[3] `http://graphik-team.github.io/graal/kiabora`

edge from r_1 to r_2 iff an application of the rule r_1 may create a new application of the rule r_2. A GRD is acyclic when it has no circuit.

Reasoning. To reason defeasibly about a conclusion, all chains of rule applications reaching that conclusion must be evaluated along with any conflict that arises from other chains of rules. This can be achieved using two kinds of approaches: First, using the idea of extensions, where reasoning chains (arguments) are built then evaluated at a later stage; this encapsulates argumentation-based techniques such as grounded semantics [12] and dialectical trees [13]. Other approaches are based on the evaluation of arguments during their construction, such as defeasible logics [22, 9].

3 Defeasible Reasoning Features

Let us concretely discuss the various **features** concerning *semantics*, *expressiveness* and *performance* based on the different **intuitions** behind defeasible reasoning [17, 20, 23, 1]. Please note that we do not discuss what intuitions are better to adopt, as these often conflict. Our aim is to facilitate the task of selecting what defeasible reasoning tool to use based on the reasoning requirements and the data at hand.

Semantics. Different intuitions impact what conclusions are accepted or rejected in a defeasible reasoning setting:

1. *Ambiguity Handling:* As illustrated in Example 1 information derived from an ambiguous (i.e. contested) claim should be used to contest another claim [25].

   Example 1 *The defeasible knowledge base $\mathscr{K}$ describes that evidence "a" suggests that the defendant 'jack' is responsible of the crime while evidence "b" indicates that he is not. According to the underlying legal system, a defendant is presumed not guilty unless responsibility has been proven. $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \varnothing)$:*
   - $\mathscr{F} = \{\top \Rightarrow evid(a, incriminating, jack),\ \top \Rightarrow evid(b, absolving, jack),\ \top \Rightarrow defendant(jack)\}$.
   - $\mathscr{R} = \{\forall X, defendant(X) \Rightarrow \neg guilty(X),\ \forall X, Y, evid(X, incriminating, Y) \Rightarrow responsible(Y), \forall X, Y, evid(X, absolving, Y) \Rightarrow \neg responsible(Y),$ $\forall X, responsible(X) \Rightarrow guilty(X)\}$.

   *Is jack not guilty? Given both evidence "a" and "b", the fact "$responsible(jack)$" is ambiguous and it is used to derive "$guilty(jack)$". The question is "Should $guilty(jack)$ be used to contest $\neg guilty(jack)$ and make it ambiguous?". In an **ambiguity blocking** setting, the ambiguity of "$responsible(jack)$" blocks (forbids) any ambiguity derived from it. Therefore "$\neg(guilty(jack))$" is uncontested and the answer to the query $Q = \neg guilty(jack)$? is 'true'. On the other hand, in an **ambiguity propagating** setting, the ambiguity of "$responsible(jack)$" is propagated to "$\neg guilty(jack)$" because its complement "$guilty(jack)$" can be derived, hence, the answer to the query Q is 'false'.*

   Ambiguity propagation results in fewer conclusions (since more ambiguities are allowed), which might make it preferable when the cost of an incorrect conclusion is high, whereas *ambiguity blocking* might be more intuitive in situations where contested claims cannot be used to contest other claims (e.g. in the legal domain) [17].

2. *Team Defeat (Direct Reinstatement):* The absence of team defeat means that a rule r_1 implying an atom f attacked by another rule r_2 implying $\neg f$ can only be defended by r_1 itself (meaning that for r_1 to 'survive' it has to be superior to r_2 i.e. $r_1 \succ r_2$ and $r_2 \not\succ r_1$). However, if we allow team defeat, r_1 can be successfully defended by another rule r_3 for f that is superior to r_2 (even if r_1 is inferior to r_2), as illustrated 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.* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \succ)$*:*
   - $\mathscr{F} = \{\top \Rightarrow animal(tweety) \wedge bird(tweety) \wedge penguin(tweety) \wedge magical(tweety)\}$.
   - $\mathscr{R} = \{r_1 : \forall X, animal(X) \Rightarrow \neg fly(X), r_2 : \forall X, bird(X) \Rightarrow fly(X),$
     $r_3 : \forall X, penguin(X) \Rightarrow \neg fly(X), r_4 : \forall X, magical(X) \wedge penguin(X) \Rightarrow fly(X)\}$.
   - $(r_2 \succ r_1)$, $(r_4 \succ r_3)$.

   The query is $Q = fly(tweety)$ *(i.e. can "Tweety" fly?). In the absence of team defeat, the answer to Q is* ***'false'*** *because there is no chain of reasoning for "*$fly(tweety)$*" that can defend itself from all attacks: even if* r_2 *defends itself from* r_1 *(because* $r_2 \succ r_1$*), it does not defend against* r_3 *(since* $r_2 \not\succ r_3$*), and the same applies for* r_4*: it defends against* r_3 *but fails against* r_1 *because* $r_4 \not\succ r_1$*. If team defeat is allowed then the answer to Q is* ***'true'*** *("Tweety" can fly) because all attacks are defeated:* r_1 *is defeated by* r_2 $(r_2 \succ r_1)$ *and* r_3 *is defeated by* r_4 $(r_4 \succ r_3)$*.*

3. *Floating Conclusions:* Sometimes two conflicting and equally strong arguments might lead to the same conclusion down the line. These conclusions are called 'floating conclusions' [20].

   Example 3 *A first witness says that "Jack" killed "John" by stabbing him while a second witness says that he shot him. Both testimonies are of equal strength and both imply that "Jack" killed "John", however they are conflicting.* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$*:*
   - $\mathscr{F} = \{\top \Rightarrow stabbed(jack, john) \wedge shot(jack, john)$.
   - $\mathscr{R} = \{r_1 : \forall X, Y, stabbed(X,Y) \Rightarrow \neg shot(X,Y),$
     $r_2 : \forall X, Y\, shot(X,Y) \Rightarrow \neg stabbed(X,Y), r_3 : \forall X, Y\, stabbed(X,Y) \Rightarrow killed(X,Y),$
     $r_4 : \forall X, Y\, shot(X,Y) \Rightarrow killed(X,Y)\}$.

   The query is $Q = killed(jack, john)$ *(i.e. did "Jack" kill "John"?). "*$killed(jack, john)$*" is a floating conclusion. One might argue that regardless if the witnesses disagree on the details, the conclusion is the same and therefore the answer to Q is* ***'true'*** *(i.e. floating conclusions should be accepted). However, one can also argue that the two witnesses undermine each other's credibility, and therefore the answer to the query Q should be* ***false*** *(i.e. floating conclusions should be rejected).*

4. *Handling of Strict Rules:* In some defeasible reasoning tools, such as the ones based on Defeasible Logics, facts that are not in direct conflict, and that are defeasibly derived from non ambiguous ones, are accepted (even if the application of strict rules and facts generates conflict). Other formalisms reject any fact that leads to conflict when strict rules and facts are applied.

   Example 4 (Consistent Answers) *Consider the following* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$*:*

– $\mathscr{F} = \{\top \Rightarrow incrim(e1, alice), \top \Rightarrow absolv(e2, alice)\}$
– $\mathscr{R} = \{r_1 : \forall X, Y\ incrim(X,Y) \rightarrow resp(Y),\ r_2 : \forall X, Y\ absolv(X,Y) \rightarrow \neg resp(X)\}$
The facts $incrim(e1, alice)$ and $absolv(e2, alice)$ are entailed in some formalisms because there is no direct attack on them, however other formalisms consider that these facts are not entailed because they lead to a conflict if strict rules are applied.

Expressiveness. Reasoning tools can be classified w.r.t. the expressiveness of their underlying language:

1. *Rules with Existential Variables:* They are logical fragments useful in applications such as Ontology Based Data Access (OBDA) [18] and Semantic Web [11]. Detecting support for existential rules is tricky since most defeasible reasoning tools omit *quantifiers* which might lead to unwanted results. Variables appearing in the head of rules are sometimes considered existential variables, for example the rule $p(X) \rightarrow q(X,Y)$ can be either interpreted as $\forall X\, p(X) \rightarrow \exists Y\, q(X,Y)$ or $\forall X, Y\, p(X) \rightarrow q(X,Y)$. Example 5 shows how this affects reasoning.

   Example 5 *Consider the following situation where "jack" is a murderer and "john" is a victim. A murderer is a person who killed someone. This situation is described in $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$ with $\mathscr{F} = \{\top \Rightarrow murderer(jack) \wedge\ victim(john)\}$ and $\mathscr{R} = \{murderer(X) \rightarrow killed(X,Y)\}$. If we run the query $Q = killed(jack, john)$? (did jack kill john?) using a tool that does not take into account existential variables the answer would be **"true"** because it assumes that all known constants (persons) are killed by all murderers (i.e. $\forall X, Y\, murderer(X) \wedge \top(Y) \rightarrow killed(X,Y)$). In fact, it will also consider that $killed(jack, jack)$ is "true".*
   *However if we run the query Q using a tool that supports existential variables, the answer would be **"false"** since it is not possible to make the link between the generated null and the constant "john".*

2. *Rules with cycles:* We consider two types of cycles: Support cycles (when the Graph of Rule Dependency is cyclic) and Attack cycles. In presence of a *cyclic GRD* some inference mechanisms (such as SLD resolution [4]) might loop infinitely as it might lead to an infinite cycle of evaluating the conclusion then the premise then the conclusion and so on as shown in the following example.

   Example 6 (Support Cycle) *Consider the following knowledge base $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$ for representing legal contracts. A person is generally an individual and an individual is a person. "bob" is a person. Is "bob" an individual $Q = individual(bob)$?*
   – $\mathscr{F} = \{\top \rightarrow person(bob)\}$
   – $\mathscr{R} = \{r_1 : \forall X\ person(X) \Rightarrow individual(X),\ r_2 : \forall X\ individual(X) \Rightarrow person(X)\}$
   Evaluating Q would require evaluating the application of r_1 generating $individual(bob)$, which would require evaluating $person(bob)$, this in turn might require evaluating $individual(bob)$ given r_2 and continue on and on.

   Cyclic conflict happens when two chains of reasoning attack each other at different levels as shown in Example 7. Some logics (such as Defeasible Logics) would loop infinitely in such situations.

Example 7 (Attack Cycle) *Let* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \succ)$*:*
- $\mathscr{F} = \{\top \Rightarrow p(a), \top \Rightarrow q(a)\}$.
- $\mathscr{R} = \{\forall X\, p(X) \Rightarrow \neg q(X), \forall X\, q(X) \Rightarrow \neg p(X)\}$

Evaluating the query $Q = \neg p(a)$ *would result in an infinite cycle if the arguments are evaluated on construction. Otherwise the answer to* Q *is* ***'false'****.*

3. *Rule Application Block:* Some situations require that rules are prevented from being applied, to express that a conclusion should not be derived and at the same time its complement is not necessarily derived either. This solves some non-intuitive results of certain logics [23]. Blocking rule applications can be achieved either by giving rules labels and considering them as atoms, or by using defeater rules.

   Example 8 *Birds generally fly. We cannot say that birds with broken wings can fly or not. Let* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$*:*
   - $\mathscr{F} = \{\top \Rightarrow bird(tweety) \wedge brokenWings(tweety)\}$.
   - $\mathscr{R} = \{r_1 : \forall X\, bird(X) \Rightarrow fly(X),\ r_2 : \forall X\, brokenWings(X) \rightsquigarrow \neg fly(X)\}$.

   Or, if we use rule labels:
   - $\mathscr{F} = \{\top \Rightarrow bird(tweety) \wedge brokenWings(tweety)\}$.
   - $\mathscr{R} = \{r_1 : \forall X\, bird(X) \Rightarrow fly(X),\ r_2 : \forall X\, brokenWings(X) \Rightarrow \neg r_1\}$.

   The answer to the query $Q = fly(tweety)$ *is* ***"false"****.*

4. *Priority Relation:* used to resolve ambiguities, can be variously expressed (cardinal, ordinal or implicitly - such as generalized specificity [13]).
5. *Type of Queries:* All defeasible tools support at least boolean ground queries, some allow for boolean existentially closed (non ground) queries.

4 Benchmark Description

The benchmark provides indications on how defeasible reasoning tools are handling the previously described features (their support and subsequent scaling up).
Benchmarking Methodology. We build upon the *propositional* defeasible logic performance oriented benchmark from [19] that generates various parameterized knowledge bases (also known as theories). We *adapt existing theories for the first order language* and *extend them with twelve additional theories* to account for features listed in the previous section. These theories serve two purposes: first, to test the tools' ability to handle the features (especially when these features are not explicitly stated in the companion paper of the tools). Second, to test their performance when faced with gradually complex situations requiring these features. For example: does the tool allow for team defeat? How does it perform when there are larger and larger instances requiring team defeat? Before defining the benchmark, two key notions must be kept in mind:

1. To test the support for a semantics feature, this feature must be "isolated", meaning that the result of the query must only depend on the feature and no other external factor. That is why most theories use only defeasible rules (to avoid the disruptive effect of handling strict rules) and no preferences.

2. While negative results (i.e. situations where tools are not able to give the results required by a certain feature) are definitive, positive results (i.e. situations where tools do provide the intended results of a feature) on the other hand *do not prove the feature is fully supported*.

Benchmark Theories. Figures 1 and 2 give a representation of the benchmark theories, dashed lines represent conflict, $\Rightarrow$ and $\rightsquigarrow$ are defeasible and defeater rules respectively.

- **Ambiguity:** (denoted $ambiguity(n)$) contains a chain of n rules $s_{i-1}(a) \Rightarrow s_i(a)$, and two chains of $2n$ rules $q_{i-1}(a) \Rightarrow q_i(a)$ and $p_{i-1}(a) \Rightarrow p_i(a)$, plus the rules $s_n(a) \Rightarrow \neg q_n(a)$ and $q_{2n}(a) \Rightarrow \neg p_{2n}(a)$, and the defeasible facts $s_0(a)$, $q_0(a)$, $p_0(a)$. The query $Q = p_{2n}(a)?$ is not entailed (false) in ambiguity propagating, but is entailed (true) in ambiguity blocking. The parameter n allows the scaling of the theory to longer and longer chains where conflicts appear further down the line.
- **Team Defeat:** (denoted $team(n)$) each conclusion is supported by a team of two defeasible rules and attacked by another team of two defeasible rules. Preferences ensure that each attacking rule is inferior to one of the supporting rules. The antecedents of these rules are in turn supported and attacked by cascades (n levels) of teams of rules. The query $Q = p_0(a)?$ is entailed if team defeat is allowed.
- **Floating Conclusions:** (denoted $floating(n)$) contains n couples of conflicting rules $\top \Rightarrow p_i(a)$ and $\top \Rightarrow \neg p_i(a)$, and n rules $p_i(a) \Rightarrow q(a)$. The query $Q = q(a)?$ is entailed if floating conclusions are allowed.
- **Consistent Derivation:** (denoted $consistent(n)$) contains two defeasible facts $p_0(a)$, $q_0(a)$ and two chains of n strict rules of the form $p_{i-1}(a) \rightarrow p_i(a)$ and $q_{i-1}(a) \rightarrow q_i(a)$ that lead to a conflict down the line because of the rules $p_n(a) \rightarrow p_{n+1}(a)$ and $q_n(a) \rightarrow \neg p_{n+1}(a)$. The query $Q = p_0(a)$ is not entailed if atoms need to be consistent w.r.t. strict rules (indirectly consistent derivation), otherwise it is entailed.

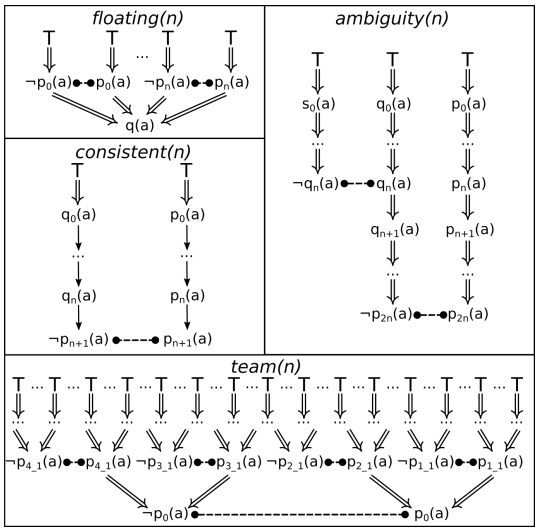

Fig. 1: Representation of semantics theories

– **Existential Rules:** (denoted $exist(n)$) composed of n rules $\top \Rightarrow p(a_i)$, and the rule without quantifiers $p(X) \Rightarrow q(X,Y)$. $Q = q(a_0,a_n)$? is not entailed if existential rules are supported.
– **FES:** (denoted $transitive(n)$) contains the rule $\top \Rightarrow p_0(a,b) \wedge p_0(b,c)$, and a chain of n *transitive rules* of the form $\forall X,Y,Z,\ p_{i-1}(X,Y) \wedge p_{i-1}(Y,Z) \Rightarrow p_{i-1}(X,Z) \wedge p_i(X,Y) \wedge p_i(Y,Z)$. Evaluating $Q = p_n(a,c)$? would result in an infinite loop if FES rules are not supported.
– **FUS:** (denoted $chainAtomB(n)$) contains the rule $\top \Rightarrow p_0(a,b)$, and a chain of n *atomic body rules* of the form $\forall X,Y,\ p_{i-1}(X,Y) \Rightarrow \exists Z,\ p_{i-1}(X,Z) \wedge p_{i-1}(Z,Y) \wedge p_i(X,Y)$. The query $Q = p_n(a,b)$? would result in an infinite loop if FUS existential rules are not supported.
– **GBTS:** (denoted $chainFrG(n)$) contains the rule $\top \Rightarrow p_0(a,b) \wedge p_0(b,c)$, and a chain of n *frontier-guarded rules* of the form $\forall X,Y,Z,\ p_{i-1}(X,Y) \wedge p_{i-1}(Y,Z) \Rightarrow \exists W,\ p_{i-1}(X,W) \wedge p_{i-1}(W,Y) \wedge p_i(X,Y) \wedge p_i(Y,W)$. Evaluating the query $Q = p_n(a,b)$? would result in an infinite loop if GBTS existential rules are not supported.
– **Cyclic GRD:** (denoted $cyclic(n)$) contains the rule $\top \Rightarrow p_1(a)$ and a cyclic chain of n defeasible rules as $p_i(X) \rightsquigarrow p_{i \bmod n}(X)$. Evaluating $Q = p_0(a)$? results in a loop.
– **Circular Reasoning:** (denoted $circular(n)$) consists of a defeasible fact $\neg p_0(a)$ and a cycle of rules of the form $p_i(a) \Rightarrow p_{i \bmod n}(a)$. Evaluating the query $Q = \neg p_0(a)$ might result in an infinite loop due to circular reasoning.
– **Cyclic Conflict:** (denoted $cyclicConf(n)$) composed of n cyclic conflict of the form $p_i(a) \Rightarrow \neg q_i(a)$ and $q_i(a) \Rightarrow \neg p_i(a)$. Evaluating $Q = p_{n+1}(a)$? might loop infinitely.
– **Rule Block:** (denoted $ruleBlock(n)$) contains n rules $\top \Rightarrow p_i(a)$ and $p_i(a) \Rightarrow q(a)$, and a single defeater rule $\top \rightsquigarrow \neg q(a)$ that blocks all other rules. The queries $Q = q(a)$? is not entailed. The parameter n determines how many rules have to be blocked. This theory tests performance with regards to handling rule applications blocking.
– **Priority:** (denoted $levels(n)$) is a cascade of n disputed conclusions $p_i(a)$. For each i, there are rules $r_i : \top \Rightarrow p_i(a)$ and $r'_i : p_{i+1} \Rightarrow \neg p_i(a)$. For each *odd* $i \geq 1$ a priority asserts that $r'_i \succ r_i$. A final rule $\top \Rightarrow p_{n+1}(a)$ gives uncontested support for $p_{n+1}(a)$. $Q = p_0(a)$? is entailed if priorities are respected.
– **Queries:** (denoted $query(n)$) composed of n rules $\top \Rightarrow p(a_i)$, and the rule $\forall X,\ p(X) \Rightarrow q(X)$. The query $Q = \exists X q(X)$? is entailed if existentially closed queries are supported (as there are n atoms of the form $q(a_i)$).

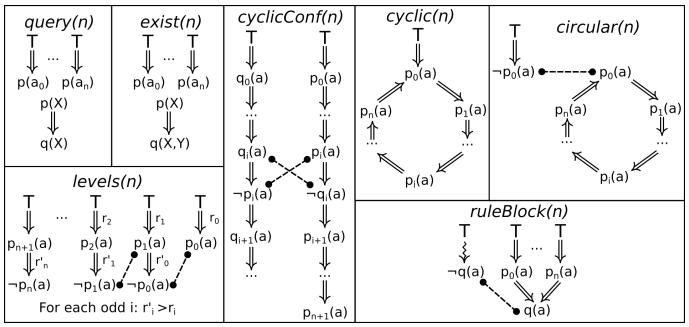

Fig. 2: Representation of some expressiveness theories

- **Chain Theory** (denoted $chain(n)$) contains the rule $\top \Rightarrow p_0(a)$ and a chain of n defeasible rules of the form $p_{i-1}(X) \Rightarrow p_i(X)$. Evaluating the query $Q = p_n(a)$? would test performance when faced with a long chain of rules.
- **Tree Theory** (denoted $tree(n,k)$) is a k-branching tree of depth n in which every atom occurs once and $p_0(a)$ is its root. The query $Q = p_0(a)$? would test performance w.r.t. a large number of short arguments.
- **Directed Acyclic Graph Theory** (denoted $dag(n,k)$) is a k-branching tree of depth n in which every literal occurs k times. The query $Q = p_0(a)$? would test performance when faced with many arguments for the same atom.

The generated knowledge bases have to be adapted to the format of each tool. For example, rules can be transformed into an equivalent set of rules with atomic head, priority relation can be transformed into a number based priority, negation can be represented with negative constraints, etc. We conclude this section by an important remark. Benchmark support for abstract classes of existential rules such as FES, FUS and GBTS is achieved via concrete decidable classes. Not satisfying a concrete class (e.g. transitive rules) implies not satisfying the corresponding abstract class (e.g. FES rules). However, please note that the inverse is not necessarily true. More generally, while negative results (i.e. situations where tools are not able to give the results required by a certain feature) are definitive, positive results on the other hand (i.e. situations where tools do provide the intended results of a feature) *do not prove the feature is fully supported*.

5 Running the Benchmark on Tools

Let us start by presenting the main defeasible reasoning implementations we used in this paper to illustrate how the proposed benchmark can be used for tool profiling (that are, to the best of our knowledge, the only still functioning, publicly available tools for first order defeasible reasoning as of the time of writing of this paper). **ASPIC+** [24] is a framework for specifying systems in structured argumentation used for defeasible reasoning thanks to its grounded semantics (equivalent to defeasible reasoning with ambiguity propagation [15]). We use a prototype implementation available online[4] (denoted here by *ASPIC**). Other implementations use a propositional language only. **DEFT**[5] [16] is a first-order defeasible reasoning tool for existential rules that uses the $Datalog^{\pm}$ language. It relies on a dialectical tree mechanism (that can correspond to ambiguity propagation [13]). **DeLP** [13] (Defeasible Logic Programming) is a formalism that relies on dialectical trees to allow for defeasible reasoning with ambiguity propagation. We use the implementation in Tweety1.7 libraries [26] (denoted here by *DeLP**). An online tool called DeLP client[6] is also available. **Flora-2** [28, 27] is a rule-based knowledge base system designed for a variety of automated tasks on the Semantic Web, ranging from meta-data management to intelligent agents. It has a commercial version called Ergo with additional functionalities.

[4] `http://aspic.cossac.org`
[5] `https://github.com/hamhec/DEFT`
[6] `http://lidia.cs.uns.edu.ar/delp\_client`

A key notion to keep in mind is that we are not comparing the formalisms themselves, we are comparing the tools based on those formalisms. A formalism might allow for more than what the tool presents. That is one of the reasons that justify having a dedicated benchmark to better analyze and understand the implementation of the tools. Furthermore, some of the tools considered are prototypes, therefore they might not have been developed with performance in mind. All experiments presented in this section were performed on an Intel core i7 2.60GHz quad core Linux machine with 8GB of RAM and a Java heap of 2GB. To avoid random performance fluctuations each test is performed five times for each tool and we record the average *in-CPU* execution time. The experiments are reproducible[7].

Tools Benchmark Results. Table 1 presents the time (in CPU seconds) required for each tool to answer the query according to the size and type of the query (the execution time includes the time needed for parsing the knowledge base and answering the query). ∞ denotes a stack overflow, $T.O.$ denotes a timeout (set to 5 minutes), and $N.A.$ indicates that a test theory is not applicable for that tool.

Theory		*ASPIC**	*DeLP**	SPINdle	Flora-2	DEFT
$ambiguity(n)$	$n=50$	0.44 (false)	148.17 (false)	**0.11 (false)** | **0.09 (true)**	1.06 **(true)**	**0.11 (false)**
	$n=1800$	∞	$T.O.$	∞	**17.237 (true)**	**12.503 (false)**
	$n=2000$	∞	$T.O.$	∞	**18.942 (true)**	∞
$team(n)$	$n=4$	**0.227 (false)**	301.19 (true)	0.289 (true)	4.358 (true)	**0.287 (true)**
	$n=7$	$T.O.$	$T.O.$	**109.46 (true)**	$T.O.$	201.917$(true)$
$floating(n)$	$n=100$	0.270 (false)	209.45 (false)	**0.332 (false)**	2.143 (false)	1.345 (false)
	$n=5000$	198.861 (false)	$T.O.$	**150.144 (false)**	$T.O.$	203.18 (false)
$consistent(n)$	$n=1000$	0.193 (true)	269.984 (false)	**0.703 (true)**	5.292 (true)	**2.969 (false)**
	$n=8000$	8.321 (true)	$T.O.$	**8.854 (true)**	36.821 (true)	239.504$(false)$
$exist(n)$	$n=100$	0.09 (true)	0.93	284.39 (true)	1.28 (true)	**0.01 (false)**
$transitive(n)$	$n=100$	$N.A.$	$N.A.$	$N.A.$	$N.A.$	**253.62 (true)**
$chainAtomB(n)$	$n=1$	$N.A.$	$N.A.$	$N.A.$	$N.A.$	$T.O.$
$chainFrG(n)$	$n=1$	$N.A.$	$N.A.$	$N.A.$	$N.A.$	$T.O.$
$cyclicSupp(n)$	$n=1000$	∞	291.37 (true)	**0.35 (true)**	5.712 (true)	0.44 (true)
	$n=10000$	∞	$T.O.$	**26.61 (true)**	51.72 (true)	288.71 (true)
$circular(n)$	$n=1000$	∞	284.90 (true)	0.31 (true)	4.38 (true)	**0.04 (true)**
$cyclicConf(n)$	$n=5$	0.627 (true)	55.89 (true)	0.903 (true)	0.922 (true)	**0.106 (true)**
	$n=1000$	$T.O.$	$T.O.$	$T.O.$	$T.O.$	**79.525 (true)**
$ruleBlock(n)$	$n=500$	19.23 (true)	$N.A.$	**1.41 (true)**	12.99 (true)	$N.A.$
$levels(n)$	$n=100$	**0.20** (true)	4.61 (true)	0.33 (true)	5.17 (true)	0.81 (true)
$query(n)$	$n=100$	$N.A.$	$N.A.$	$N.A.$	**1.25** (true)	$N.A.$
$chain(n)$	$n=600$	108.05 (true)	99.46 (true)	**0.24 (true)**	2.35 (true)	0.33 (true)
	$n=10000$	∞	$T.O.$	**16.04 (true)**	45.44 (true)	288.71 (true)
$tree(n,5)$	$n=2$	0.04 (true)	193.64 (true)	**0.03 (true)**	0.83 (true)	0.022 (true)
	$n=7$	∞	$T.O.$	∞	211.94 (true)	**182.83 (true)**
$dag(n,10)$	$n=1$	∞	239.75 (true)	**7.51 (true)**	18.41 (true)	19.53 (true)
	$n=10$	∞	$T.O.$	**60.82 (true)**	113.53 (true)	73.05 (true)

Table 1: Execution time in seconds (selected results). 'true' and 'false' indicate query entailment and are used to check support of the feature (the best time is shown in bold)

[7] `https://github.com/anoConf/Benchmark`

Results discussion. The main objective of the proposed benchmark is to help with the tools profiling according to the defeasible knowledge base and reasoning they can handle. To this end, from the results in Table 1, we can draw the following conclusions (summarised in Table 2):

Feature		*ASPIC**	*DeLP**	SPINdle	Flora-2	DEFT
Ambiguity	Prop.	✓	✓	✓	-	✓
	Block.	-	-	✓	✓	-
Team Defeat	TD	-	✓	✓	✓	✓
	noTD	✓	-	-	-	-
Floating Conclusions	FC	-	-	-	-	-
	noFC	✓	✓	✓	✓	✓
Consistent Derivation	Direct	-	✓	-	-	✓
	Indirect	✓	-	✓	✓	-
Existential Rules	S-FES	-	-	-	-	✓
	FUS	-	-	-	-	-
	GBTS	-	-	-	-	-
Cycles	Support	-	✓	✓	✓	✓
	Attack	✓	✓	✓	✓	✓
Rule Block		✓	-	✓	✓	-
Preference	$\succ$	-	✓	✓	✓	✓
	$\mathbb{R}$	✓	-	-	-	-
Non-gound Queries		-	-	-	✓	-

Table 2: Classification results (✓indicates the tool supports the feature).

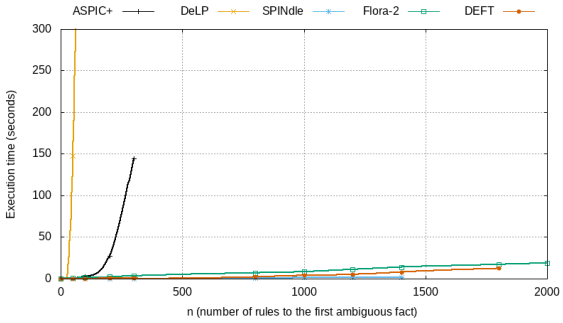

Fig. 3: Response time for *ambiguity*(n)

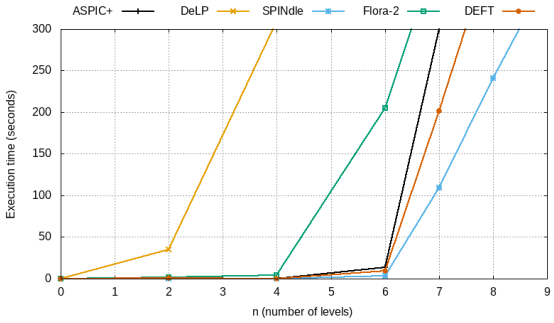

Fig. 4: Response time for *team*(n)

– **Semantics:** The underlying theoretical and practical choices affect the semantics the tools can handle: *Ambiguity Handling: ASPIC**, DEFT and *DeLP** cannot express ambiguity blocking and correspond to ambiguity propagation due to the underlying formalisms. Flora corresponds only to ambiguity blocking while SPINdle

is the only tool that can handle both blocking and propagating. Performance wise (Figure 3), *DeLP** has a timeout at $n = 10$, *ASPIC** stops at $n = 300$, SPINdle at $n = 1400$, and DEFT $n = 1800$ due to stack overflow. Flora-2, DEFT, and SPINdle can scale to longer chain of rules for ambiguous facts with significantly low response time. *Team Defeat:* Most tools allow only for team defeat except *ASPIC** that does not allow for it. While Defeasible Logics can represent the presence and absence of team defeat, SPINdle and Flora-2 only implement the presence of team defeat. SPINdle has the best performance, followed by DEFT, *ASPIC**, Flora-2, then *DeLP**. *Floating Conclusions:* None of the considered tools support floating conclusions due to their underlying formalisms. *Handling Strict Rules:* DEFT and *DeLP** use indirectly consistent derivations while *ASPIC**, Flora and SPINdle do not. This directly impacts performance results (as seen in Figure 6).

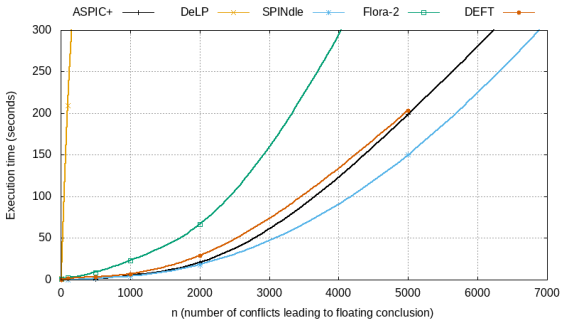

Fig. 5: Response time for $floating(n)$

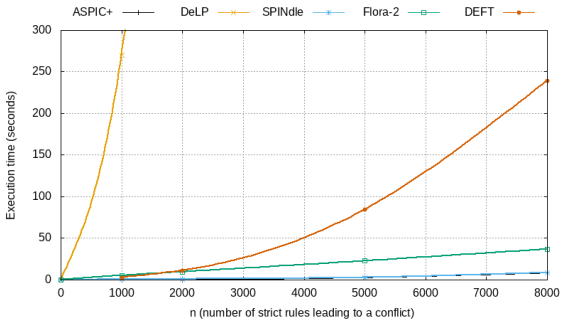

Fig. 6: Response time for $consistent(n)$

– **Expressiveness:** The choice of the inference mechanism affects the expressiveness the tool can handle: *Existential rules:* *ASPIC**, *DeLP**, SPINdle, and Flora were not designed to account for existential rules. As supported by the results of the Existential theory, FES, FUS and GBTS are not applicable in their context. DEFT can handle existential rules in general, and SkolemFES rules in particular (due to its use of forward chaining), however it loops infinitely in FUS and GBTS fragments. *Cycles:* DEFT, *DeLP**, and Flora can handle cyclic GRDs and circular reasoning (*support cycle*) contrary to ASPIC+. This is due to the fact that *ASPIC** relies on SLD resolution (which loops infinitely in presence of cycles in the GRD), while DEFT uses a chase mechanism (which is guaranteed to stop when no existential rule is used). *DeLP**, SPINdle and Flora rely on resolution with a grounding phase and cycle checks. All considered tools can handle cyclic conflicts (*attack cycles*). However, the attack cycle checks are not needed for DEFT since arguments are evaluated after construction, that is why it outperforms other tools (e.g. $n = 1000$ in Figure 8). *Rule Application Block:* *ASPIC** uses negated labels of rules to block their application, SPINdle uses defeater rules, while Flora uses the predicate '$\backslash cancel(label)$'. DEFT and *DeLP** have no support for such feature. As seen in Table 1, SPINdle has the best performance followed by Flora-2 and *ASPIC**. *Preference between rules:* *ASPIC** uses decimal values to express priority on rules (this priority relation is total and might lead to unwanted be-

havior as it is hard to express incomparability between rules). *DeLP**, SPINdle, Flora, and DEFT use a partial priority relation based on labeled rules. *Non-ground queries:* Only Flora supports non ground queries.

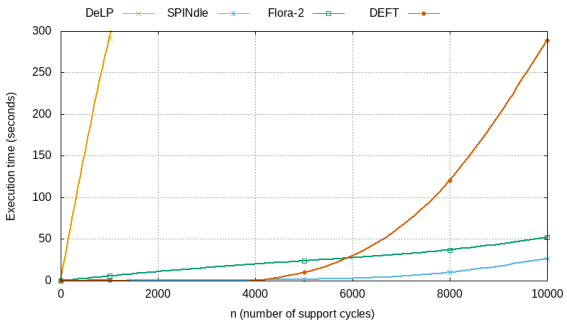

Fig. 7: Response time for $cyclicSupp(n)$

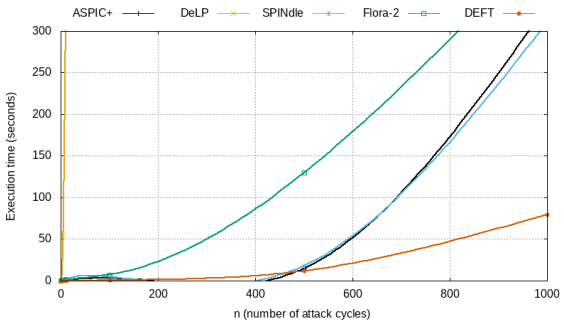

Fig. 8: Response time for $cyclicConf(n)$

– **Performance:** In case of a tie in expressiveness or semantics, one can use performance to make an informed choice on the tool to be used. From Table 1 we can see that each tool makes trade-offs between performance and expressiveness. In general, SPINdle has the best performance compared to other tools, followed by DEFT and Flora-2. *ASPIC** is as fast as SPINdle on small knowledge bases but it does not scale well. DEFT has the best performance when there are attack cycles. These differences in performance are due to three main factors. First, *grounding phase is costly*. DEFT, for instance, achieves its performance results thanks to its forward chaining algorithms that ground rules on the fly, contrary to the other tools. Second, *handling cycles is costly*. *ASPIC** is faster than *DeLP** for example because the latter relies on cycle checks to avoid infinite loops. DEFT does not need to check loops contrary to all other tools. Third, *expressiveness is costly*, DEFT and *DeLP** has to perform consistency checks using strict rule. Flora also provides very powerful syntactic features (dynamic rule labels, higher order syntax, etc.) which might affect its performance.

Tool profiling. let us show how a data engineer could make practical use of our benchmark in order to chose the appropriate defeasible reasoning tool.

Example 9 *Consider the following decision scenario of an emergency response team that wants to determine if a person victim of an accident is an organ donor. A person being hurt in an accident is considered a victim. Legally any victim is assumed not to be an organ donor. A person that gives her consent is considered an organ donor. A person in a critical condition generally cannot give her consent. The legal tutor of a person can give his consent for her being an organ donor. The following KB describes this use case.* $\mathscr{K} = (\mathscr{F}, \mathscr{R}, \emptyset)$ *where:*

– $\mathscr{F} = \{\top \Rightarrow hurt(john)\}$.
– $\mathscr{R} = \{r_1 : \forall X\, hurt(X) \Rightarrow victim(X),\ r_2 : \forall X,\ victim(X) \Rightarrow \neg OrganDonor(X),$
$r_3 : \forall X, consentFor(X,X) \Rightarrow organDonor(X),\ r_4 : \forall X,\ critical(X) \Rightarrow \neg ConsentFor(X,X)$
$r_5 : \forall X,Y,\ legalTutor(X,Y) \wedge consentFor(X,Y) \Rightarrow organDonor(Y),\ \}$.

This knowledge base does not use existential rules and is acyclic. Therefore, given the results of the benchmark, all considered tools can be applied. If the data engineer wants to use ambiguity blocking with team defeat then she can either use SPINdle or Flora-2 (SPINdle is in this case recommended given the performance results), if she does not want to allow for team defeat then ***no tool can be used****. If on the other hand the data engineer wants to use ambiguity propagation then she can either use DeLP* or DEFT if she needs team defeat (DEFT is in this case recommended given the performance results) or ASPIC* if she does not. Let us add the rule that a victim is probably someone who is hurt ($\forall X$, $victim(X) \Rightarrow hurt(X)$). In this case the knowledge base becomes cyclic. Therefore ASPIC* cannot be used (cf. Table 2). Let us now add a new existential rule stating that if someone is an organ donor then somebody gave his consent (the person or her tutor i.e. $\forall X\, organDonor(X) \rightarrow \exists Y\, consentFor(Y,X)$). In this case, according to the Table 2 results, only DEFT can be used.*

6 Discussion

We proposed an informative benchmark allowing to shed light on the capabilities of existing tools for defeasible reasoning. This paper presents an exhaustive list of operational first order defeasible reasoning tools; other implementations such as DR-Device [7] and DR-Prolog [8] are not maintained and propositional defeasible reasoning tools of [10] are not available. Our study provides insights about current gaps in the state of the art. For instance, we can observe in Table 2 that some features such as floating conclusions are not supported by any tool, or that support for some desirable combinations of features (such as ambiguity propagating with team defeat) is lacking.

The list of features does not consider (1) expressing conflicting literals using negative constraint, (2) implicit priority relations, and (3) negation-as-failure. The former is due to the fact that conflicts can easily be translated between the two representations [1], e.g. $\neg p(X)$ is transformed into $np(X)$ and the negative constraint $p(X) \wedge np(X) \rightarrow \bot$ is added to the rules). Regarding implicit priorities, they sometimes can be represented using explicit ones [23]. Negation-as-failure can treated by theory rewriting [2].

References

1. G. Antoniou. Defeasible reasoning: A discussion of some intuitions. *International Journal of Intelligent Systems*, 21(6):545–558, 2006.
2. G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic (TOCL)*, 2(2):255–287, 2001.
3. G. Antoniou, D. Billington, and M. J. Maher. On the analysis of regulations using defeasible rules. In *Systems Sciences, 1999. HICSS-32. Proceedings of the 32nd Annual Hawaii International Conference on*, pages 7–23. IEEE, 1999.
4. K. R. Apt and M. H. Van Emden. Contributions to the theory of logic programming. *Journal of the ACM (JACM)*, 29(3):841–862, 1982.
5. J.-F. Baget, F. Garreau, M.-L. Mugnier, and S. Rocher. Extending Acyclicity Notions for Existential Rules. In *ECAI*, pages 39–44, 2014.
6. J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.
7. N. Bassiliades, G. Antoniou, and I. Vlahavas. A defeasible logic reasoner for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2(1):1–41, 2006.

8. A. Bikakis and G. Antoniou. DR-Prolog: a system for reasoning with rules and ontologies on the semantic web. In *AAAI*, volume 5, pages 1594–1595, 2005.
9. D. Billington. Defeasible Logic is Stable. *Journal of logic and computation*, 3(4):379–400, 1993.
10. D. Bryant and P. Krause. A review of current defeasible reasoning implementations. *The Knowledge Engineering Review*, 23(03):227–260, 2008.
11. A. Cali, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *Proceedings of the VLDB Endowment*, 3(1-2):554–565, 2010.
12. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.
13. A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory and practice of logic programming*, 4(1+ 2):95–138, 2004.
14. D. R. Garcia, A. J. Garcia, and G. R. Simari. Planning and defeasible reasoning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 856—-858. ACM, 2007.
15. G. Governatori, M. J. Maher, G. Antoniou, and D. Billington. Argumentation Semantics for Defeasible Logic. *Journal of Logic and Computation*, 14(5):675–702, 2004.
16. A. Hecham, M. Croitoru, and P. Bisquert. Argumentation-Based Defeasible Reasoning For Existential Rules. In *Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent Systems*, pages 1568—-1569, 2017.
17. J. F. Horty, D. S. Touretzky, and R. H. Thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 476–482, 1987.
18. M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.
19. M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal on Artificial Intelligence Tools*, 10(04):483–501, 2001.
20. D. Makinson and K. Schlechta. Floating conclusions and zombie paths: two deep difficulties in the "directly skeptical" approach to defeasible inheritance nets. *Artificial intelligence*, 48(2):199–209, 1991.
21. L. Morgenstern. Artificial Intelligence Inheritance comes of age : applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103:237–271, 1998.
22. D. Nute. *Defeasible reasoning: a philosophical analysis in prolog*. Springer, 1988.
23. H. Prakken. Intuitions and the modelling of defeasible reasoning: some case studies. In *Ninth Int Workshop on Nonmonotonic Reasoning*, pages 91–99, Toulouse, 2002.
24. H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.
25. L. A. Stein. Resolving ambiguity in nonmonotonic inheritance hierarchies. *Artificial Intelligence*, 55(2-3):259–310, 1992.
26. M. Thimm. Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14)*, 2014.
27. H. Wan, M. Kifer, and B. N. Grosof. Defeasibility in answer set programs with defaults and argumentation rules. *Semantic Web*, 6(1):81–98, 2015.
28. G. Yang, M. Kifer, and C. Zhao. 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*, pages 671–688, 2003.