EMIL: Extracting Meaning from Inconsistent Language

Towards argumentation using a controlled natural language interface

Hannes Strass [a], Adam Wyner [b,*] and Martin Diller [c]
[a] *REVISE Computer Science Institute, Leipzig University, Germany*
E-mail: strass@informatik.uni-leipzig.de
[b] *School of Law and Department of Computer Science, Swansea University, United Kingdom*
E-mail: a.z.wyner@swansea.ac.uk
[c] *Institute for Information Systems, Technical University Vienna, Austria*
E-mail: mdiller@kr.tuwien.ac.at

Abstract.
There are well-developed formal and computational theories of argumentation to reason in the face of inconsistency, some with implementations; there are recent efforts to extract arguments from large textual corpora. Both developments are leading towards automated processing and reasoning with inconsistent, linguistically expressed knowledge in order to provide explanations and justifications in a form accessible to humans. However, there remains a gap between the knowledge-bases of computational theories of argumentation, which are generally coarse-grained and semi-structured (e.g. propositional logic), and inferences from knowledge-bases derived from natural language, which are fine-grained and highly structured (e.g. predicate logic). Arguments that occur in textual corpora are very rich, highly various, and incompletely understood. We identify several subproblems which must be addressed in order to bridge the gap, requiring the development of a computational foundation for argumentation coupled with natural language processing. For the computational foundation, we provide a *direct semantics*, a formal approach for argumentation, which is implemented and suitable to represent and reason with an associated natural language expression for defeasibility. It has attractive properties with respect to expressivity and complexity; we can reason by cases; we can structure higher level argumentation components such as cases and debates. With the implementation, we output experimental results which emphasise the importance of our efficient approach. To motivate our formal approach, we identify a range of issues found in other approaches. For the natural language processing, we adopt and adapt an existing controlled natural language (CNL) to interface with our computational theory of argumentation; the tool takes natural language input and automatically outputs expressions suitable for automated inference engines. A CNL, as a constrained fragment of natural language, helps to control variables, highlights key problems, and provides a framework to engineer solutions. The key adaptation incorporates the expression 'it is usual that', which is a plausibly 'natural' linguistic expression of defeasibility. This is an important, albeit incremental, step towards the incorporation of linguistic expressions of defeasibility; yet, by engineering such specific solutions, a range of other, relevant issues arise to be addressed. Overall, we can input arguments expressed in a controlled natural language, translate them to a formal knowledge base, represent the knowledge in a rule language, reason with the rules, generate argument extensions, and finally convert the arguments in the extensions into natural language. Our approach makes for fine-grained, highly structure, accessible, and linguistically represented argumentation evaluation. The overall novel contribution of the paper is an integrated, end-to-end argumentation system which bridges a gap between automated defeasible reasoning and a natural language interface. The component novel contributions are the computational theory of 'direct semantics', the motivation for our theory, the results with respect to the direct semantics, the implementation, the experimental results, the tie between the formalisation and the CNL, the adaptation of a CNL defeasibility, and an 'engineering' approach to fine-grained argument analysis.

Keywords: Argumentation, Non-monotonic reasoning, Controlled natural language, Defeasible reasoning

*Corresponding author. E-mail: a.z.wyner@swansea.ac.uk.

1. Introduction

Approaches to artificial intelligence in general and to automated problem solving in particular should – in virtue of their intelligence – explain and justify their conclusions and actions in a rational discourse. This is not always done in machine learning approaches: the Go playing computer program *AlphaGo* [89], while very proficient in choosing the right move (i.e. solving a range of problems), cannot explain to a human user *why* it chose that particular move (i.e. justifying its solution). A recent Nature editorial concluded that "[t]he machine becomes an oracle; its pronouncements have to be believed." (Nature 529, p. 437) Even the popular press discusses the problems of explanations derived from machine learning approaches to artificial intelligence.[1] While there may be many applications where no explanation or justification is needed, for others explanation and justification may be essential, e.g. medical diagnosis and legal argumentation [33]. For instance, it may not be sufficient to automatically determine cases based on machine learning, where it is important to challenge or appeal outcomes based on specific aspects of the reasoning or procedure. It is often important to scientific progress to have theories and structures that can be tested and modified. As well as explain and justify, artificial intelligence must often address nonmonotonicity, where knowledge bases are inconsistent or change.

To explain/justify and reason nonmonotonically, we set this work in the context of computational argumentation, where a central aim is to reason from premises to a conclusion using a rule, e.g. the reasoning pattern *Modus Ponens* is an argument in classical propositional logic. Recent efforts to formalise and instantiate "abstract" argumentation have focused on reasoning with defeasible knowledge bases (KBs) [34, 80 among others]. We also want to use arguments to communicate amongst human users. In the area of computational semantics of natural language, arguments can be linguistically expressed, translated to a formal language, and reasoned with to conclusions [17], though without addressing defeasibility. In argument mining, efforts are made to extract linguistically expressed arguments (constructs of rules, premises, and conclusions) from large textual corpora, which are structured into chains for reasoning [70, 77]. The mined expressions are classified sentences, yielding essentially propositional forms without further fine-grained internal structure. Between abstract and instantiated argumentation and linguistically represented arguments, there is a substantial gap: while abstract argumentation can reason with defeasible KBs, its abstraction from linguistic information limits its applicability; for current computational semantic approaches, reasoning with inconsistency is problematic and generic reasoning is absent. Moreover, while argument mining can extract information from textual corpora, it does not take the next step to extract the fine-grained, highly structured linguistic information for semantic representations, i.e. at the level of predicate logic, that are needed for inference in many domains. Yet, arguments that occur in textual corpora are very rich, highly various, and incompletely understood; classified sentences vary greatly in terms of word choice, syntactic form, and discourse relations. In our approach, we adopt and adapt an existing controlled natural language (CNL) to interface with our computational theory of argumentation; the tool takes natural language input and automatically outputs expressions suitable for automated inference engines. A CNL, as a constrained fragment of natural language, helps to control linguistic variables, highlights key problems, and provides a framework to engineer solutions; the CNL perhaps could be applied to the outputs of argument mining in an overall framework of refinement. The key adaptation incorporates the expression 'it is usual that', which is a plausibly 'natural'

[1] New York times, November 21, 2017 `https://www.nytimes.com/2017/11/21/magazine/can-ai-be-taught-to-explain-itself.html?_r=0`. Accessed November 24, 2017

linguistic expression of defeasibility. This is an important, albeit incremental, step towards the incorporation of linguistic expressions of defeasibility; yet, by engineering such specific solutions, a range of other, relevant issues arise to be addressed.

The overall novel contribution of the paper is an integrated, end-to-end argumentation system which is an incremental step towards bridging the gap between automated defeasible reasoning along the lines of predicate logic and a natural language interface. The overall contribution has a range of components: the computational theory of 'direct semantics', the motivation and desiderata for our theory, the results with respect to the direct semantics, the implementation, the experimental results, the tie between the formalisation and the CNL, the adaptation of a CNL with a linguistic expression for defeasibility, and an "engineering" approach to fine-grained argument analysis. We dub our pipeline *EMIL* for *Extracting Meaning from Inconsistent Language*. A significant underlying theme of the paper is the interdisciplinary integration of the analysis of natural language expressions, a logical representation of the expressions, and a computable rule language for the logical representation. In Section 1.1, we elaborate on the context, motivation, and direction for this work. We detail the main contributions and outline the remainder of the paper in Section 1.2.

1.1. Setting the context

In this section, we preliminarily discuss some themes about computational argumentation and CNLs, which are developed further in the course of the paper, along with a motivating example to bridge between computational argumentation and CNLs.

Towards an Implementable Computational Theory for Argumentation. There are approaches to reasoning with knowledge bases consisting of strict and defeasible rules [1, 3, 16, 25, 26, 36, 80, 93, 107]. At the theoretical level, we subscribe to none in particular and opt to make our approach conceptually parametric, abstracting from the concrete reasoning back-end that is used. For our work, we only assume that the back-end receives as input a set of strict and defeasible rules, provides output in the form of conclusions with respect to some semantics, and yields (upon request) justifications of the conclusions.

However, since we also provide an implementation of our approach, on the level of concrete realisation, we clearly have to opt for one specific approach to reasoning with defeasible theories. For this, we use a straightforward semantics for defeasible theories that can be implemented via answer set programming (ASP), a declarative knowledge representation and problem-solving paradigm [46]. Our main goal with this was to have a concrete instance of a reasoning back-end that we understand well, is freely available to use and develop in our implementation, and satisfies certain basic *performance requirements*. Roughly, while other approaches to reasoning with defeasible theories exist, there are generally limits as to how "computational" such approaches are.

What exactly do we mean by a model of argumentation being "computational"? For our purposes, we identify two senses. One sense of "computational" means the model is formalised in set theory or logic; these are formal models. In this respect, there is much work on computational models of argumentation [12]. Another sense of "computational" means that the model can – in principle and in practice – be stored and processed by a computer; these are implementable (feasible) models. We leave aside degrees of implementable models and distinguish between formal models that are and are not implementable. For implementable models, a crucial aspect of storage and processing is representation size. If a model produces descriptions of infinite size, the model is not computational in principle, since infinite descriptions cannot be processed by finite machines. If a model produces descriptions of at least exponential size in the best case, it is computational in principle but not in practice, as reasoning cost (in terms of

computation time) and representation size tend to correlate positively. In this paper, we are concerned with computational models of argumentation that are formal and implementable, focusing largely on implementability.

Implementability is especially important where we want a model of argumentation that can be used to reason with defeasible theories with predicates and variables (see Section 2.5) which represent knowledge bases derived from natural language expressions (see Section 3).

Dung formalised abstract argumentation frameworks (AFs) [34], where nodes represent abstract arguments (that have no internal structure) and arcs between nodes represent attacks (inconsistency). A variety of semantics determine the extensions of arguments, which essentially are sets of acceptable arguments wherein no argument attacks another argument. Clearly, Dungian abstract AFs are computational models that are formal and implementable (in principle whenever they are finite, in practice whenever they are of reasonable size), since abstract arguments and pairs of such (that is, attacks) could be represented by bit strings (for example).

To employ Dungian abstract AFs, the argument nodes are usually constructed from a knowledge base (KB) [14, 25, 73, 80]; these are *instantiated* AFs. A widely used language for formulating KBs in this approach consists of sets of strict and defeasible rules over a propositional language with negation [25, 73, 80]. The rules are inference rules with a conjunction of literals as premises and a single literal as conclusion. A notion of contrastiveness or consistency, usually derived from the KB's informal semantics, is then used to derive attacks between the constructed arguments. Given arguments and attacks derived from the KB, an instantiated AF is obtained for calculating acceptable arguments. Those arguments then point to conclusions that would be inferred with respect to the originating KB.

Whether instantiating an AF from a KB and reasoning with it is computationally feasible depends on two factors:

(1) How complex is it to construct the AF from the KB?
(2) How large is the obtained AF (in comparison to the KB)?

For most semantics, computing the acceptable arguments of an AF is a computationally demanding task [32, 39], and the resources needed for it increase considerably as AF size increases. In view of the instantiation process reviewed above, it becomes clear that the overall approach is only computationally feasible when the instantiation produces *small* AFs.

Unfortunately, the approaches to instantiated argumentation over strict and defeasible rules that we found in the literature [25, 73, 80] have several problems. We highlight one problem here and return to others in Section 2.10. Most importantly, the approaches can produce abstract argumentation frameworks of considerable size in the worst case. More specifically, some defeasible theories lead to exponentially many structured "arguments", where (informally) an "argument" in that setting is a proof tree of rules. Defeasible implication is represented with $\Rightarrow$, while strict implication would be $\rightarrow$.

Example 1. *The sequence* $(D_n)_{n\in\mathbb{N}}$ *of rule sets is given by*

$$
\begin{aligned}
D_0 &= \{\Rightarrow p_0, \quad \Rightarrow q_0\},\\
D_1 &= D_0 \cup \{p_0 \Rightarrow p_1, \quad q_0 \Rightarrow p_1\} \text{ and}\\
D_{i+1} &= D_i \cup \{p_0, p_i \Rightarrow p_{i+1}, \quad q_0, p_i \Rightarrow p_{i+1}\} \text{ for all } i \geqslant 1.
\end{aligned}
$$

For any $n \in \mathbb{N}$, the size of D_n is linear in n, but D_n leads to 2^{n+1} arguments, among them 2^n arguments for p_n. Here are the sets A_i of arguments for D_i for $0 \leqslant i \leqslant 2$:

$$A_0 = \{[\Rightarrow p_0], [\Rightarrow q_0]\}$$
$$A_1 = A_0 \cup \{[[\Rightarrow p_0] \Rightarrow p_1], [[\Rightarrow q_0] \Rightarrow p_1]\}$$
$$A_2 = A_1 \cup \{[[\Rightarrow p_0], [[\Rightarrow p_0] \Rightarrow p_1] \Rightarrow p_2], [[\Rightarrow p_0], [[\Rightarrow q_0] \Rightarrow p_1] \Rightarrow p_2],$$
$$[[\Rightarrow q_0], [[\Rightarrow p_0] \Rightarrow p_1] \Rightarrow p_2], [[\Rightarrow q_0], [[\Rightarrow q_0] \Rightarrow p_1] \Rightarrow p_2]\}$$

A Motivating Example. The previous discussion highlights a problem about implementing formal argumentation. We want to bridge a gap between the formal approaches and language, so we turn to consider a particular issue with a current formal way of representing a linguistic argument. For human-machine communication, there are CNL tools which translate natural language into first-order logic formulas and interface to (non-monotonic) inference engines [42, 43, 54, 60]. Yet, there are still issues with defeasible and/or conflicting information. More pointedly, defeasible propositions are often modelled using "not provably not", which we show has a different interpretation than the natural expression "it is usual that" (similarly "usually"), which is a normative quantifier expression over contexts [59, 66]. The expression "not provably not" is not attested in a large corpora of English sentences[2], so is not preferable to use.

The following running example is paraphrased from Pollock [78] and illustrates these matters.

Example 2 (Moustache Murder)**.** *We have the following propositions about an armed robbery with a gunman.*

> *Jones is a person. Paul is a person. Jacob is a person. It is usual that a person is reliable. If Jones is reliable then the gunman has a moustache. If Paul is reliable then Jones is not reliable. If Jacob is reliable then Jones is reliable.*

Clearly not both Paul and Jacob can be reliable. Crucially, any semantics should provide a choice between the different (and mutually exclusive) consistent viewpoints of this narrative. An interpretation of "it is usual that" should facilitate such choices. In the approaches of [42] and [54], the expression "it is usual that" (alternatively "usually") is translated as "not provably not" (perhaps along with an abnormality predicate), e.g. a paraphrase for "it is usual that a person is reliable" is along the lines of "if a person is not provably not reliable then the person is reliable".

However, this formalisation can be incorrect, as demonstrated by its straightforward ASP implementation*:*

```
1: person(jones). person(paul). person(jacob).
2: has(gunman,moustache) :- reliable(jones).
3: -reliable(jones) :- reliable(paul).
4: reliable(jones) :- reliable(jacob).
5: reliable(X) :- person(X), not -reliable(X).
```

This answer set program is inconsistent. The literal `-reliable(jacob)` cannot ever be derived from the program, so `reliable(jacob)` must be in every answer set by `(5)` and `(1)`. Thus

[2] A search on December 19, 2017 in the British National Corpus of English for the phrase "not provably not" returns no entries

`reliable(jones)` must be in every answer set by `(4)`. However, the same holds for `paul`, whence the literal `reliable(paul)` must be in every answer set. Thus `-reliable(jones)` must be in every answer set by `(3)`. Consequently, any answer set would have to contain both `reliable(jones)` and `-reliable(jones)`, therefore no answer set exists. Note that while ASP per se can deal with this example with auxiliary components and by other means[3], our point is that the common "not provably not" reading of "it is usual that" phrases is not always correct. The formal semantics, implementation, and natural language are not appropriately linked.

Yet, a correctly formalized logic program ought to produce the intended interpretations as stable models. Thus, the "not provably not" reading of "it is usual that" phrases is not always correct.[4] In contrast, the correct reading is obtained by interpreting "it is usual that ⟨*statement*⟩" as a defeasible rule in a defeasible theory.

The example shows how one (manual) translation of linguistic expressions to a formal language can be used to represent arguments in natural language and for reasoning, yet the formalisation does not yield intuitively correct results. We would like an approach that makes coherent and cohesive connections between formal argumentation, the formal language, and natural language syntax and semantics, preferably using automation to facilitate the connections.

Towards a Controlled Natural Language for Argumentation. Natural language processing has, in our work, a crucial role in connection to formal argumentation and the formal language, since we wish to provide explanations and justifications in a form that humans understand for communication.

One approach to relating argumentation and natural language is found in research on argument mining, wherein arguments are automatically identified and extracted from large corpora of natural language texts [70], primarily using machine learning approaches. While there have been advances, such approaches treat the textual passages as atomic propositions, thus missing semantically meaningful, structured information that is relevant for fine-grained inference in First-order Logic. Rule-based approaches [103], which can extract arguments and some of the information within sentences, are not yet sufficiently well-developed to extract highly structured information for a KB.

In contrast, controlled natural languages (CNLs) are engineered languages with finite lexicons and fixed grammatical constructions [63]; that is, they reduce and control linguistic (lexical, syntactic, semantic, and discourse) variables found in natural language. Among the variety of purposes and applications, we focus on CNLs which provide unambiguous translations to machine readable semantic representations of First-order Logic such as Attempto Controlled English (ACE) with associated Prolog inference engine RACE [42, 43] or Processable English (PENG$^{\text{ASP}}$ [54]) with associated inference engine in answer set programming (ASP). Both ACE and PENG$^{\text{ASP}}$ provide some facility for non-monotonic reasoning using negation-as-failure, which we have touched on above. As indicated, the constraints of CNLs are useful, particularly to control ambiguity, and help to focus attention on particular phenomena. Some CNLs, e.g. ACE, have useful auxiliary functionalities such as input editors and verbalisers from semantic representations, facilitating communication with human users. As neither RACE nor PENG$^{\text{ASP}}$ are open source, we work with the ACE related open source tool AceRules [60]. While there are powerful, wide-coverage tools for parsing and semantic representation, e.g. C&C/Boxer [17], which could be used as CNLs, but would require controls over the linguistic variables (including alternatives), lack

[3] Discussing the same example, [88] incorporate meta-level model construction and Bayesian reasoning, yet do not relate natural language defeasibility to a semantic representation.

[4] Adding an abnormality atom into the body of line 5 (like in rule (12) of [9]) would address inconsistency, but not get us our intended reading. It would introduce the issue of having to create abnormality predicates from language input, where such predicates are not explicit.

the useful auxiliary functionalities, and analysis of output semantic representations to check they are correct [106]. A CNL takes an *engineering approach* to natural language since it is highly structured, constrained, evaluated, and can be systematically modified. On the one hand, a CNL can hypothetically be used as a "target" for homogenisation and structuring of mined information, and on the other hand, it can be used as an input tool to provide well-structured KBs for a defeasible inference engine. In this paper, we focus on the latter.

1.2. Our Contribution

Summarizing, in this work, which develops from [31, 94, 101, 105], we motivate and present a novel argumentation inspired semantics of theories consisting of defeasible rules which avoids some of the semantic limitations and computational pitfalls of alternative argumentation based accounts. Moreover, we report on initial progress to tie our formal model of argumentation to a controlled natural language allowing for expression of potentially incomplete and/or inconsistent information.

Concretely our contributions are as follows:

- We propose a formal argumentation model (direct semantics of defeasible theories) which can handle several of the important examples that have been proposed in the literature to motivate the need for argumentation (rather than e.g. classical logical) based reasoning. We refer to these as the "standard examples from the literature" from now on. The model:
  - does not require auxiliary components to facilitate defeasible reasoning that are linguistically implicit or explicit. See Section 2.1.
  - does not require regeneration of arguments when the knowledge base changes. See Section 2.1.
  - does not introduce redundant and opaque arguments or attacks. See Section 2.1.
  - satisfies the rationality postulates. See Section 2.2.
  - has at least the expressivity of propositional logic. See Section 2.3.
  - does not lead to exponential growth of argumentation frameworks. See Section 2.3, Section 2.4, and Section 2.6.
  - has attractive complexity properties. See Section 2.4
  - formalisation suits natural language expressions. See Section 1.1, Section 2.5, and Section 3.
  - has an available implementation. See Section 2.6.
  - treats partial knowledge bases. See Section 2.7.
  - expresses various senses of argument. See Section 2.8.
- We describe steps undertaken to implement a CNL interface to the formal argumentation model described in Section 2. This is the content of Section 3. Concretely, for our CNL interface we build on the controlled natural language ACE [43], extending it with a natural construct for expressing defeasibility: "it is usual that". The interface:
  - uses the reasoning engine for (non-defeasible) logical rules expressed in ACE, AceRules [60], as a starting point for our own implementation. We give a brief overview of the internal mechanisms of AceRules which is, in large part a product of reverse-engineering, in Section 3.1. In Section 3.2, we then describe a prototype that relies heavily on AceRules and which we have reported on in [31]. At the back-end, this system also relies on the (more or less) static encodings of defeasible rules to disjunctive answer set programming (ASP) referred to in Section 2.6. This system is able to deal with several important standard examples from the literature when encoded in (extended, yet restricted) ACE.

– details, in Section 3.3, some limitations that we encountered with the previous approach and then outline a revised implementation, which builds on but is independent of AceRules, using dynamic rather than static encodings to ASP. In particular, the revised implementation allows for a form of existential quantification as well as "wide literals" in the body and head of rules. It thus provides an implementation of an extension of the rule language presented in Section 2. We nevertheless translate this extension into the language from the latter section. The extension of the language proposed in Section 2 and its implementation bares resemblance to work on $\exists$-ASP [45].
– provides an extended example in the context of a potential use case of our CNL interface in Section 3.4. The use case in question is an extension of AceWiki; a version of the popular online encyclopedia Wikipedia, but where articles are written using ACE rather than unrestricted natural language.

Putting the above together, we have a first step in the development of an end-to-end argumentation system which we have dubbed *EMIL* (acronym for "Extracting Meaning from Iconsistent Language"). The system has adequate semantic and computational foundations, and is able to make sense out of potentially inconsistent information expressed in a restricted subset of natural language. We have motivated a potential use case for our system in some detail.

Beyond our contributions on different aspects of the argumentation pipeline, our work brings together developments in the areas of formal argumentation models, systems for formal argumentation, and work on controlled natural languages. In particular, tying in our model of argumentation to an adequate CNL serves as proof of the concrete (vs. formal) expressive power of the formal model and, via an implementation, allows for experimentation (e.g. in the context of a use case or application). We have thus laid the groundwork for further development and/or revision of *EMIL*, either from the perspective of expressiveness, computation, the CNL, and/or the connections between the different components of the pipeline.

In Section 2, we present the direct semantic approach in detail. In Section 3, we first discuss how we have adapted a CNL to work with defeasible rules, and then we present examples and the inferences we draw. Finally, Section 4 closes with some general observations and future work. Thus, overall, the paper makes a novel contribution to research that communicates explanations/justifications for nonmonotonic knowledge bases, threading a path from natural language expressions, to a logical representation of them, and hence to a computable rule language for the logical representation.

2. The Semantics of Defeasible Theories

In this section, we discuss our proposal for the semantics of defeasible theories, which we dub *Direct Semantics*. It is a semantics in the sense that it provides a formal account of the meaning of defeasible theories; from among all possible “readings” of a given theory (sets of literals), it selects those that fit certain quality criteria. It is not a “new” semantics in the sense of e.g. *stage*, *ideal*, and so on of argumentation frameworks [10], as we do not deal with argumentation frameworks but rather with the theory directly. There are a range of subsidiary presentations, which we outline here. In Section 2.1, our approach to propositional defeasible theories is presented in detail. As the Rationality Postulates are taken to be required for a semantics of defeasible theories, we discuss how the postulates are satisfied in Section 2.2. The expressivity and complexity of Direct Semantics are analysed in Section 2.3 and Section 2.4. As one of our main objectives is to tie the semantics to natural language processing, which must support some aspects of first-order logic, Section 2.5 introduces defeasible theories with

variables. In Section 2.6 outline our implementation based on disjunctive answer set programming [46]. In Section 2.7, the approach is augmented with reasoning by cases to reason with incompletely specified knowledge bases. Our approach does not construct arguments as complex "objects" as in [14, 80] or others. Yet, such complex objects are clearly referred to in everyday language and useful. In Section 2.8, we show how such we can construct different sorts of argument objects on top of Direct Semantics. As pointed out at the onset, we can abstract from the specific manifestations of our approach as in Section 2.9. Finally, in Section 2.10, we compare our approach with a range of *desiderata* for theories of argumentation, highlighting how our approach addresses them along with a comparison to other possible alternatives.

2.1. Propositional Defeasible Theories

For a set $\mathcal{P}$ of atomic propositions, the set $\mathcal{L}_\mathcal{P}$ of its literals is $\mathcal{L}_\mathcal{P} = \mathcal{P} \cup \{\neg p \mid p \in \mathcal{P}\}$. A *rule* over $\mathcal{L}_\mathcal{P}$ is a pair (B,h) where the finite set $B \subseteq \mathcal{L}_\mathcal{P}$ is called the *body* (premises) and the literal $h \in \mathcal{L}_\mathcal{P}$ is called the *head* (conclusion). For $B = \{b_1,\ldots,b_k\}$ with $k \in \mathbb{N}$, we sometimes write rules in a different way: a *strict* rule is of the form "$b_1,\ldots,b_k \to h$"; a *defeasible* rule is of the form "$b_1,\ldots,b_k \Rightarrow h$". In case $k = 0$ we call "$\to h$" a *fact* and "$\Rightarrow h$" an *assumption*.

The intuitive meaning of a rule (B,h) is that whenever we are in a world (that is, a consistent set of literals) where all literals in B hold (are contained in the world), then also literal h holds. Given a world L, a rule (B,h) is *applicable* if $B \subseteq L$ and *inapplicable* otherwise. We say that a rule (B,h) *holds* for a set L of literals if $B \subseteq L$ implies $h \in L$. (Put another way, (B,h) holds for L iff $B \cup \{h\} \subseteq L$ or $B \not\subseteq L$.) So a rule makes a statement with respect to a world, and yet it can hold for one world but possibly not so for another. For example, the rule $(\{a\},b)$ holds in the worlds $\{a,b\}$ and $\emptyset$ but not in $\{a\}$ or $\{a,\neg b\}$. In particular, a rule $(\{a\},b)$ is not equivalent to its contrapositive $(\{\neg b\},\neg a)$, as the former holds in the world $\{\neg b\}$ but the latter does not. Thus rules are not to be confused with material implication in propositional logic.

The difference between strict and defeasible rules is the following: a strict rule holds in *all* possible worlds where it is applicable; a defeasible rule holds in *most* possible worlds where it is applicable, i.e. those worlds that are 'usually' or 'normatively' the case. Thus, there are some non-normative worlds such that for defeasible rule (B,h), $B \subseteq L$ and $h \notin L$. On the other hand, a world where some strict rule is applicable, yet does not hold, is impossible.

A *defeasible theory* is a tuple $\mathcal{T} = (\mathcal{P},\mathcal{S},\mathcal{D})$ where $\mathcal{P}$ is a set of atomic propositions, $\mathcal{S}$ is a set of strict rules over $\mathcal{L}_\mathcal{P}$ and $\mathcal{D}$ is a set of defeasible rules over $\mathcal{L}_\mathcal{P}$. For the purposes of this paper, we restrict ourselves to finite vocabularies $\mathcal{P}$ and thus to finite defeasible theories.[5]

The meaning of defeasible theories is defined as follows. To define the meta-level negation of literals, we set $\overline{p} = \neg p$ and $\overline{\neg p} = p$ for every $p \in P$. A set $L \subseteq \mathcal{L}_\mathcal{P}$ of literals is *consistent* iff for all $z \in \mathcal{L}_\mathcal{P}$ we find that $z \in L$ implies $\overline{z} \notin L$. For a set $R \subseteq \mathcal{S} \cup \mathcal{D}$ of rules and a set $L \subseteq \mathcal{L}_\mathcal{P}$ of literals, we define $R(L) = \{h \in \mathcal{L}_\mathcal{P} \mid (B,h) \in R, B \subseteq L\}$; a set L of literals is *closed under* R iff $R(L) \subseteq L$.

We next present the first part of our direct semantics. The main underlying intuition goes back to foundational work on the treatment of inconsistency by Rescher and Manor [85], and to work on defeasible logical reasoning by Poole [79].

[5] For finite $\mathcal{P}$, the set $2^\mathcal{P}$ of all subsets of $\mathcal{P}$ is finite and thus the set $2^\mathcal{P} \times \mathcal{P}$ of all possible rules is finite. While it may be worthwhile from a theoretical viewpoint to analyse infinite defeasible theories, such is irrelevant to our intended application as a back-end in reasoning about defeasibility in a controlled natural language.

Definition 1. *Let $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ be a defeasible theory. A set $M \subseteq \mathcal{L}_\mathcal{P}$ of literals is a* possible set *for $\mathcal{T}$ if and only if there exists a set $\mathcal{D}_M \subseteq \mathcal{D}$ such that:*

(1) *M is consistent;*
(2) *M is closed under $\mathcal{S} \cup \mathcal{D}_M$;*
(3) *$\mathcal{D}_M$ is $\subseteq$-maximal with respect to items 1 and 2.*

Intuitively, a possible set of literals is consistent, closed under strict rules, and maximally consistent with respect to the applicability of defeasible rules. It follows that each possible set M induces a set $\mathcal{D}_M$ of defeasible rules that hold in M.

Not every defeasible theory has possible sets:

Example 3. *The theory $(\{a\}, \{\rightarrow a, \quad a \rightarrow \neg a\}, \emptyset)$ does not have a possible set: for any candidate L, closure would yield $a \in L$ and thus $\neg a \in L$, thereby violating consistency.*

Regarding the "usually, if B then h" reading of a defeasible rule (B, h), the maximality condition in Definition 1 ensures that possible sets are as "usual" as possible (with respect to the given rules). But in a possible set, there might still be cyclic or otherwise unjustified conclusions.

Example 4. *Consider $\mathcal{T} = (\{a, b\}, \emptyset, \{a \Rightarrow b, \quad b \Rightarrow a\})$, a simple defeasible theory with seven possible sets, $M_1 = \emptyset$, $M_2 = \{\neg a\}$, $M_3 = \{\neg b\}$, $M_4 = \{\neg a, \neg b\}$, $M_5 = \{a, \neg b\}$, $M_6 = \{\neg a, b\}$, $M_7 = \{a, b\}$. Almost all of the possible sets (except $M_1 = \emptyset$) contain unjustified conclusions. For example in M_2, literal $\neg a$ is just there although there is no rule support for it. Likewise, in M_7, literal a holds because b does and vice versa. In some contexts, e.g. causal reasoning [30], a model like M_7 is unintended as there is no "outside" support (no causal reasons) for any of a, b.*

Below, we further refine our direct semantics to rule out interpretations where some literals cannot be justified. We start with the notion of a derivation, which is basically a minimal proof of a literal using only modus ponens over rules.

Definition 2. *Let $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ be a defeasible theory. A* derivation in $\mathcal{T}$ *is a set $R \subseteq \mathcal{S} \cup \mathcal{D}$ of rules with a partial order $\preccurlyeq$ on R (its associated strict partial order denoted $\prec$) such that:*

(1) *$\preccurlyeq$ has a greatest element $(B_z, z) \in R$;*
(2) *for each rule $(B, h) \in R$, we have: for each $y \in B$, there is a rule $(B_y, y) \in R$ with $(B_y, y) \prec (B, h)$;*
(3) *R is $\subseteq$-minimal with respect to items 1 and 2.*

Intuitively, a derivation always concludes some specific unique literal z via a rule (B_z, z), and then in turn contains derivations for all $y \in B_z$ needed to derive z, and so on, down to facts and assumptions. Minimality ensures that there are no spurious rules that are not actually needed to derive z. The partial order $\preccurlyeq$ guarantees that derivations are acyclic. For the above, we say that *R is a derivation for z.*

Example 5. *Consider the defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ with $\mathcal{P} = \{a, b, c\}$, strict rules $\mathcal{S} = \{\rightarrow a, \quad a, b \rightarrow c\}$, and defeasible rules $\mathcal{D} = \{\Rightarrow b, \quad a \Rightarrow c\}$. There are two distinct derivations for the literal c, where the order of presentation reflects the ordering $\prec$ on the rules:*

$$d_1 = \{(\emptyset, a), (\emptyset, b), (\{a, b\}, c)\} \mathrel{\hat{=}} \{\rightarrow a, \quad \Rightarrow b, \quad a, b \rightarrow c\}$$
$$d_2 = \{(\emptyset, a), (\{a\}, c)\} \mathrel{\hat{=}} \{\rightarrow a, \quad a \Rightarrow c\}$$

Now we refine the direct semantics such that only literal sets with derivations for all its elements are considered.

Definition 3. *Let $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ be a defeasible theory and $M \subseteq \mathcal{L}_\mathcal{P}$ be a possible set for $\mathcal{T}$. M is a* stable set for $\mathcal{T}$ *iff for every $z \in M$ there is a derivation of z in $(\mathcal{P}, \mathcal{S}, \mathcal{D}_M)$.*

Thus stable sets are possible sets where all contained literals are grounded in facts and assumptions. It does not matter which of the two – there is no ontological distinction between defeasible and strict rules on the level of a single stable set. Intuitively, a stable set is a coherent, *justified* set of beliefs in which the world is as normal as possible. Each stable set M is uniquely characterised by a set $\mathcal{D}_M$ of applied defeasible rules; we will sometimes make use of that fact in this paper.

Having defined our main semantics, we now analyse some of its properties.

2.2. Rationality Postulates

It is immediate from Definition 1 (possible sets) that the semantics satisfies the rationality postulates closure and direct consistency [25] simply because they are built into the definition.

Proposition 1. *Let $\mathcal{T}$ be a defeasible theory. All possible sets of $\mathcal{T}$ are consistent and closed under strict rules.*

The satisfaction of indirect consistency, and the same properties for stable sets then follow as easy corollaries. Given that these postulates are considered to be the gold standard of conclusions from defeasible theories, we conclude that our direct semantics provides intuitive, rational answers.

In essence, we could also interpret our direct semantics as adding two new postulates to consistency and (strict) closure: (1. Groundedness) For every conclusion there must be a derivation. (This is implicit in approaches with structured arguments.) (2. Defeasible Closure) There must be a maximal (subject to consistency and strict closure) set $\mathcal{D}_M$ of defeasible rules such that the set of conclusions is closed under $\mathcal{D}_M$.

2.3. Formal Expressiveness

With regard to the measure of being able to express sets of two-valued interpretations [51], it is quite clear that our approach is as expressive as propositional logic. Consider a propositional formula φ over a propositional vocabulary P. Clearly φ can be transformed into an equivalent formula ψ in conjunctive normal form, that is, of the form $\psi = \psi_1 \wedge \ldots \wedge \psi_n$ where each ψ_i is a disjunction of literals.We create a defeasible theory $\mathcal{T}_\varphi = (P, \mathcal{S}_\varphi, \mathcal{D}_P)$ as follows: the defeasible rules are $\mathcal{D}_P = \{\Rightarrow p, \Rightarrow \neg p \mid p \in P\}$; for each conjunct $\psi_i = \psi_i^1 \vee \ldots \vee \psi_i^{m_i}$ of ψ, the set $\mathcal{S}_\varphi$ contains the strict rules $\overline{\psi_i^2}, \overline{\psi_i^3}, \ldots, \overline{\psi_i^{m_i}} \to \psi_i^1$, $\overline{\psi_i^1}, \overline{\psi_i^3}, \ldots, \overline{\psi_i^{m_i}} \to \psi_i^2, \ldots, \overline{\psi_i^1}, \overline{\psi_i^2}, \ldots, \overline{\psi_i^{m_i-1}} \to \psi_i^{m_i}$. (Intuitively, these rules correspond to all transpositions of the disjunction ψ_i.)

Proposition 2. *For any propositional formula φ, the stable sets of $\mathcal{T}_\varphi$ correspond one-to-one with the models of φ.*

Proof. By definition of $\mathcal{D}_P$, each stable set S of $\mathcal{T}_\varphi$ is a maximally consistent set of literals, that is, for each $p \in P$ we have either $p \in S$ or $\neg p \in S$. For convenience we assume that $\varphi = \psi$ is in CNF and all clauses ψ_i are non-empty. Let $M \subseteq P$. We show that $M \models \varphi$ iff $L_M = M \cup \{\neg p \mid p \in P \setminus M\}$ is a stable set for $\mathcal{T}_\varphi$.

$\Longrightarrow$: Let $M \models \varphi$. Since L_M is maximally consistent and its $\mathcal{D}_M$ is uniquely determined, it remains to show that L_M is closed under $\mathcal{S}_\varphi$. Let $(B,h) \in \mathcal{S}_\varphi$. By definition, this rule originates from a clause of ψ, say ψ_i. Now since $M \models \psi$, we have in particular that $M \models \psi_i$. That is, there is a $1 \leqslant j \leqslant m_i$ such that $M \models \psi_i^j$. There are two cases:

(1) $\psi_i^j = h$. Then by definition and $M \models \psi_i^j$, we get $h = \psi_i^j \in L_M$ whence (B,h) holds in L_M.
(2) $\overline{\psi_i^j} \in B$. Then since $\psi_i^j \in L_M$ and L_M is consistent we have $B \not\subseteq L_M$ whence (B,h) holds in L_M.

In any case, L_M is closed under $\mathcal{S}_\varphi$.

$\Longleftarrow$: Let L_M be a stable set for $\mathcal{T}_\varphi$. We have to show that for each clause ψ_i of ψ, we have $M \models \psi_i$. Let $\psi_i = \psi_i^1 \vee \ldots \vee \psi_i^{m_i}$ be a clause of ψ. Assume to the contrary that $M \models \psi_i$, that is, for all $1 \leqslant j \leqslant m_i$ we find $M \not\models \psi_i^j$, that is, $\overline{\psi_i^j} \in L_M$. Then by definition the rule $\left(\left\{\overline{\psi_i^1}, \ldots, \overline{\psi_i^{m_i-1}}\right\}, \psi_i^{m_i}\right) \in \mathcal{S}_\varphi$ is applicable to L_M. Since L_M is a stable set for $\mathcal{T}_\varphi$, it is in particular closed under strict rules, whence $\psi_i^{m_i} \in L_M$. Thus we have $\psi_i^{m_i} \in L_M$ and $\overline{\psi_i^{m_i}} \in L_M$ and L_M is inconsistent. Contradiction. Thus $M \models \psi_i$. Since ψ_i was chosen arbitrarily, we have $M \models \psi$. □

2.4. Computational Complexity

We first analyse the most important decision problems associated with our direct semantics, namely stable set verification, stable set existence, and credulous and sceptical reasoning.

Proposition 3. (1) *The problem "given a defeasible theory $\mathcal{T}$ and a set $M \subseteq \mathcal{L}_\mathcal{P}$ of literals, decide whether M is a stable set of $\mathcal{T}$" is* coNP*-complete.*
(2) *The problem "given a defeasible theory $\mathcal{T}$, decide whether it has a stable set" is Σ_2^P-complete.*
(3) *The problem "given a defeasible theory $\mathcal{T}$ and a literal $z \in \mathcal{L}_\mathcal{P}$, decide whether z is contained in some stable set of $\mathcal{T}$" is Σ_2^P-complete.*
(4) *The problem "given a defeasible theory $\mathcal{T}$ and a literal $z \in \mathcal{L}_\mathcal{P}$, decide whether z is contained in all stable sets of $\mathcal{T}$" is Π_2^P-complete.*

Proof. (1) in coNP: Consistency, closure, and existence of derivations can be verified in deterministic polynomial time. For derivations, this works as follows: We first identify the set $R_M \subseteq \mathcal{S} \cup \mathcal{D}$ of rules (B,h) where both $B \subseteq M$ and $h \in M$. Then we construct an AND-OR graph where each rule $r \in R_M$ is an AND-node, each literal $z \in M$ is an OR-node. There is an edge from (B,h) to z iff $z = h$; there is an edge from z to (B,h) iff $z \in B$. Towards obtaining derivations for all $z \in M$, the resulting graph is labelled as follows. For each rule $(\emptyset,h)$, we label its rule node with 0. Then we iterate the following: (1) For each unlabelled literal node z with a direct rule node predecessor labelled i, we label the literal node z with $i+1$. (2) For each unlabelled rule node (B,h) where for each literal $b \in B$, the literal node b is already labelled with at most j, we label the rule node (B,h) with $j+1$. Since there are only finitely many nodes,

this process stops eventually. Derivations can be obtained from the labelled graph as follows. Assume that we want to construct a derivation for literal $z \in M$, and that its literal node z is labelled i. Amongst the predecessors of z, all of them rule nodes, we choose (lexicographic tie-breaking if there are multiple candidates) one r with the smallest possible label $j \leqslant i$. This tells us that r can be used to derive z; for each of the predecessor nodes of r, all literal nodes, we proceed recursively. Since labels never increase, we eventually get down to 0-labelled rule nodes, that is, facts and assumptions.

To show that M is *not* a stable set of $\mathcal{T}$, we can guess a set $M' \supseteq M$ and show closure and consistency for M'.

coNP-hard: We reduce from the problem of verifying whether a given set of arguments is a preferred extension for an AF $F = (A, R)$ [32, 38]. Given F, define the following defeasible theory $\mathcal{T}_F = (\mathcal{P}, \mathcal{S}, \mathcal{D})$:

$$\begin{aligned}\mathcal{P} &= A\\ \mathcal{S} &= \{b \to \neg a \mid (a,b) \in R\} \cup\\ &\qquad \left\{\neg a_1, \ldots, \neg a_n \to b \;\mid\; b \in A, \{a_1, \ldots, a_n\} = R^{-1}(b)\right\}\\ \mathcal{D} &= \{\Rightarrow a \mid a \in A\}\end{aligned}$$

Intuitively, the strict rules in $\mathcal{S}$ verify that a given three-valued interpretation (represented by a set of literals) is a fixpoint of the characteristic operator of the AF, that is, that an argument is true in the set E iff all its attackers are false in the set. The defeasible rules serve to model the maximisation of true arguments, as preferred semantics can be rephrased as information-maximal complete semantics.

We now show that any $E \subseteq A$ is a preferred extension of F if and only if the set $\tilde{E} = E \cup \{\neg b \mid \exists a \in E : (a,b) \in R\}$ is a stable set for $\mathcal{T}_F$.

$\Longrightarrow$: Let $E \subseteq A$ be preferred for F. In particular, E is admissible and thus conflict-free, so E is consistent and also $\tilde{E}$ is consistent. We next show that $\tilde{E}$ is closed under the strict rules $\mathcal{S}$. Consider any literal $z \in \mathcal{S}(\tilde{E})$.

(a) $z = \neg a$ for some $a \in A$. Then there is a $b \in E$ with $(a,b) \in R$ due to the definition of $\mathcal{S}$. Since E is admissible, there is a $c \in E$ with $(c,a) \in R$. By definition of $\tilde{E}$, we find $\neg a \in \tilde{E}$.

(b) $z = a \in A$. Then all attackers of a occur as negative literals in $\tilde{E}$. Since E is in particular complete, we have $a \in E \subseteq \tilde{E}$.

It remains to show that $\tilde{E}$ is $\subseteq$-maximal. This follows from the fact that E is $\subseteq$-maximal and the strict rules only enforce completeness.

$\Longleftarrow$: Let $E \subseteq A$ be such that $\tilde{E}$ is a stable set for $\mathcal{T}_F$. We show that E is preferred for F, that is, complete and $\subseteq$-maximal with respect to being complete.

complete: We show that for all $a \in A$, we have: $a \in E$ iff all attackers of a are attacked by E. By definition of $\tilde{E}$, this can be reformulated as: $a \in E$ iff for all $b \in R^{-1}(a)$, $\neg b \in \tilde{E}$.

if: Let $a \in A$ be such that for all $b \in R^{-1}(a)$, we find $\neg b \in \tilde{E}$. Then by definition of $\mathcal{S}$ we get $a \in \tilde{E}$ whence $a \in E$.

only if: Let $a \in E$. Then clearly $a \in \tilde{E}$ and by definition of $\mathcal{S}$, for each attacker b of a we find $\neg b \in \mathcal{S}(\tilde{E})$. Since $\tilde{E}$ is a stable set for $\mathcal{T}_F$, we have $\neg b \in \mathcal{S}(\tilde{E}) \subseteq \tilde{E}$.

$\subseteq$-maximal: This follows from the fact that $\tilde{E}$ is $\subseteq$-maximal and the strict rules only enforce completeness.

(2) in Σ_2^P: Given a defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$, we can guess a set $M \subseteq \mathcal{L}_\mathcal{P}$ and verify in coNP that it is a stable set for $\mathcal{T}$.

Σ_2^P-hard: We consider the problem of sceptical reasoning under preferred semantics in AFs and recall that it is Π_2^P-complete [38]. We provide a reduction from the co-problem, that is, "given an AF $F = (A, R)$ and an $a \in A$, is there some preferred extension $E \subseteq A$ with $a \notin E$?". In the first part of the proof, we have seen a construction that assigns to each AF F a defeasible theory $\mathcal{T}_F$ such that the preferred extensions of F correspond one-to-one to stable sets of $\mathcal{T}_F$. We now extend this construction such that for the argument $a \in A$ whose sceptical acceptance is to be tested, we add the strict rules $\mathcal{S}_a = \{\rightarrow z, a \rightarrow \neg z\}$ making use of the additional atom z. Thus for a given instance of sceptical preferred acceptance consisting of $F = (A, R)$ (with associated defeasible theory $\mathcal{T}_F = (\mathcal{P}, \mathcal{S}, \mathcal{D})$) and $a \in A$, we create the defeasible theory $\mathcal{T}_{F,a} = (\mathcal{P} \cup \{z\}, \mathcal{S} \cup \mathcal{S}_a, \mathcal{D})$. We claim that this defeasible theory has a stable set iff F has a preferred extension E with $a \notin E$. To see why this is so, consider any preferred extension E of F. If $a \notin E$ then by the correspondence result above the set $\tilde{E} \cup \{z\}$ would be a stable set for $\mathcal{T}_{F,a}$. If, on the other hand, $a \in E$ then by correspondence the argument a would also be contained in the corresponding stable set of $\mathcal{T}_{F,a}$ (if there were one). However, due to the rules concluding z (from the fact) and $\neg z$ (using a and the second rule in $\mathcal{S}_a$), any preferred extension containing a cannot lead to a stable set of $\mathcal{T}_{F,a}$. Put another way, only preferred extensions E for F with $a \notin E$ survive as stable sets of $\mathcal{T}_{F,a}$, which is exactly what we need to obtain the desired reduction.

(3) in Σ_2^P: Given a defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ and a literal $z \in \mathcal{L}_\mathcal{P}$, we can guess a set $M \subseteq \mathcal{L}_\mathcal{P}$ with $z \in M$ and verify in coNP that it is a stable set for $\mathcal{T}$.

Σ_2^P-hard: We reduce from the stable set existence problem: Given $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$, we construct $\mathcal{T}' = (\mathcal{P} \cup \{z\}, \mathcal{S} \cup \{\rightarrow z\}, \mathcal{D})$ for $z \notin \mathcal{P}$. It is clear that $\mathcal{T}'$ has a stable set M with $z \in M$ iff $\mathcal{T}$ has a stable set.

(4) in Π_2^P: Given a defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ and a literal $z \in \mathcal{L}_\mathcal{P}$, in order to verify that z is *not* sceptically entailed by $\mathcal{T}$, we can guess a set $M \subseteq \mathcal{L}_\mathcal{P}$ with $z \notin M$ and verify in coNP that it is a stable set for $\mathcal{T}$.

Π_2^P-hard: We reduce stable set existence to the complement of this problem: Given $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$, we construct $\mathcal{T}'' = (\mathcal{P} \cup \{z\}, \mathcal{S}, \mathcal{D})$ for $z \notin \mathcal{P}$. Clearly $\mathcal{T}$ has a stable set iff $\mathcal{T}''$ has a stable set iff $\mathcal{T}''$ has a stable set M with $z \notin M$. □

2.5. Defeasible Theories with Variables

Having seen a language for defeasible reasoning and analysed some of its formal properties, in this section we add a limited set of first-order features that bring defeasible theories closer to natural language and thereby eases specification of input theories. More precisely, we will add first-order predicates, variables and constants to the language. This will enable us to express properties of and relationships between objects, make repeated references to objects, and provide a limited form of universal quantification. The resulting language of defeasible rules follows standard logical (Herbrand-style) approaches

and will still be essentially propositional [68, 87]; thus, it is still effectively decidable by the same bounds established earlier.

Let $\mathcal{V} = \{x_0, x_1, x_2, \ldots\}$ be a countable set of first-order variables and $\mathcal{C}$ be a finite set of constants (null-ary function symbols). For a finite first-order predicate signature $\Pi = \{p_1/k_1, \ldots, p_n/k_n\}$ (where p_i/k_i denotes that p_i is a predicate with arity k_i), the set of all atoms over Π, $\mathcal{V}$ and $\mathcal{C}$ is $\mathit{atoms}(\Pi, \mathcal{V} \cup \mathcal{C}) = \{p(t_1, \ldots, t_k) \mid p/k \in \Pi \text{ and } t_1, \ldots, t_k \in \mathcal{V} \cup \mathcal{C}\}$. A *defeasible theory with variables* is of the form $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ where $\mathcal{P} \subseteq \mathit{atoms}(\Pi, \mathcal{V} \cup \mathcal{C})$ and (as usual) $\mathcal{S}$ and $\mathcal{D}$ are sets of (strict and defeasible, respectively) rules over literals $\mathcal{L}_{\mathcal{P}}$. In particular, rules can mention variables.

The semantics of defeasible theories with variables is defined via ground instantiation. A *ground substitution* is a function $\gamma : \mathcal{V} \to \mathcal{C}$. Applying a ground substitution γ to a rule works via its homomorphic continuation $\tilde{\gamma}$: $\tilde{\gamma}((B,h)) = (\{\tilde{\gamma}(b) \mid b \in B\}, \tilde{\gamma}(h))$, where for $P/n \in \Pi$ we have $\tilde{\gamma}(P(t_1, \ldots, t_n)) = P(\tilde{\gamma}(t_1), \ldots, \tilde{\gamma}(t_n))$ with $\tilde{\gamma}(c) = c$ for all $c \in \mathcal{C}$ and $\tilde{\gamma}(v) = \gamma(v)$ for all $v \in \mathcal{V}$. The grounding of a defeasible theory with variables $\mathcal{T} = (\mathit{atoms}(\Pi, \mathcal{V}, \mathcal{C}), \mathcal{S}, \mathcal{D})$ has a vocabulary of all ground atoms and contains all ground instances of its rules:

$$\mathit{ground}(\mathcal{T}) = (\mathit{atoms}(\Pi, \mathcal{C}), \mathit{ground}(\mathcal{S}), \mathit{ground}(\mathcal{D}))$$
$$\mathit{ground}(\mathcal{R}) = \{\gamma(r) \mid r \in \mathcal{R}, \gamma : \mathcal{V} \to \mathcal{C}\}$$

A set $M \subseteq \mathcal{L}_{\mathit{atoms}(\Pi,\mathcal{C})}$ is a *stable set for a defeasible theory with variables* $\mathcal{T}$ iff M is a stable set of $\mathit{ground}(\mathcal{T})$.

We illustrate the language with our running example.

Example 2 (Continued)**.** *There has been a shooting involving three witnesses that contradict each other.*

Jones is a person. Paul is a person. Jacob is a person. If Jones is reliable then the gunman has a moustache. If Paul is reliable then Jones is not reliable. If Jacob is reliable then Jones is reliable. Usually, persons are reliable.

Clearly Paul and Jacob cannot both be reliable, and any semantics should be able to "choose" between the two options. The text above leads to this defeasible theory with variables:

$$\Pi = \{\mathit{person}/1, \mathit{reliable}/1, \mathit{has}/2\}$$
$$\mathcal{C} = \{\mathit{jones}, \mathit{paul}, \mathit{jacob}, \mathit{gunman}, \mathit{moustache}\}$$
$$\mathcal{T} = (\mathit{atoms}(\Pi, \mathcal{V}, \mathcal{C}), \mathcal{S}, \mathcal{D})$$
$$\begin{aligned}\mathcal{S} = \{&\to \mathit{person}(\mathit{jones}), \to \mathit{person}(\mathit{paul}), \to \mathit{person}(\mathit{jacob}),\\ &\mathit{reliable}(\mathit{jones}) \to \mathit{has}(\mathit{gunman}, \mathit{moustache}),\\ &\mathit{reliable}(\mathit{paul}) \to \neg\mathit{reliable}(\mathit{jones}),\\ &\mathit{reliable}(\mathit{jacob}) \to \mathit{reliable}(\mathit{jones})\}\end{aligned}$$
$$\mathcal{D} = \{\mathit{person}(x_1) \Rightarrow \mathit{reliable}(x_1)\}$$

This defeasible theory has two stable sets:

$$M_1 = M \cup \{\mathit{reliable}(\mathit{jacob}), \mathit{reliable}(\mathit{jones}), \mathit{has}(\mathit{gunman}, \mathit{moustache})\} \text{ and}$$

$M_2 = M \cup \{\mathit{reliable}(\mathit{paul}), \neg\mathit{reliable}(\mathit{jones})\}$, *with*

$M = \{\mathit{person}(\mathit{jones}), \mathit{person}(\mathit{paul}), \mathit{person}(\mathit{jacob})\}$.

Thus, our stable-set semantics makes an exclusive choice, either Jacob is reliable or Paul is reliable, avoiding inconsistency. In only one case we know about the moustache.

Reiter [82] introduced the following classic.

Example 6. *Consider the constants* $\mathcal{C} = \{\mathit{tweety}, \mathit{tux}\}$ *and predicates* $\Pi = \{\mathit{bird}/1, \mathit{flies}/1, \mathit{penguin}/1\}$. *Using that vocabulary, we can build the following defeasible theory:*

$$\begin{aligned}
\mathcal{P} &= \{\mathit{bird}(x_1), \mathit{flies}(x_1), \mathit{penguin}(x_1)\} \\
\mathcal{S} &= \{\mathit{penguin}(x_1) \to \neg\mathit{flies}(x_1), \quad \to \mathit{penguin}(\mathit{tux}) \\
&\qquad \mathit{penguin}(x_1) \to \mathit{bird}(x_1), \quad \to \mathit{bird}(\mathit{tweety})\} \\
\mathcal{D} &= \{\mathit{bird}(x_1) \Rightarrow \mathit{flies}(x_1)\}
\end{aligned}$$

As is, the theory has the following stable set:

$$M = \{\mathit{bird}(\mathit{tweety}), \mathit{penguin}(\mathit{tux}), \mathit{bird}(\mathit{tux}), \mathit{flies}(\mathit{tweety}), \neg\mathit{flies}(\mathit{tux})\}$$

If we now extend the defeasible theory base by "Tweety is a penguin" (a fact that was not known before and has somehow materialised in the meantime), thus set $\mathcal{S}' = \mathcal{S} \cup \{\to \mathit{penguin}(\mathit{tweety})\}$, *we get the stable set*

$$M' = \{\mathit{penguin}(\mathit{tweety}), \mathit{bird}(\mathit{tweety}), \mathit{penguin}(\mathit{tux}), \mathit{bird}(\mathit{tux}), \neg\mathit{flies}(\mathit{tweety}), \neg\mathit{flies}(\mathit{tux})\}$$

Reiter and Criscuolo [84] introduced the Nixon diamond, an instance of conflicting defeasible conclusions.

Example 7 (Nixon diamond)**.** *Consider this text:*

> *Nixon is a republican and is a quaker. Usually, a republican is not a pacifist. Usually, a quaker is a pacifist.*

The text leads to the following defeasible theory:

$$\begin{aligned}
\Pi &= \{\mathit{republican}/1, \mathit{quaker}/1, \mathit{pacifist}/1\} \\
\mathcal{C} &= \{\mathit{nixon}\} \\
\mathcal{T} &= (\mathit{atoms}(\Pi, \mathcal{V}, \mathcal{C}), \mathcal{S}, \mathcal{D}) \\
\mathcal{S} &= \{\to \mathit{republican}(\mathit{nixon}), \quad \to \mathit{quaker}(\mathit{nixon})\} \\
\mathcal{D} &= \{\mathit{republican}(x_1) \Rightarrow \neg\mathit{pacifist}(x_1), \\
&\qquad \mathit{quaker}(x_1) \Rightarrow \mathit{pacifist}(x_1)\}
\end{aligned}$$

This defeasible theory has two stable sets, where $M = \{\mathit{republican}(\mathit{nixon}), \mathit{quaker}(\mathit{nixon})\}$*:*

$$M_1 = M \cup \{\mathit{pacifist}(\mathit{nixon})\}$$
$$M_2 = M \cup \{\neg\mathit{pacifist}(\mathit{nixon})\}$$

The advantage of having rules with variables is that we could easily add another quaker, for whom we could then infer (independently of Nixon) to be a pacifist.

It is no restriction or modification of the above definitions to additionally assume the existence of a Leibniz-style interpreted equality predicate $=/2$, as it could be easily axiomatised using rules with variables that state reflexivity ($\to x_1 = x_1$), symmetry ($x_1 = x_2 \to x_2 = x_1$), and transitivity ($x_1 = x_2, x_2 = x_3 \to x_1 = x_3$). We will now also use such equality literals, where we write $\neg t_1 = t_2$ as $t_1 \neq t_2$.

Example 8 (Tandem)**.** *John, Mary, and Suzy drive in the countryside. Usually, if someone drives in the countryside then that person drives in the red car. Whenever there are three distinct persons and two persons drive in the red car, the third one does not drive in the red car.*

$$\Pi = \{\mathit{drivesIn}/2\}$$
$$\mathcal{C} = \{\mathit{john}, \mathit{mary}, \mathit{suzy}, \mathit{countryside}, \mathit{redCar}\}$$
$$\mathcal{T} = (\mathit{atoms}(\Pi, \mathcal{V}, \mathcal{C}), \mathcal{S}, \mathcal{D})$$
$$\begin{aligned}\mathcal{S} = \{&\to \mathit{drivesIn}(\mathit{john}, \mathit{countryside}),\\ &\to \mathit{drivesIn}(\mathit{mary}, \mathit{countryside}),\\ &\to \mathit{drivesIn}(\mathit{suzy}, \mathit{countryside}),\\ &\mathit{drivesIn}(x_1, \mathit{redCar}), \mathit{drivesIn}(x_2, \mathit{redCar}), x_1 \neq x_2, x_1 \neq x_3, x_2 \neq x_3 \to \neg\mathit{drivesIn}(x_3, \mathit{redCar})\}\end{aligned}$$
$$\mathcal{D} = \{\mathit{drivesIn}(x_1, \mathit{countryside}) \Rightarrow \mathit{drivesIn}(x_1, \mathit{redCar})\}$$

This defeasible theory has three stable sets:

$$M_1 = M \cup \{\mathit{drivesIn}(\mathit{john}, \mathit{redCar}), \mathit{drivesIn}(\mathit{mary}, \mathit{redCar}), \neg\mathit{drivesIn}(\mathit{suzy}, \mathit{redCar})\}$$
$$M_2 = M \cup \{\mathit{drivesIn}(\mathit{john}, \mathit{redCar}), \neg\mathit{drivesIn}(\mathit{mary}, \mathit{redCar}), \mathit{drivesIn}(\mathit{suzy}, \mathit{redCar})\}$$
$$M_3 = M \cup \{\neg\mathit{drivesIn}(\mathit{john}, \mathit{redCar}), \mathit{drivesIn}(\mathit{mary}, \mathit{redCar}), \mathit{drivesIn}(\mathit{suzy}, \mathit{redCar})\}$$
$$M = \{\mathit{drivesIn}(\mathit{john}, \mathit{countryside}), \mathit{drivesIn}(\mathit{mary}, \mathit{countryside}), \mathit{drivesIn}(\mathit{suzy}, \mathit{countryside})\}$$

As expected, as many people as possible drive in the red car, but not more.

2.6. Implementation

We implemented our semantics in disjunctive answer set programming [46], which we presume familiarity with. For specifying input defeasible theories, its constituent rules are represented by ASP terms. The binary predicates `head/2` and `body/2` declare rule heads and bodies, respectively; predicate `def/1` declares a rule to be defeasible. The implementation consists of a reasonably small encoding

of Definition 3 into ASP; the maximisation aspects are implemented using saturation techniques [41]. The encoding works such that the union of the encoding together with the specification of a defeasible theory is given to a solver, and the answer sets of the resulting logic program union correspond one-to-one to the stable sets of the defeasible theory. With another encoding added, each answer set program will also contain derivations for each literal in the corresponding stable set. The derivations then provide dialectical justifications for conclusions[6].

Example 2 (Continued)**.** *Our version of Pollock's moustache example in the language of the implementation is reproduced below. Interestingly, we can use first-order variables (such as the symbol* `X`*) directly in the implementation. For example, for the first-order rule* `head(p(X), person(X)) :- c(X)`*, the variable symbol* `X` *will be instantiated to each of the three constant symbols* `jones`*,* `paul`*, and* `jacob`*, leading to the three ground facts that those three constant symbols represent persons.*

```
c(jones). c(paul). c(jacob).

head(p(X), person(X)) :- c(X).

head(jm, moustache).
body(jm, reliable(jones)).

head(pj, neg(reliable(jones))).
body(pj, reliable(paul)).

head(jj, reliable(jones)).
body(jj, reliable(jacob)).

def(pr(X)) :- c(X).
head(pr(X), reliable(X)) :- c(X).
body(pr(X), person(X)) :- c(X).
```

Calling the answer set solver `clingo`[7] *on the above instance file* `tbex_pollock_moustache.lp` *via the command line call (instructing it to compute all answer sets using command-line argument '0')*

```
clingo 0 stable_sets.lp tbex_pollock_moustache.lp
```

yields the expected two answer sets along with justifications for all literals. Intuitively, ASP atoms of the form `h(l)` *state that the literal l holds in the respective stable set; ASP atoms of the form* `because(l,r)` *state that membership of literal l in the stable set M at hand can be justified by rule r.*

```
Answer: 1
h(person(jones)) because(person(jones),p(jones))
h(person(paul)) because(person(paul),p(paul))
h(person(jacob)) because(person(jacob),p(jacob))
h(reliable(jones)) because(reliable(jones),pr(jones))
```

[6]The implementation is available at github: `https://github.com/hstrass/defeasible-rules`

[7]See `http://www.potassco.org`.

```
h(reliable(jacob)) because(reliable(jacob),pr(jacob))
h(moustache) because(moustache,jm)
Answer: 2
h(person(jones)) because(person(jones),p(jones))
h(person(paul)) because(person(paul),p(paul))
h(person(jacob)) because(person(jacob),p(jacob))
h(reliable(paul)) because(reliable(paul),pr(paul))
h(neg(reliable(jones))) because(neg(reliable(jones)),pj)
SATISFIABLE
```

Consequently, to build a derivation for why a specific literal l holds in an answer set, one only has to recursively collect `because` *atoms for all literals in the body of rules mentioned in (previously collected)* `because` *atoms.*

Computational Experiments. We experimented with our implementation, which supports our claim to have addressed the problem of exponential blowup of instantiated argumentation. For the experiments, we extend the family $(D_n)_{n\in\mathbb{N}}$ of defeasible theories from Example 1 by the following rules: first, $p_0 \Rightarrow \neg p_1$ and $q_0 \Rightarrow \neg p_1$, and then for each $i \geqslant 1$ the rules $p_0, p_i \Rightarrow \neg p_{i+1}$ and $q_0, p_i \Rightarrow \neg p_{i+1}$. This is done in order to obtain a non-trivial semantics, as for each original D_n, there is only one stable set containing all atoms of the language; with the additional rules, each D_n has $n+1$ stable sets.

Below we plot the runtime of our implementation (employing clingo[8] as ASP solver) for computing a single stable set in terms of the theory size parameter n, with $n \in \{2, \ldots, 400\}$. Runtimes are averaged over three runs. We can see that for $n \approx 300$, computing a stable set can be done in under one second,

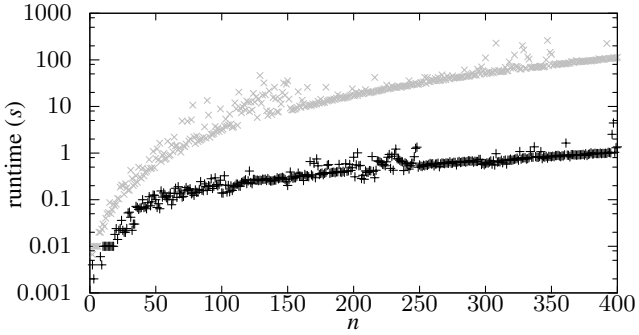

Fig. 1. Solving runtimes (log scale) in terms of theory parameter n. Black: computing a stable set; grey: additionally computing one derivation for each literal in the stable set.

with explanations in under one minute. In contrast, any approach that explicitly constructs all resulting nested arguments would have to create at least 2^{300} arguments, which greatly exceeds the number of atoms in the known universe.[9]

[8] Version 5.1.0, see `http://potassco.org` for more details.

[9] The number of atoms in our universe is estimated to be approximately 10^{80}, and clearly $2^{300} \approx 2 \cdot 10^{90} > 10^{80}$.

2.7. Reasoning by cases

By definition, stable-set semantics does not do reasoning by cases, that is, does not explicitly consider that literals might hold for unspecified reasons. Existing approaches to structured argumentation also do not reason by cases "off the shelf", although assumptions can be added for the same effect. Wyner et al. [107] have argued why and when such behaviour can be useful, for example when dealing with incompletely specified knowledge bases. For example, consider the defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ with $\mathcal{P} = \{a, b, c\}$, strict rules $\mathcal{S} = \{\rightarrow a, \quad a, b \rightarrow c\}$, and defeasible rules $\mathcal{D} = \emptyset$; we would still like to reason about c. Our possible-set semantics of Definition 1 naturally does reasoning by cases and still satisfies the rationality postulates; it thus combines the strengths of both approaches. The fact that the possible-set semantics suffers from unjustified conclusions is not a problem here, as reasoning by cases expressly and explicitly considers the possibility of literals holding for unspecified reasons.

2.8. Three Senses of "Argument"

While the direct semantics approach does not construct argument objects to compute sets of conclusions as in approaches using instantiated AFs, it is nonetheless compatible with higher-level constructions that are relevant to argumentation. Wyner et al. [107] provided an analysis of the different terminological meanings of the word "argument" and how the term is used in instantiated abstract argumentation. In their view, there are three distinct (although related) meanings of "argument": (i) a one-step reason for a claim (also called *argument* in this paper), (ii) a chain of reasoning leading towards a claim (a *case*), (iii) reasons for and against a claim (a *debate*). Wyner et al. [107] then went on to define an AF-based approach for dealing with problems that they observed to result from conflating the three senses in existing work. Although technically their approach falls short of satisfying all our needs, we nevertheless agree with their initial analysis. In what follows, we show how the three different senses of "argument" according to Wyner et al. [107] appear as distinct formal entities in the approach of this paper.

Definition 4. *Let $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ be a defeasible theory, $M \subseteq \mathcal{L}_\mathcal{P}$ be a stable set of $\mathcal{T}$ and $z \in \mathcal{L}_\mathcal{P}$ be a literal.*

- *An* argument for z from M *is a rule* $(B, z) \in \mathcal{S} \cup \mathcal{D}_M$.
- *A* case for z in M *is a derivation for z in* $(\mathcal{P}, \mathcal{S}, \mathcal{D}_M)$.
- *A* debate about z *is a pair $\langle C^+, C^- \rangle$ of sets of cases, where C^+ only contains cases for z and C^- only contains cases for $\bar{z}$, i.e., cases against z.*

Intuitively, an argument is just an atomic deduction where a single rule (B, h) of the defeasible theory is used to make the claim "h holds because all of B hold." A case involves a whole chain of reasoning (possibly involving several arguments building on top of one another) that must be grounded in facts and assumptions, and internally consistent (as witnessed by there being a stable set where the derivation applies). A debate, in turn, involves several cases that might originate from different (possibly incompatible) stable sets.

Example 2 (Continued)**.** *In Pollock's moustache example with witness testimonies about reliability, the derivation*

$$C_1 = \{\rightarrow \mathit{person}(\mathit{jones}),$$

$$person(jones) \Rightarrow reliable(jones),$$
$$reliable(jones) \Rightarrow has(gunman, moustache)\}$$

is a case for $has(gunman, moustache)$ *in* M_1*, and so is*

$$C_2 = \{\rightarrow person(jacob),$$
$$person(jacob) \Rightarrow reliable(jacob),$$
$$reliable(jacob) \rightarrow reliable(jones),$$
$$reliable(jones) \Rightarrow has(gunman, moustache)\}.$$

Both of these cases contain (sub-)derivations that are cases for $reliable(jones)$*. We can also construct a case for the opposite literal* $\neg reliable(jones)$ *in the other stable set* M_2*:*

$$C_3 = \{\rightarrow person(paul),$$
$$person(paul) \Rightarrow reliable(paul),$$
$$reliable(paul) \rightarrow \neg reliable(jones)\}$$

Taking the sub-cases of C_1 *and* C_2 *for* $reliable(jones)$ *and* C_3 *together leads to a debate about* $reliable(jones)$*, where the "participants" argue from different views,* M_1 *and* M_2*.*

For the rule set $\mathcal{S} = \{\rightarrow p, \quad p \rightarrow p\}$, which is troublesome for approaches with nested arguments, our definitions above just yields two arguments for p, of which only one ($\rightarrow p$) leads to a case for p. For Example 1, our definition would also lead to an exponential number of derivations for each p_n; the important difference to previous approaches is that *we do not explicitly compute on them*. Derivations only become relevant *after* the semantics is computed.

The senses of argument here are related to, but different from, arguments in AF analyses of instantiated argumentation. An argument in Definition 4 is just a rule in ASPIC+, LBA, or ABA, where arguments require a deduction. A case in Definition 4 is an argument in the other approaches. A debate in Definition 4 is a Rebuttal attack in ASPIC+ and LBA.

The theory and implementation provide explanations for literals in a stable set. However, it remains to future work to generate correlated natural language explanations, which require the application of discourse connectives.

2.9. Possible further abstractions

The semantics of defeasible theories are a topic of ongoing work in argumentation theory [1, 25, 93, 107]. We have presented our own take on this issue in the preceding two sections. In this penultimate subsection, we wish to demonstrate that the remaining developments of this paper do not hinge on the previous definition of the direct semantics. In fact, it would be a minor adjustment to abstracting away from any concrete manifestations of existing approaches.

For that, we would only have to make a few (mild) assumptions about the approach to assigning semantics to defeasible theories that is used to draw inferences (the "back-end"). More specifically, we would assume that the reasoning back-end:

(1) accepts a defeasible theory $\mathcal{T} = (\mathcal{P}, \mathcal{S}, \mathcal{D})$ as input. An additional step might be needed to transform $\mathcal{T}$ into the reasoner's native input format, a point we raise again later.
(2) can produce "interpretations" (consistent viewpoints, e.g. extensions) and/or (sets of) credulous/sceptical conclusions of the defeasible theory with respect to one or more semantics, e.g. stable, complete, preferred, grounded [34]. We comment further on this below.
(3) can produce graph-based justifications for its conclusions as a derivation of that literal as obtained from an argument extension.

It may be more or less straightforward to lift these restrictions, depending on the concrete approaches. Our assumptions cover considerable common ground of the various approaches in the literature; they are a meaningful and non-trivial starting point for our contributions in the remainder of the paper. While there are several roles for argumentation, for our purposes, it serves to provide graph-based justifications for conclusions, which contrasts to other approaches.

While our CNL interface in Section 3 makes use of the direct semantics, but it could be used with other concrete approaches with the common ground above. However, we use our semantics since we have an implementation for it.

2.10. Related work

In this section, we discuss related work. We list desiderata (with abbreviations) which we take to be appropriate for an approach to argumentation that is useful for human-oriented explanation and justification. The desiderata are referenced to our work presented in the previous sections. While we may comment on whether or not related works address desiderata, we leave a systematic study for a future report.

AC: does not require auxiliary components to facilitate defeasible reasoning that are linguistically implicit or explicit. See Section 2.1.
AI: has an available implementation. See Section 2.6.
CP: has attractive complexity properties. See Section 2.4
EP: does not lead to exponential growth of argumentation frameworks. See Section 2.3, Section 2.4, and Section 2.6.
NL: formalisation suits natural language expressions. See Section 1.1, Section 2.5, and Section 3.
PE: has at least the expressivity of propositional logic. See Section 2.3.
PK: treats partial knowledge bases. See Section 2.7.
RA: does not require regeneration of arguments when the knowledge base changes. See Section 2.1.
RO: does not introduce redundant and opaque arguments or attacks. See Section 2.1.
RP: satisfies the rationality postulates. See Section 2.2.
SA: expresses various senses of argument. See Section 2.8.

We do not claim the definition of our direct semantics to be a major, groundbreaking novelty; however, it is straightforward and simple, provides intuitive answers (as per the rationality postulates), and has attractive complexity properties in contrast to structured argument analyses. In what follows, we briefly relate our approach to a selection of prominent alternatives. While we indicate some of the contrasts or limitations of these approaches, we have not argued that they cannot be optimised, which would make them computationally feasible. However, while we have proven that the direct semantics is computationally attractive, optimisations remain a future challenge for the proponents of the other theories.

In general, overgeneration of instantiated "argument" objects is a problem in all approaches that create explicit instances of Dung's abstract AFs with the aim of computing on them. Of course, some method may be to block or filter overgeneration; however, it is clearly worthwhile to avoid overgeneration in the first place.

Logic-based Argumentation. Besnard and Hunter [14] proposed an approach for reasoning from maximal consistent subsets of given logical formulas. In their approach, there is no distinction between strictness and defeasibility – all elements of the given knowledge base are considered defeasible, as they can be left out if needed to restore consistency. The criticism of exponential blowup applies directly: for a propositional language over n atoms $\{p_1, \ldots, p_n\}$, the knowledge base of all literals $\{p_1, \neg p_1, \ldots, p_n, \neg p_n\}$ immediately leads to 2^n arguments, as each two-valued interpretation of the atoms corresponds to a maximally consistent subset of the knowledge base. Furthermore, the approach offers no rule syntax, it is therefore not straightforward to create explanations for conclusions.

ASPIC. Although the ASPIC definitions slightly pre-date the work of Caminada and Amgoud [25], we use their definitions. As previously demonstrated, ASPIC suffers from computational problems, foremost the unrestricted blowup in the step from knowledge bases (defeasible theories) to argumentation frameworks. Aside from that, ASPIC also exhibits a certain opacity of attacks, as it is (due to the nesting of "arguments") not directly clear why a specific abstract "argument" is attacked – the attack could be due to its own top-level conclusion, or the conclusion of some sub-argument.

More fundamentally, ASPIC seems to follow an altogether different intuition on what defeasible theories mean (their informal semantics). The only candidate for comparison is stable AF semantics (as all other semantics guarantee existence of extensions), so let us look at a small, concrete example. Consider the defeasible theory with $\mathcal{S} = \{p \to \neg p\}$ and $\mathcal{D} = \{\Rightarrow p\}$. In our approach, the only stable set is $\emptyset$, where no rule has been applied (application of the only defeasible rule would make the only strict rule applicable and thus lead to inconsistency). In ASPIC, there are two defeasible arguments, $A_1 = [\Rightarrow p]$ and $A_2 = [[\Rightarrow p] \to \neg p]$. The AF that results has attacks from A_2 to itself and from A_2 to A_1 (using the attack definition of restrictive rebut).[10] Thus the derived AF has no stable extension, a behaviour that is different from our direct semantics.[11]

In ASPIC+ [73, 80], there are preferences between rules, but in our setting this would not immediately be of use, because this is not a phenomenon that typically occurs directly expressed in natural language. Rather, such priorities are implicitly given from background knowledge about specificity, recency, or other criteria not expressly in the input text.[12]

Assumption-based Argumentation. In assumption-based argumentation [16], which uses tree-shaped arguments, the same exponential overgeneration can be observed. In recent work, Craven and Toni [26] addressed some of the computational problems of tree-shaped arguments in ABA; however, their definition of "argument graph" still allows for exactly the above (exponentially many distinct) structures. For the work of Craven and Toni [26], this is not a substantial problem since they focus on credulous and

[10] For (unrestricted) rebut, there would also be an attack from A_1 to A_2, thus a stable extension $\{A_1\}$ with conclusion p witnessing ASPIC's non-satisfaction of the closure postulate.

[11] Moreover, the resulting AF is asymmetric, showing that the language considered in this paper does not immediately trivialize the Dung AFs obtained from applying the ASPIC definitions. Indeed, there are even easier examples for asymmetric Dung AFs obtained from defeasible theories: for example, take $\mathcal{S} = \{\to a\}$ and $\mathcal{D} = \{\Rightarrow \neg a\}$ where the ASPIC approach leads to two arguments $[\to a]$ and $[\Rightarrow \neg a]$ with the first attacking the second, but not vice versa.

[12] In law, there are interpretive heuristics which express argument preferences, e.g. *Lex Superiori*, wherein a superior court decision trumps a subordinate court decision. However, such meta-rules are not lexical expressions.

sceptical reasoning (albeit not for stable semantics). That being said, our notion of derivation is similar to what Craven and Toni [26] would call a "focussed, rule-minimal argument graph", they however work in a different formal setting.

Defeasible Logics. Nute and others proposed a range of rule-based languages for defeasible reasoning that are collectively known as defeasible logics [6, 75]. They are mostly of low reasoning complexity and provide semantics based on delicate, finely-grained proof theories for justification statuses of conclusions. Additionally, they offer "defeaters", where it is unclear what the linguistic counterparts are; furthermore, the delicate proof theory is not immediately amenable to creating justifications in natural language. In our approach, on the other hand, we have opted for a very simple rule language where justifications can be straightforwardly expressed in (controlled) natural language.

DefLog. Verheij [100] presented a simple and elegant logic – termed DefLog – for analysing the truth of prima-facie justified assumptions. The language offers two connectives, $\rightsquigarrow$ for expressing prima-facie support, and $\times$ for expressing dialectical defeat. The semantics is based on variations of the stable-semantics theme that is prevalent in the nonmonotonic reasoning literature.[13] As such, there are no elements of maximisation, like in our approach. Still, we are unaware of a way to parse (controlled) natural language into the formal syntax of DefLog.

CR-Prolog. Another approach to avoiding inconsistency in ASP formalisations of "usually, *P*s are *Q*s" are logic programs with consistency-restoring rules [8, 47]. In that approach, the extended logic programming paradigm is extended via a new construct, so-called *consistency-restoring rules*, which are just like extended logic program rules, only with a syntactic variant of rule implication. The intended meaning of those rules is that should the regular part of the program be inconsistent (have no answer set), a minimal subset of the CR rules can be added to the program if that restores consistency. Since the consistency-restoring rules have to be *added* to the program, that approach seems somewhat orthogonal to ours, where the semantics disregards some of the specified defeasible rules to preserve consistency.

In any case, due to the richness of the language, it is not immediately clear in the general case how explanations for conclusions should look like. Even for "ordinary" extended logic programs, that is, those without consistency-restoring rules, it required dedicated research effort to allow constructing explanations for obtaining conclusions [86].

Circumscription. McCarthy [71] presented circumscription as a form of non-monotonic reasoning. Technically, it minimises the extension of a given predicate over all models of a given formula in first-order logic. Typically, it is used to minimise the occurrence of "abnormal" circumstances, which are formalised using "abnormality" predicates. (For example, an abnormal bird does not fly, and to obtain that birds normally fly, such abnormalities are assumed away unless there is reason to believe them.) While circumscription can be and has been put to used in knowledge representation and reasoning [72], we do not see an immediate linguistic counterpart of those "abnormality" predicates or other predicates to be minimised.

Others. Dung and Son [36] define "defeasible derivations" (which need not be minimal but are otherwise just like our derivations) and a (stable) extension semantics without explicit argument construction. However, the definition of the semantics still makes use of "attacks" from sets of defeasible rules to

[13] The most influential and classical version of that theme are Reiter's extension semantics for default logic [83], Gelfond and Lifschitz' stable model semantics for normal logic programs [48], and Dung's stable extension semantics for argumentation frameworks [34]. That these all address the same theme is most neatly analyzed via approximation fixpoint theory [29, 91].

single defeasible rules and thus implicitly features the exponential blowup of explicit argument construction. This means that their definition gives no indication how to implement the semantics all the while avoiding blowup. With our semantics and implementation, we have successfully addressed this aspect. In any case, the defeasible theory with $\mathcal{S} = \{p \to \neg p\}$ and $\mathcal{D} = \{\Rightarrow p\}$ witnesses that our semantics is different from theirs: similarly to what is the case in ASPIC, as the only "default" (defeasible rule) is self-"attacking" (would lead to inconsistency via a strict rule if applied), there is no stable extension in the approach of Dung and Son; this is in contrast to our direct semantics, where $\emptyset$ is a stable set. Evidently, our approach is more crash-resistant, as it is able to completely ignore the impending inconsistency posed by $p \to \neg p$ by simply not applying the triggering $\Rightarrow p$.

Amgoud and Besnard [1] have a notion that is similar to our semantics: for a stable set M, they would call $(\mathcal{P}, \mathcal{S}, \mathcal{D}_M)$ an "option" of $\mathcal{T}$. The approach of Amgoud and Nouioua [3] essentially uses the ASPIC argument construction.

3. Towards a Controlled Natural Language Interface for Direct Semantics

In this section, we report on our work on a CNL interface to defeasible rules, in particular the formal and implemented approach of Section 2. As discussed in Section 1.1, controlled natural languages (CNLs) are engineered languages with finite lexicons and fixed grammatical constructions [54, 63].

CNLs facilitate an engineered solution to argumentation in NL by providing a testing-ground for formal argumentation models. Tying a formal argumentation model to a CNL allows for evaluation of the expressive power of the formal model via concrete (rather than formal) means. The "material adequacy" of the argumentation model in particular, can be established by considering the ease with which it can be linked to a CNL and allowing the encoding of significant scenarios pertaining to argumentation. We thus have a concrete handle on the sense in which the formal argumentation model can deal with information expressed in natural language (the natural application domain of argumentation) from the start.

Moreover, CNLs that provide an interface with inference engines enable testing of different forms and combinations of transformations from natural language to the formal argumentation model. Crucially, we can scope, experimentally control, and systematically augment the CNL as needed; this allows for an incremental development cycle of the underlying argumentation model and its connection with the CNL.

While there are, in our view, clearly advantages to working with a CNL, it is important to acknowledge its limitations as well. We work with highly restricted forms of arguments in natural language, which are otherwise rich, complex, and not fully understood; that is, natural language arguments contain a range of lexical, syntactic, semantic, and discourse variables. Mined arguments are essentially propositional in form; yet, many domains require inference from fine-grained, predicate logical representations. Arguments mined could perhaps, in future work, be processed further in order to be sufficiently analysed to suit an inference engine; in other words, the output of mining could be the input of CNL processing. Furthermore, as an engineered language, there are lexical items and grammatical constructions that are not available from the source natural language - the range of linguistic variation is highly constrained, allowing systematic analysis and development. Relatedly, there are interpretive, contextual, or idiomatic matters that a CNL does not address. Users of a language must learn to work with a CNL in ways that they do not in natural languages. Despite these limitations, we believe the advantages of an engineered language outweigh them.

In light of these remarks, we build on the framework for the CNL Attempto Controlled English (ACE) [43], which includes several open-source tools including input editors and verbalisers from semantic representations, facilitating communication with human users. In addition to extensive development, ACE has been evaluated in various ways, demonstrating its robustness across the range of constructions and semantic representations [62, 64]. In terms of inference engines ACE has an associated Prolog inference engine RACE [42, 43], while Processable English (PENG^{ASP} [54]) is associated with an inference engine in answer set programming (ASP). However, ACE and PENG^{ASP} work with the expression "not provably not", which we have discussed as an inadequate treatment for the natural language expression "it is usual that". Moreover, to progress, we work with an open-source tool that we can understand and modify; neither RACE nor PENG^{ASP} are open source. Other tools are discussed in Section 1.1.

AceRules [60], on the other hand, is an open-source ACE-based CNL with allied inference engines to reason with (non-defeasible) logical rules. We take AceRules as the starting point of our own implementation, adding a natural expression for defeasibility to the ACE language which is then provided a rule representation for our implementation of argumentation-based semantics for defeasible theories.

Concretely, the contributions we report on in this section are as follows:

- In Section 3.1, we give a brief overview of the internal mechanisms of AceRules. This description is, in large part, a product of reverse-engineering the well documented source code; although some aspects of AceRules are described in [60].
- In Section 3.2, we then describe a prototype that relies heavily on AceRules and which we have reported on in [31]. At the back-end, this system also relies on the (more or less) static encodings of defeasible rules to disjunctive ASP referred to in Section 2.6.
- In Section 3.3, we detail some limitations that we encountered with the previous approach and then outline a revised implementation, which builds on but is independent of AceRules, using dynamic rather than static encodings to ASP. In particular, the revised implementation allows for a form of existential quantification as well as "wide literals" in the body and head of rules. It thus provides an implementation of an extension of the rule language presented in Section 2. The extension of our proposed language and its implementation bares resemblance to work on $\exists$-ASP [45].
- An extended example in the context of a potential use case of our system is provided in Section 3.4. The use case in question is an extension of AceWiki; a version of the popular online encyclopedia Wikipedia, but where articles are written using ACE rather than unrestricted natural language.
- Finally, we discuss issues that require further work.

3.1. AceRules and Its Transformations

In this section, we discuss AceRules, its processes, and several of the internal transformations. In subsequent sections 3.2 and 3.3, we present our modifications relative to what is given by 'standard' AceRules.

The controlled natural language ACE avoids the ambiguities of natural language by having a restricted syntax as well as associating a unique meaning to every (ACE-) text via a small set of interpretation rules [43]. ACE supports, among others, active and passive voice, singular and plural noun phrases, relative clauses, existential and universal quantification, negation, logical connectives, as well as several modalities. The parser for ACE, APE, translates ACE texts into discourse representation structures (DRSs) [15]. DRSs are constructed dynamically to support anaphora resolution, but once computed can be translated to first oder logical formulas. Moreover, APE can verbalise DRSs into two different fragments of ACE

- Core ACE and NP ACE [56]. To simplify matters for the reader in the following we refer only to the first order logic output and not the DRS output of APE.

AceRules builds on APE to provide an ACE-based interface to formal rule systems. More specifically, the target formal rule systems are logic programs under the stable [48] as well as the courteous semantics [53]. Thus, the rule systems are as defeasible theories described in Section 2.5, yet have no special connective for making defeasible rules explicit and have negation as failure in addition to strong negation. Evaluation via the courteous semantics also allows defining priorities over rules but is, on the other hand, only defined for acyclic programs. AceRules relies on external solvers for evaluation via the stable semantics and includes a native implementation of the courteous semantics.

A large part of AceRules consists in taking the DRS or first order logic output as given by APE and applying a series of transformations with the goal of making this output conform to the syntax of logical rules. Roughly, AceRules works by first, more or less, checking the DRSs that result from applying APE to the input ACE texts, filtering out those with DRSs (or first order logic formulas) that involve "wide literals", i.e. negated or non-negated conjunctions of atoms, as well as implications with wide literals in the body and a single wide literal in the head[14]. Then several transformations are applied to the DRSs that survive this first check to attempt to make them conform to the syntax of logic programs. These transformations are *predicate condensation*, *grouping*, and *skolemisation*.

The first of these transformations is "predicate condensation". APE produces reified or flat notations for logical atoms that are used for reasoning in first order logic systems. However, such notations are drawbacks when using the APE output for reasoning via rule. By reified notation we mean that there are, for instance, general predicates for elements of a syntactic class (see below), such as *object* introduced by nouns, *property* introduced by adjectives, *predicate* introduced by verbs, *adverbial modifier* introduced by verb phrase adverbs and propositional phrases. This is in contrast to each noun, adjective, verb, etc. being represented as a predicate *per se*, e.g. as *give(D,mary,C,A)* instead of reified *predicate(D,give,mary,C,A))*. The problem targeted by "predicate condensation" is that APE introduces representations with variables (D in the example) whereby modifiers can modify verbs [15]. Such representations cannot be processed in a rule-based reasoning system, whenever the variables that are introduced are implicitly existentially quantified over (as there are no existentially quantified variables in rule-based systems).

For example, the sentence "If someone asks Mary then Mary happily gives him/her a cat." is represented in the first order logic output of APE (with some simplifications) as[16]

$$\forall AB\Big((object(A, somebody) \wedge predicate(B, ask, A, mary)) \rightarrow$$
$$(\exists CD(object(C, cat) \wedge modifier\_adv(D, happily) \wedge predicate(D, give, mary, C, A)))\Big)$$

"Predicate condensation" has, as the name suggests, the effect of aggregating predicates referring to verbs with those of their modifiers, removing the variables introduced to denote verbs. In the above

[14] Sub-formulas involving double implications are also allowed as they can be transformed into the desired syntactic form. We also ignore here the treatment given by AceRules of modalities. We based our investigation of the internal mechanisms of AceRules on the very clear and well commented source code of AceRules. Grouping, which we also describe in this section, is considered in [60].

[15] As in Neo-Davidsonian Event-theoretic semantics [76].

[16] The example is a simplification for "If someone asks Mary *to give her a cat*, then Mary happily gives her a cat."

(simplified) example the result would be as follows, where the *adverbial modifier* is an argument of the new predicate *pred_mod* used to denote verbs and their modifiers[17]:

$$\forall A\Big((object(A, somebody) \wedge pred\_mod(ask, A, mary, [])) \rightarrow (\exists C(object(C, cat) \wedge pred\_mod(give, mary, C, A, [modifier\_adv(happily)]))\Big)$$

The second transformation applied by AceRules on DRSs, "grouping", targets the problem that can also be seen in our current example. This is the problem of having several atoms occurring in the heads of implications. An additional problem arises where several atoms occur within the scope of a (strict or default) negation, in particular when existentially quantified variables occur in the atoms[18]. Grouping, as with predicate condensation, amounts to aggregating predicates and removing variables, but in contrast to predicate condensation, certain restrictions apply. The restrictions are first of all that the removed variables do not occur outside of the group (these are either universally quantified or are existentially quantified and subject to skolemisation later on). Secondly, there must not be any other group of predicates in the program that unifies with the aggregated predicates, but where the variable corresponding to the removed variable in the latter set of predicates is referred to by some other predicate that is not part of the unifying group. This second restriction is because in order for grouping to succeed to aggregate predicates, e.g. in the head of an implication, it must thereafter also be possible to apply grouping on all groups of predicates occurring in the program that unify with the initially aggregated predicates. In our current example, grouping in the head of the rule would result, more or less, in the rule

$$\forall A\Big((object(A, somebody) \wedge pred\_mod(ask, A, mary, [])) \rightarrow group([object(gv(0), cat), pred\_mod(give, mary, gv(0), A, [modifier\_adv(happily)])])\Big)$$

where the variable C has been replaced with a constant $gv(0)$. This amounts to having one atom representing "Mary happily gives someone a cat". If one were to add the sentence "Mary happily gives Julia a cat." to our example then the resulting predicates would also be grouped by AceRules in a consistent manner resulting in the fact

$$group([object(gv(0), cat), pred\_mod(give, mary, gv(0), julia, [modifier\_adv(happily)])$$

[17] Relations introduced by the word "of" as in "Jack is the brother of John" receive the same treatment. Also the predicate "be" occurs in "Jack is the brother of John" and requires special treatment. We note also that there is a bug in AceRules regarding predicate condensation and their analogues in that modifiers of predicates are grouped together in unordered lists, while at an abstract level they should be represented as sets. Finally, we note that predicate condensation in AceRules follows APE in considering separate occurrences of the same verbs but with different modifiers as a-priori unrelated, e.g. from "Mary loves John passionately. Mary loves John fiercely." it does not follow without further ado that "Mary loves John passionately and fiercely".

[18] We note that in reality only the occurrence of existentially quantified variables within atoms presents a serious problem for rule systems since cases where this is not true can be handled by some rewriting of the relevant rules or negative wide literals using several rules. The use of grouping to also deal with these latter cases therefore unnecessarily restricts the texts that AceRules can handle.

If one were to add, right after “Mary happily gives Julia a cat.”, the sentence “The cat is cute.” grouping would fail as reference to a cat independent of Mary happily giving someone the cat in question is needed[19].

A final transformation that AceRules applies on the DRSs that result from APE is skolemisation of (existentially quantified) variables occurring in positive facts. Thus, for instance, the text “Mary happily gives Julia a cat. The cat is cute.” (without the rule “If someone asks Mary then Mary happily gives him/her a cat.”) results in the formula

$$\begin{aligned}&object(sk(0), cat) \wedge \\ &pred\_mod(give, mary, sk(0), julia, [modifier\_adv(happily)]) \wedge \\ &property(sk(0), cute)\end{aligned}$$

where the skolem constant $sk(0)$ has been introduced in place of the variable referring to the cat that Mary gives Julia.

If and when AceRules transforms the DRSs that result from APE to rules, the system then evaluates the rules using the semantics desired by the user. The outputs of the evaluation (the answer sets) are then converted to the DRS format (this also involves reversing skolemisation, grouping, and predicate condensation), in order to make use of the APE (to Core ACE) verbalisation component, which finally outputs the results using ACE.

To this point, we have reviewed what is built into AceRules and the transformations that are required to facilitate connection to the available stable and courteous semantics. However, as we want to modify AceRules to integrate with a defeasible rule system, we must modify the interface. In the following, we have two ways to interface AceRules with our defeasible rule system. In Section 3.2, a script mutates the output of the AceRules parser into a form that can be further processed with the rule system. This approach works with our extended example in Section 3.4. In addition, a further, more substantive integration is presented in Section 3.3, which works with benchmark test examples, yet introduces additional matters to be addressed in future work.

3.2. An Simple Prototype on Top of AceRules

In this section, we outline one rather simple approach to integrating defeasible rules tied to AceRules.

As an initial testing-ground for our approach, we wrote a relatively simple script that mutates AceRules into a CNL interface to defeasible theories evaluated under our direct stable semantics. Concretely, at the CNL level our script allows for the user to specify assumptions and defeasible rules in addition to facts and strict rules using the constructs *It is usual that ...* and *If ... then it is usual that*

[19] There are several shortcomings in the current implementation of the grouping phase in AceRules. We briefly review some of them. In the first place, as is documented in the AceRules source code, groups are treated as unordered lists of predicates, while they should be treated as sets. Secondly, there should also be no strict sub-groups of a set of grouped predicates occurring in the program. This is quite a serious restriction. For instance, simply adding “All cats are cute.” instead of “Mary happily gives Julia a cat. The cat is cute.” to our example should also result in an error (yet it does not). This is because the sentence "All cats are cute." also requires reference to a “cat” independent of whether Mary has happily given the cat to anyone. Thirdly, matching of groups should be done considering more general versions of the groups. E.g., were one to have the rule “If *John* asks Mary then Mary happily gives him a cat.” instead of “If *someone* asks Mary then Mary happily gives him/her a cat.”, then the predicates aggregated by AceRules representing "Mary happily gives *John* a cat." do not match with those in the body of the rule "If Mary happily gives *someone* a cat then Mary is happy." when they should.

. . . .[20] Modalities and negation as failure, on the other hand, are not supported. On the reasoning side we evaluate defeasible theories obtained with the help of AceRules using the encodings of the direct stable semantics to ASP referred to in Section 2.6.

For our script we separated the AceRules parser and verbaliser components (which in turn, as explained in Section 3.1, make use of the APE parser and verbalisation tools). The script then consists in a interleaving of calls to the AceRules (+ APE) parser, the encodings for the direct-stable semantics with an ASP solver, and finally the AceRules (+ APE) verbaliser.

Crucially, we pre-process the input text removing all constructs indicating defeasibility and make use of the AceRules (+ APE) parser "as if" all rules in the input were strict, but at the same time tracking which rules are defeasible and which are not (for this we make use of labels that can be attached to rules when using the courteous semantics in AceRules). By differentiating the rules in this way, we are able to use the encodings for the direct-stable semantics from Section 2.6 (together with an ASP solver) later on in the pipeline. At the level of the stable sets, the distinction between strict and defeasible rules is irrelevant; and we are, hence, also able to make direct use of the AceRules (+ APE) verbaliser component with the caveats that we mentioned in Section 5.2 of [31].

The result is that we have a script which has the same effect of modifying the internal workings of AceRules (and APE) to support defeasible rules[21]. This demonstrates that we can adapt AceRules to work with defeasible rules and tie it in to the encodings for the direct-stable semantics. More significantly, we were able to run several examples, including all those mentioned in this work[22] using this script.

However, though this approach works, we have identified a range of shortcomings. In particular, grouping as discussed in Section 3.1 (see also the discussion in the footnotes) unnecessarily restricts the fragment of ACE we can target. Also, it is necessary for reasoning in natural language to use existential quantification. Finally, static encodings to ASP (as defined in Section 2.6) raise problems for reasoning with defeasible theories with variables. Given these shortcomings, we implemented a more substantive revision of an interface to defeasible theories. This approach relies on APE and builds on AceRules, but also deviates from AceRules in some significant aspects. This is outlined in the next section.

3.3. Towards an Interface with Existential, Defeasible Rules with ACE

Given the shortcomings of the prototype described in Section 3.2, we developed an ACE-based interface to defeasible theories which is a modular system that builds on but is independent of AceRules. This system differs with respect to the previous one in two important regards:

(1) First, we target a more expressive rule language, allowing more general forms of (defeasible) rules as well as a restricted form of existential quantification. This language is encoded into defeasible theories with function symbols[23]. Clearly, existential quantification is unavoidable in natural language. Moreover, there are ASP solvers which can handle restricted forms of existential quantification in an efficient manner.

[20] The phrase 'it is usual that' does not appear in the body of rules.

[21] The script together with other necessary files as well as the examples mentioned in [31] is available at `https://www.dbai.tuwien.ac.at/proj/adf/dAceRules`.

[22] The test-set that comes with AceRules (with modifications whenever negation as failure or priorities over rules are used), all ACE versions of the examples mentioned in Section 2.5, and the different versions of the example in Section 3.4

[23] Although not included in the definition of defeasible theories from Section 2.5, function symbols can be used when specifying the rules for the encodings as shown in Section 2.6 if the ASP solver used at the back-end also supports them.

(2) Second, rather than using the static encodings at the back-end (see Section 2.6), we use dynamic encodings of defeasible theories to ASP. By doing this we can piggyback on the advanced grounding techniques incorporated in modern ASP solvers and do not need to re-invent (or re-implement) them.

Turning to a more detailed description of this approach, the extended rule language we target consists of rules of the form

$$b_1, \ldots, b_m, \neg(n_1^1, \ldots, n_{u_1}^1), \ldots, \neg(n_1^s, \ldots, n_{u_s}^s) \triangleright H$$

where e.g. $\neg(n_1^1, \ldots, n_{u_1}^1)$ is a "wide literal" and $\triangleright \in \{\rightarrow, \Rightarrow\}$ and H is of the form $h_1, \ldots, h_n$ or $\neg(h_1, \ldots, h_n)$. Also, $h_1, \ldots, h_n, b_1, \ldots, b_m, n_1^1, \ldots, n_{u_1}^1, \ldots, n_1^s, \ldots, n_{u_s}^s$ are atoms as defined in Section 2.5 and $m, s \geqslant 0$, and $n, u_1, \ldots, u_s \geqslant 1$. Variables occurring in the negative atoms $n_1^1, \ldots, n_{u_1}^1, \ldots, n_1^s, \ldots, n_{u_s}^s$ but not in the positive atoms $b_1, \ldots, b_m$ are interpreted as existentially quantified. The same holds for those variables occurring in the head H but not in the positive atoms in the body $b_1, \ldots, b_m$.

We provide meaning to such "existential defeasible rules" via encodings to defeasible theories in a way that is quite similar to the encoding of $\exists$-ASP into ASP as defined by [45]. Namely, we have a "normalization" phase to remove the conjunctions of atoms from negative parts of the rules as well as remove existential variables from these negative parts. We also use skolemization to remove existential variables in the positive heads of rules. Moreover, we have an "expansion" phase to remove conjunctions of atoms in positive heads of the rules.

Crucial differences are due to the fact that our interpretation of negation is different to that of $\exists$-ASP and that we can make use of defeasible implication. Hence we allow negation over conjunctions of atoms in the heads of rules which we need to treat. Furthermore, our "normalization" phase is different to that needed for $\exists$-ASP. Nevertheless, the technicalities are more or less standard, and there are sufficient similarities that it should be relatively straightforward for the interested reader to translate the formal exposition of [45] to that of our scenario, so we rather give an example-based explanation of our translation here. We consider examples where only strict rules occur, as the treatment for defeasible rules is analogous.

To start, let us consider the ACE rule "If someone owns a house then he/she owns a car". The result of parsing this sentence using APE is, with some simplifications, the rule

$$\forall ABC\big((object(A, somebody) \wedge object(B, house) \wedge predicate(C, own, A, B)) \rightarrow \exists ED(object(D, car) \wedge predicate(E, own, A, D))\big)$$

As an existential rule this can be written as follows:

$$object(A, somebody), object(B, house), predicate(C, own, A, B) \rightarrow object(D, car), predicate(E, own, A, D)$$

In our translation this results in the following non-existential rules:

$$object(A, somebody), object(B, house), predicate(C, own, A, B) \rightarrow xxx\_auxPH1(A, B, C)$$
$$xxx\_auxPH1(A, B, C) \rightarrow object(xxx\_sk1(A, B, C), car)$$
$$xxx\_auxPH1(A, B, C) \rightarrow predicate(xxx\_sk2(A, B, C), own, A, xxx\_sk1(A, B, C))$$

where $xxx\_auxPH1(A, B, C)$ is an auxiliary predicate that replaces the head of the original rule, and $xxx\_sk1(A, B, C)$ as well as $xxx\_sk2(A, B, C)$ are skolem functions standing for the house that A owns and the predicate expressing that A owns a house.

If the original rule has a negative head "if someone owns a house then he/she does not own a car" the result would rather be the following:

$$object(A, somebody)object(B, house), predicate(C, own, A, B) \rightarrow xxx\_aux1(A, B, C)$$
$$object(D, car), xxx\_aux1(A, B, C), predicateName(E) \rightarrow \neg predicate(E, own, A, D)$$
$$predicate(E, own, A, D), xxx\_aux1(A, B, C) \rightarrow \neg object(D, car)$$

where *predicateName* is a new predicate which holds for all constants that refer to predicates that can be derived from the program (the definition requires extra rules which we don't show here) and is required to make the second rule safe (i.e. all variables in the head are also present in the body).

We finally consider the more involved case where the head of a rule involves negated existential variables i.e. as in "if someone does not own a house then he/she owns a car". As a rule this can be written in the following form

$$object(A, somebody), \neg xxx\_auxNB1(A) \rightarrow xxx\_auxPH1(A, B, C)$$

where $xxx\_auxPH1(A, B, C)$ is defined as above and $\neg xxx\_auxNB1(A)$ stands for "A does not own a house". The predicate xxx_auxNB1 can be interpreted in several ways.

Interpretation A. One possible way is similar to that used in [45] and allows $\neg xxx\_auxNB1(A)$ to hold whenever there is nothing in the program that explicitly suggests that A *does* own a house (and A not owning a house does not lead to inconsistency).

$$object(B, house), predicate(C, own, A, B) \rightarrow xxx\_auxNB1(A)$$
$$object(A, somebody) \Rightarrow \neg xxx\_auxNB1(A)$$

Interpretation B. A more "cautious" interpretation requires that $\neg xxx\_auxNB1(A)$ holds only when there is some evidence that A does in fact not own a house; more concretely, for every object mentioned in the program either it is explicitly stated that the object is not a house or that A does not own it:

$$\neg object(B, house), \neg predicate(C, own, A, B) \rightarrow xxx\_VauxNB1(A, B, C)$$
$$\neg object(B, house), predicate(C, own, A, B) \rightarrow xxx\_VauxNB1(A, B, C)$$
$$object(B, house), \neg predicate(C, own, A, B) \rightarrow xxx\_VauxNB1(A, B, C)$$
$$object(A, somebody), objectName(B), predicateName(C) \Rightarrow \neg xxx\_VauxNB1(A, B, C)$$
$$\neg xxx\_VauxNB1(A, B, C) \rightarrow xxx\_auxNB1(A)$$
$$object(A, somebody) \Rightarrow \neg xxx\_auxNB1(A)$$

Interpretation C. Finally, a safe (but inefficient) form of reasoning would allow, in the absence of concrete evidence, to conclude that both A having a house and not having a house are possible. Such an interpretation would allow one to entertain the implications from both circumstances. One way to implement this would be as follows:

$$object(B, house), predicate(C, own, A, B) \rightarrow xxx\_auxNB1(A)$$
$$object(A, somebody) \Rightarrow \neg xxx\_auxNB1(A)$$
$$object(A, somebody) \Rightarrow xxx\_auxNB1(A)$$
$$xxx\_auxNB1(A) \rightarrow object(xxx\_sk1(A), house)$$
$$xxx\_auxNB1(A) \rightarrow predicate(xxx\_sk2(A), own, A, xxx\_sk1(A))$$
$$\neg xxx\_auxNB1(A), predicate(C, own, A, B) \rightarrow \neg object(B, house)$$
$$\neg xxx\_auxNB1(A), object(B, house), predicateName(C) \rightarrow \neg predicate(C, own, A, B)$$

A more involved implementation may include additional rules for $\neg xxx\_auxNB1(A)$ as well as only derive a house that is owned by A (when $xxx\_auxNB1(A)$ is derived) if there is not already a house that A owns.

Turning to the need for dynamic encodings of defeasible theories (now as in Section 2.5; in particular, without existentially quantified variables) to ASP, consider the rule

$$q(X), p(X_1), \ldots, p(X_n) \rightarrow p(X)$$

as an ASP rule, i.e.,

```
p(X) :- q(X), p(X1), ... , p(XN).
```

and assume it needs to be grounded for two constants a and b. Naive grounding would yield 2^{n+1} ground instances of the rule (since for each X_i the grounding procedure can choose between a and b), while most ASP grounders will realize that this rule actually amounts to the rules [21]

```
p(a) :- q(a),p(b)
p(b) :- q(b),p(a)
```

Now consider the specification of this rule in order to evaluate it via our static encodings as shown in Section 2.6[24]:

```
c(a). c(b).

head(p(X,X1,...,XN), p(X)) :- c(X),c(X1),...,c(XN).

body(p(X,X1,...,XN),q(X)) :- c(X),c(X1),...,c(XN).

body(p(X,X1,...,XN),p(X1)) :- c(X),c(X1),...,c(XN).

...

body(p(X,X1,...,XN),p(XN)) :- c(X),c(X_1),...,c(XN).
```

Clearly, here the structure of the original rule is broken and thus the exponential grounding cannot be avoided (as, for instance, the ASP grounder needs to derive the fact $head(p(X,X1,...,XN),p(X))$ for each possible assignment of $X,X1,\ldots,XN$ to a and b in order for the stable encoding to work). This is not to say that a more intelligent grounding mechanism could not also be devised for defeasible theories, rather that this would amount to re-implementing the grounding mechanisms (of a specific ASP-solver), while dynamic encodings allow us to piggyback on the grounding developments for any ASP grounder (+solver) we wish to use. This is particularly useful when using skolemisation, as naive grounding easily leads to infinite groundings in this case.

Our dynamic encodings of defeasible theories to ASP work by first guessing what defeasible rules to apply (for each possible ground instance of the rules) and then checking whether more defeasible rules could have been applied without making the program inconsistent. It filters out those guesses for which the latter does not hold (i.e. applying more defeasible rules would make the program inconsistent). As indicated previously, the dynamic encodings, in contrast to the static encodings from Section 2.6 do not make use of disjunction. The following shows the dynamic encoding of the "Tweety" example (see Example 6) (with some comments, preceded with "%", to aid understanding):

```
%Rules w.r.t. a "base guess" of defeasible rules
%to apply

holds(bird(tweety)).
holds(penguin(tux)).

holds(neg(flies(X))) :- holds(penguin(X)).
holds(bird(X)) :- holds(penguin(X)).

holds(flies(X)) :- holds(bird(X)),apply(r1,X,y).

%The base guess:
```

[24]Recall that *head* and *body* have been introduced in Section 2.6

```
%Choose whether to apply or not apply
%the defeasible rule r1 for input X

apply(r1,X,y) :- not apply(r1,X,n),holds(bird(X)).
apply(r1,X,n) :- not apply(r1,X,y),holds(bird(X)).

%Rules for extending the base guess
%for each input Z to the rule r1
%that the base guess has resulted in
%r1 not being applied

holds(r1,Z,X) :- holds(X),apply(r1,Z,n).

holds(r1,Z,bird(tweety)) :- apply(r1,Z,n).
holds(r1,Z,penguin(tux)) :- apply(r1,Z,n).

holds(r1,Z,neg(flies(X))) :- holds(r1,Z,penguin(X)),apply(r1,Z,n).
holds(r1,Z,bird(X)) :- holds(r1,Z,penguin(X)),apply(r1,Z,n).

holds(r1,Z,flies(X)) :- holds(r1,Z,bird(X)),apply(r1,X,y),apply(r1,Z,n).

holds(r1,Z,flies(Z)) :- holds(r1,Z,bird(Z)),apply(r1,Z,n).

%Rules for defining constraints

inconsistent :- holds(neg(X)),holds(X).

inconsistent(r1,Z) :- holds(r1,Z,neg(X)),holds(r1,Z,X).

%Constraint #1: The "base guess" should not lead to
%inconsistency
:- inconsistent.

%Constraint #2: There should not be any way to extend the
%"base guess" that does not lead to inconsistency

:- not inconsistent, not inconsistent(r1,Z),apply(r1,Z,n).
```

To conclude, we have an initial working version of an implementation incorporating the changes described above[25]. We have only implemented *Interpretation A* for dealing with negated existential variables in the body of rules. We plan to incorporate predicate condensation as defined in AceRules and described in Section 3.1 into our system, but leave this to future work. Predicate condensation is useful to avoid the unnecessary variables introduced by parsing of verbs by APE, which moreover easily

[25] A link to the system will be provided in the final version of this work.

leads to defeasible theories with existential variables that cannot be dealt with in an efficient manner. We have nevertheless been able to successfully run most test-cases (with some modifications in case of there being negation as failure as well as priorities over rules) available for AceRules (around 40 of them) by using partial grounding to deal with variables introduced by the predicate "to be"[26].

3.4. An Extended Example extending AceWiki with defeasible rules

In this section, we discuss an extended working example. Our main aim in this section is to demonstrate by example the added value of defeasible rules and reasoning using a CNL. The results we refer to in this section have been obtained with the system described in Section 3.2. We used that system rather than the newer system described in Section 3.3, because the previous system results in nicer verbalisations than the latter at the moment. We only discuss the examples, results, and issues; the DRSs (without "it is usual that") can all be viewed using ACE's online APE webclient.[27]

We now exemplify and further motivate our approach by showing its use in the context of AceWiki [61],[28] a prototype of an encyclopedia in the style of the popular Wikipedia,[29] but where articles are written using ACE rather than unrestricted natural language. The advantage to using ACE in a wiki is that non-expert users can edit AceWiki entries, while at the same time users can use complex question answering and draw inferences. As it currently stands, AceWiki can represent a consistent KB about some domain and uses only strict rules.

Although the use of full ACE in the context of AceWiki is desirable, the undecidability of ACE (see 43) also means that it is not feasible in practice. Restricting ACE to efficiently decidable fragments, e.g. via translation to a form of rule language, provides a more promising way forward. We base our example on current entries in the AceWiki about geographical information,[30] which have been restricted to a variant of ACE that can be translated into the rule language OWL 2 RL, and thus also, in principle, into the fragment of ACE admitted by AceRules. However, AceWiki could similarly be deployed in other fields such as Biology, Medicine, or more generally in any context where a structured KB would be useful.

As a motivating example, consider the entry for *island* in the geographical AceWiki. Some straightforward statements pertaining to the strict definition of *island* appear, e.g. *Every island is a land-mass* and *Every island is surrounded by a body of water*. Using ACE, such statements can be written in a straightforward manner and are automatically translated to a rule language:[31]

[26] By partial grounding for the predicate "to be" we mean that we replace, for instance, the rule corresponding to "Every woman is happy." with rules corresponding to "If Mary is a woman then Mary is happy.", "If John is a woman then John is happy." in the case that Mary and John are the only persons that are made reference to in the program. This solution allows us to in fact run *all* test-cases for AceRules without having infinite groundings, but there are incorrect answers whenever adjectives don't "add up": e.g. while our implementation would, in contrast to AceRules, *correctly* derive that "Mary is happy" from "Mary is a happy girl. Mary is a person." it also *incorrectly* derives that "Mary is a tall person." from "Mary is a tall girl. Mary is a person." The latter could be incorrect since Mary may just be tall for her very young age. However, the treatment of different classes of adjectives is a matter tangential to our study. See [58] for an introductory discussion of *intersective* and *subsective* adjectives.

[27] APE webclient: `http://attempto.ifi.uzh.ch/site/resources/`

[28] AceWiki can be accessed at `http://attempto.ifi.uzh.ch/acewiki/`.

[29] `https://www.wikipedia.org/`

[30] `http://attempto.ifi.uzh.ch/webapps/acewikigeo/`

[31] We extended the lexicon of ACE with some further terms for complex noun phrases that APE does not (yet) parse, e.g. *land-mass* and *body-of-water*, but suppress further discussion.

```
(1) Every island is a land-mass.
(2) If X is an island then a body-of-water surrounds X.
```

Statement (1) leads to a rule like $\textit{island}(x) \rightarrow \textit{land-mass}(x)$ with a first-order variable x. Due to a lack of space, we will not explicitly present further rules in the paper; in any case, they can be obtained from the presented text via AceRules.

ACE enables the addition of lexical entries, such as proper names *Mainland-Shetland* or *St-Ninians-Isle*. Moreover, ACE is often able to deduce the word class for words that are not in its lexicon from the context. There are some interactions in ACE/AceRules in relation to the verb form, quantifier scope, and the verbaliser (among other subtleties) such that, for example, we have represented (2) as a rule; we suppress further such incidental comments. We do however further note that rule (2) illustrates the situation where the input text introduces an implicitly existentially quantified variable in the head of a rule (here, referring to a body-of-water), which needs to be treated by the grouping-mechanism of AceRules we alluded to in Section 3.1.

The problem, which we develop, is to add a new entry for *tied-island* to this AceWiki. However, as we show, this would lead to inconsistency were we to only have strict rules. According to Wikipedia, tied islands "are landforms consisting of an island that is connected to land only by a tombolo: a spit of beach materials connected to land at both ends."[32] With slight simplification, this definition can be written into AceWiki as follows:

```
(3) Every tied-island is an island.
(4) Every tied-island attaches-to a land-mass.
```

A prominent example of a *tied-island* according to the Wikipedia entry is *St. Ninian's Isle*, which is attached to *Mainland Shetland*, the largest of the Shetland Islands off the coast of Scotland. Thus, entries for St. Ninian's Isle and Mainland Shetland in AceWiki would be:

```
(5) Mainland-Shetland is an island.
(6) St-Ninians-Isle is a tied-island.
(7) St-Ninians-Isle is a part of the Shetland-Islands.
```

According to the Wikipedia entry for St. Ninian's Isle, during the winter strong wave action removes sand from the tombolo that connects St. Ninian to Mainland Shetland such that the tombolo is usually covered at high tide and occasionally throughout the tidal cycle. Hence, simply stating that *St. Ninian's Isle attaches to Mainland Shetland* would be incorrect. Spelling out the exact conditions under which St. Ninian's Isle is connected to Mainland-Shetland, which corresponds to using exceptions in strict rules, seems quite difficult if even possible (or desirable) and would be rather uncommon for an application like AceWiki. Rather, an easy solution is provided by the use of the predicate *it is usual that* applied to a statement:

```
(8) It is usual that St-Ninians-Isle attaches-to Mainland-Shetland.
```

Let us now turn to a more fundamental reason for being able to distinguish between defeasible and strict statements in a CNL. Consider now the result of having all of the above statements in the AceWiki together with the following fairly uncontroversial statements referring to the meanings of *being attached*

[32] `https://en.wikipedia.org/wiki/Tied_island` (accessed on 4/4/2017)

to a land mass, *being surrounded by water*, and *being a part of*. We initially highlight the issue using further strict rules. In particular, statements (12) and (13) are needed too, because we need to define in the wiki's KB that St. Ninian's Isle is attached to exactly one land mass, namely Mainland Shetland[33].

```
(9) If X attaches-to a land-mass then it is false that a body-of-water
    surrounds X.
(10) If a body-of-water surrounds X then it is false that X
     attaches-to a land-mass.
(11) If St-Ninians-Isle attaches-to Mainland-Shetland then St-Ninians-Isle
     is a part of Mainland-Shetland.
(12) If St-Ninians-Isle attaches-to Mainland-Shetland then St-Ninians-Isle
     attaches-to a land-mass.
(13) If St-Ninians-Isle attaches-to a land-mass then St-Ninians-Isle
     attaches-to Mainland-Shetland.
```

Since according to (6) St. Ninian's Isle is a tied island, and according to (3) every tied island is an island, and both (3) as well as (6) are strict rules, the direct stable semantics forces one to conclude that St. Ninian's Isle is an island. Now, because St. Ninian's Isle is an island and following (2), we conclude that a body of water surrounds St. Ninian's Isle. But from the fact that St. Ninian's is also a tied island and (4), St. Ninian's Isle attaches to a land mass. This leads to a contradiction according to statements (9) and (10). Hence, the entire AceWiki is deemed inconsistent and further reasoning is invalidated.

Note that the AceWiki remains inconsistent even after removing statement (8); the reason for the apparent contradiction in the Wiki is the fact, as is stated in the Wikipedia entry referring to St. Ninian's Isle,[34] that "[d]epending on the definition used, St. Ninian's is [...] either an island, or a peninsula." This reveals that the definition for *tied-island* in (3) should also be *defeasible*. However, in contrast to the reasons for the defeasiblity of (8), this is now due to the fact that there is no consensus on the meaning of *tied island*. Thus, we replace (3) with the more accurate statement:

```
(3') If X is a tied-island then it is usual that X is an island.
```

The consequence is that there is now one stable set:

```
ANSWER-TEXT #1:

There is a body-of-water X1.
St-Ninians-Isle is a tied-island.
Mainland-Shetland is a land-mass.
Mainland-Shetland is an island.
St-Ninians-Isle is a part of Shetland-Islands.
St-Ninians-Isle is a part of Mainland-Shetland.
St-Ninians-Isle attaches-to a land-mass.
The body-of-water X1 surrounds Mainland-Shetland.
St-Ninians-Isle attaches-to Mainland-Shetland.
It is false that Mainland-Shetland attaches-to a land-mass.
It is false that a body-of-water surrounds St-Ninians-Isle.
```

[33] These rules are unnecessary when using the implementation as described in Section 3.3.

[34] `https://en.wikipedia.org/wiki/St_Ninian's_Isle` (accessed on 4.4.2017)

Here the conclusion is that St. Ninian's Isle is a tied island that is attached to Mainland Shetland, while nothing can be said regarding whether St. Ninian's is also an island or not. The reason is that since statement (4) is strict, (8) is also effectively interpreted as a strict rule; that is, (8) strictly holds. To make (4) consistent with the *intended reading* of (8), (4) should be replaced with:

```
(4′) If X is a tied-island then it is usual that X attaches-to a
     land-mass.
```

The result is that there are now two stable sets (answer-texts), which have in common the statements:

```
There is a body-of-water X1.
St-Ninians-Isle is a tied-island.
Mainland-Shetland is a land-mass.
Mainland-Shetland is an island.
St-Ninians-Isle is a part of Shetland-Islands.
The body-of-water X1 surrounds Mainland-Shetland.
It is false that Mainland-Shetland attaches-to a land-mass.
```

One stable set contains the following statements in addition to the common statements:

```
St-Ninians-Isle is a part of Mainland-Shetland.
St-Ninians-Isle attaches-to a land-mass.
St-Ninians-Isle attaches-to Mainland-Shetland.
It is false that a body-of-water surrounds St-Ninians-Isle.
```

The other stable set contains the following statements in addition to the common statements:

```
There is a body-of-water X2.
St-Ninians-Isle is a land-mass.
St-Ninians-Isle is an island.
The body-of-water X2 surrounds St-Ninians-Isle.
It is false that St-Ninians-Isle attaches-to a land-mass.
```

The interpretation of the latter set of statements is that St. Ninian's Isle is a tied island, but can only be called an island when it is not attached to Mainland-Shetland. Also relaxing the definition of *island* by changing (2) to

```
(2′) If X is an island then it is usual that a body-of-water
     surrounds X.
```

has the consequence that St. Ninian's Isle can also (always) be considered an island, despite the fact that the isle is not always surrounded by water.

Summarizing, we have shown that by distinguishing between defeasible and strict statements, we can resolve apparent inconsistencies such as might arise, in our example, because of the use of generic statements that allow for exceptions or because different meanings can be attached to certain words.

However, our approach does not require explicit statement of exceptions or alternatives. Using non-artificial, specific exceptions together with negation-as-failure in strict rules is often not feasible nor desirable. More fundamentally, using artificial exceptions, e.g. *abnormality* predicates specific to each

rule, will usually not lead to a satisfactory result. Consider, for instance the effect of having the statement (8") below rather than the statement (8) mentioned previously, while replacing (3) with (3") rather than (3'), (4) with (4") rather than (4'), as well as (2) with (2") rather than (2').

```
(8'')If it is not provable that it is false that St-Ninians-Isle attaches-to
     Mainland-Shetland then St-Ninians-Isle attaches-to Mainland-Shetland.
(3'') If X is a tied-island and it is not provable that X is not an
      island then X is an island.
(4'') If X is a tied-island and it is not provable that it is false that
      X attaches-to a land-mass then X attaches-to a land-mass.
(2'') If X is an island and it is not provable that it is false that a
      body-of-water surrounds X then a body-of-water surrounds X.
```

The resulting text does not have any answer set under the standard stable semantics for logic programs.[35] Interpreting the text under the courteous semantics does produce a unique answer set, but this approach is unsatisfactory in general. First, because the rules must be acyclic and second because the resulting answer is often uninformative or somewhat arbitrary. In the current case, the rules are in fact cyclic and hence no answer is produced.

An auxiliary point is that the discussion above shows the utility of the tool, for it allows us to experiment using natural language with alternative inputs to determine alternative outputs (and compare semantics between them). From such alternatives, we can identify our preferred inputs, semantics, and outputs, which are intuitively plausible and computationally feasible.

In this section, we have discussed an extended example using the approach from Section 3.2, which relies heavily on AceRules. There are a range of issues and subtleties with the analysis, which we have reported on in [31]: we have not generated explanations; preferences are not used; defeasibility is sometimes expressed contextually rather than grammatically; the semantic scope of the modal operator remains to be explicated. These and other matters remain for future research and development.

4. Discussion

The paper has motivated the development of a CNL with defeasible rules, adapting the AceRules system with defeasible rules. We have provided some background on defeasible theories and presented a semantics for defeasible theories that facilitates an implementation via answer set programming, yet caters for the expressive needs of AceRules. We have next provided a discussion about how defeasible rules are introduced to AceRules. We have presented an extended example to exercise the tool, showing the utility and advantages of representing and reasoning with defeasible rules.

Although we argue for our approach from first principles, several of its elements have precursors in the literature. In slightly more distant related work, Denecker et al. [28] introduced a general theory of justifications, where there are also rules involving literals; however, they have a decidedly more philosophical/mathematical motivation. For example, they allow infinite justifications, which is not immediately useful for our setting.

[35] This is not to say that it is not possible to simulate the evaluation of ACE texts under the direct-stable semantics by using logic programs without the defeasible conditional; in fact the encodings referred to in sections 2.6 as well as 3.3 provide such a simulation (the first via disjunctive logic programs, while the second are dynamic). On the other hand, the complexity results from Section 2.4 also suggest that any such simulation via normal or extended logic programs will involve a worst-case exponential blow-up in general (unless the polynomial hierarchy collapses to its first level), at least for ACE texts which can be parsed as *grounded* defeasible theories.

While we noted limitations of approaches using structured arguments, such approaches are still relevant from a conceptual point of view, e.g. in modularising argument for planning. With our contribution, we move those approaches closer to actual implementations by reformulating their semantics and decoupling the (necessary) semantical computation from (optional) argument construction.

There are numerous opportunities for further development. ACE's APE parser can be augmented with capacities to represent tense, additional verb constructions, and subordinate clauses for justifications such as *because*. The example can be incrementally extended, testing the output results to ensure they comply with intuitions. There are a range of issues to address about the verbalisation, most importantly to provide some means to reconstruct claims and justifications in the output results, e.g. "X because Y", rather than producing a list of statements. More fundamentally, the interactions between the rule expressions output by AceRules, the rule language of the inference engine, and the verbaliser need greater study and development. As it is, AceRules uses grouping to adjust DRSs to the rule language, which raises several complexities; whether this can be relaxed to take advantage of recent advances in rules languages remains to be seen. Relatedly, further work needs to be carried out on the optimisation of grounding of variables, which has not been discussed in this paper. Other avenues of investigation might be rule decomposition, which could help to optimise grounding. More generally, we expect to consider incorporating other natural language expressions of defeasibility or genericity into a CNL with defeasible rules. In [65], a range of generic expressions are discussed, some of which allow for defeasibility (e.g., *Lions have manes*) and some that do not (e.g., *Dogs are mammals*). While generic expressions have some similarity to *it is usual that/usually*, they are not equivalent. In the introduction, we suggested a connection between argument mining and a CNL approach, wherein arguments that are mined could perhaps be processed further in order to be sufficiently analysed to suit an inference engine. We would look to extending our exploration of such matters in subsequent work. Clearly, the approach we have developed opens a range of avenues for future research.

References

[1] L. Amgoud and P. Besnard, A Formal Characterization of the Outcomes of Rule-Based Argumentation Systems, in: *SUM*, LNCS, Vol. 8078, Springer, 2013a, pp. 78–91.

[2] L. Amgoud and P. Besnard, Logical limits of abstract argumentation frameworks, *Journal of Applied Non-Classical Logics* **23**(3) (2013b), 229–267.

[3] L. Amgoud and F. Nouioua, Undercutting in Argumentation Systems, in: *SUM*, LNCS, Vol. 9310, Springer, 2015, pp. 267–281.

[4] I. Androutsopoulos and P. Malakasiotis, A Survey of Paraphrasing and Textual Entailment Methods, *Journal of Artificial Intelligence Research* **38** (2010), 135–187.

[5] A. Anonymous, Digital intuition: A computer program that can outplay humans in the abstract game of Go will redefine our relationship with machines, *Nature* **529** (2016), 437, Editorial..

[6] G. Antoniou, D. Billington, G. Governatori and M. Maher, A Flexible Framework for Defeasible Logics, in: *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, H.A. Kautz and B.W. Porter, eds, AAAI Press / The MIT Press, 2000, pp. 405–410.

[7] N. Asher, *Reference to Abstract Objects in Discourse*, Kluwer Academic Publishers, 1993.

[8] M. Balduccini and M. Gelfond, Logic programs with consistency-restoring rules, in: *International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series*, Vol. 102, 2003.

[9] C. Baral and M. Gelfond, Logic Programming and Knowledge Representation, *Journal of Logic Programing* **19/20** (1994), 73–148.

[10] P. Baroni and M. Giacomin, Semantics of Abstract Argument Systems, in: *Argumentation in Artificial Intelligence*, G.R. Simari and I. Rahwan, eds, Springer, 2009, pp. 25–44.

[11] P. Baroni, F. Cerutti, M. Giacomin and G.R. Simari (eds), Computational Models of Argument: Proceedings of COMMA 2010, Desenzano del Garda, Italy, September 8-10, 2010, in: *COMMA*, Frontiers in Artificial Intelligence and Applications, Vol. 216, IOS Press, 2010. ISBN 978-1-60750-618-8.
[12] P. Baroni, D. Gabbay, M. Giacomin, B. Liao and L. van der Torre (eds), *Handbook of Formal Argumentation*, Vol. 1, College Publications, 2017.
[13] P. Besnard and A. Hunter, Practical First-order Argumentation, in: *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, AAAI Press, 2005, pp. 590–595.
[14] P. Besnard and A. Hunter, Argumentation based on classical logic, in: *Argumentation in Artificial Intelligence*, Springer, 2009, pp. 133–152.
[15] P. Blackburn and J. Bos, *Representation and Inference for Natural Language: A First Course in Computational Semantics*, CSLI Publications, 2005.
[16] A. Bondarenko, P.M. Dung, R. Kowalski and F. Toni, An Abstract, Argumentation-Theoretic Approach to Default Reasoning, *Artificial Intelligence* **93** (1997), 63–101. `citeseer.ist.psu.edu/bondarenko97abstract.html`.
[17] J. Bos, Wide-Coverage Semantic Analysis with Boxer, in: *Semantics in Text Processing. STEP 2008 Conference Proceedings*, J. Bos and R. Delmonte, eds, Research in Computational Semantics, College Publications, 2008, pp. 277–286.
[18] G. Brewka, Preferred Subtheories: An Extended Logical Framework for Default Reasoning, in: *AAAI*, 1989, pp. 1043–1048.
[19] G. Brewka and T.F. Gordon, Carneades and Abstract Dialectical Frameworks: A Reconstruction, in: *Proceedings of COMMA 2010*, P. Baroni, F. Cerutti, M. Giacomin and G.R. Simari, eds, Frontiers in Artificial Intelligence and Applications, Vol. 216, IOS Press, 2010, pp. 3–12. ISBN 978-1-60750-618-8.
[20] G. Brewka and S. Woltran, Abstract Dialectical Frameworks, in: *KR*, 2010, pp. 102–211.
[21] G. Brewka, T. Eiter and M. Truszczynski, Answer set programming at a glance, *Commun. ACM* **54**(12) (2011), 92–103.
[22] G. Brewka, I. Niemelä and M. Truszczynski, Preferences and Nonmonotonic Reasoning, *AI Magazine* **29**(4) (2008), 69–78.
[23] E. Cabrio and S. Villata, Natural Language Arguments: A Combined Approach, in: *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, 2012, pp. 205–210.
[24] E. Cabrio and S. Villata, Abstract Dialectical Frameworks for Text Exploration, in: *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016), Volume 2, Rome, Italy, February 24-26, 2016.*, SciTePress, 2016, pp. 85–95.
[25] M. Caminada and L. Amgoud, On the evaluation of argumentation formalisms, *Artificial Intelligence* **171**(5–6) (2007), 286–310.
[26] R. Craven and F. Toni, Argument graphs and assumption-based argumentation, *Artificial Intelligence* **233** (2016), 1–59.
[27] B. Davis, G. Pace and A. Wyner (eds), Controlled Natural Language - 5th International Workshop, CNL 2016, Aberdeen, UK, July 25-27, 2016, Proceedings, in *Lecture Notes in Computer Science*, Vol. 9767, Springer, 2016.
[28] M. Denecker, G. Brewka and H. Strass, A Formal Theory of Justifications, in: *LPNMR*, Springer, 2015, pp. 250–264.
[29] M. Denecker, V.W. Marek and M. Truszczyński, Uniform Semantic Treatment of Default and Autoepistemic Logics, *Artificial Intelligence* **143**(1) (2003), 79–122.
[30] M. Denecker, D. Theseider-Dupré and K. van Belleghem, An Inductive Definition Approach to Ramifications, *Linköping Electronic Articles in Computer and Information Science* **3**(7) (1998), 1–43.
[31] M. Diller, A. Wyner and H. Strass, Defeasible AceRules: A Prototype, in: *Proceedings of the Twelfth International Conference on Computational Semantics (IWCS)*, C. Gardent and C. Retoré, eds, 2017.
[32] Y. Dimopoulos and A. Torres, Graph Theoretical Structures in Logic Programs and Default Theories, *Theoretical Computer Science* **170**(1–2) (1996), 209–244.
[33] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, S. Schieber, J. Waldo, D. Weinberger and A. Wood, Accountability of AI Under the Law: The Role of Explanation, *ArXiv e-prints* (2017).
[34] 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) (1995), 321–358.
[35] P.M. Dung, An axiomatic analysis of structured argumentation with priorities, *Artificial Intelligence* **231** (2016), 107–150.
[36] P.M. Dung and T.C. Son, An argument-based approach to reasoning with specificity, *Artificial Intelligence* **133**(1–2) (2001), 35–85.
[37] P.M. Dung, R. Kowalski and F. Toni, Assumption-Based Argumentation, in: *Argumentation in Artificial Intelligence*, Springer, 2009, pp. 199–218.
[38] P.E.. Dunne and T.J.M.. Bench-Capon, Coherence in Finite Argument Systems, *Artificial Intelligence* **141**(1/2) (2002), 187–203.
[39] P.E. Dunne and M. Wooldridge, Complexity of Abstract Argumentation, in: *Argumentation in Artificial Intelligence*, Springer, 2009, pp. 85–104.

[40] D. Edgington, Conditionals, in: *Stanford Encyclopedia of Philosophy*, Stanford University, 2006, http://plato.stanford.edu/entries/conditionals/.
[41] T. Eiter and G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, *Annals of Mathematics and Artificial Intelligence* **15**(3–4) (1995), 289–323.
[42] N.E. Fuchs, Reasoning in Attempto Controlled English: Non-monotonicity, in: *Controlled Natural Language – 5th International Workshop, CNL 2016, Aberdeen, UK, July 25-27, 2016, Proceedings*, B. Davis, G. Pace and A. Wyner, eds, Lecture Notes in Computer Science, Vol. 9767, Springer, 2016, pp. 13–24.
[43] N.E. Fuchs, K. Kaljurand and T. Kuhn, Attempto Controlled English for Knowledge Representation, in: *Reasoning Web*, 2008, pp. 104–124.
[44] A.J. García and G.R. Simari, Defeasible Logic Programming: An Argumentative Approach, *Theory and Practice of Logic Programming* **4**(1) (2004), 95–137.
[45] F. Garreau, L. Garcia, C. Lefèvre and I. Stéphan, ∃-ASP, in: *JOWO@IJCAI*, CEUR Workshop Proceedings, Vol. 1517, CEUR-WS.org, 2015.
[46] M. Gebser, R. Kaminski, B. Kaufmann and T. Schaub, Answer set solving in practice, *Synthesis Lectures on Artificial Intelligence and Machine Learning* **6**(3) (2012), 1–238.
[47] M. Gelfond and Y. Kahl, *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*, Cambridge University Press, 2014.
[48] M. Gelfond and V. Lifschitz, The Stable Model Semantics for Logic Programming, in: *Proceedings of the International Conference on Logic Programming (ICLP)*, The MIT Press, 1988, pp. 1070–1080.
[49] M. Gelfond and V. Lifschitz, Logic Programs with Classical Negation, in: *Logic Programming, Proceedings of the Seventh International Conference, Jerusalem, Israel, June 18-20, 1990*, D.H.D. Warren and P. Szeredi, eds, MIT Press, 1990, pp. 579–597.
[50] V. Gervasi and D. Zowghi, Reasoning about inconsistencies in natural language requirements, *ACM Trans. Softw. Eng. Methodol.* **14**(3) (2005), 277–330.
[51] G. Gogic, H. Kautz, C. Papadimitriou and B. Selman, The Comparative Linguistics of Knowledge Representation, in: *IJCAI*, Morgan Kaufmann, 1995, pp. 862–869.
[52] G. Governatori, M.J. Maher, G. Antoniu and D. Billington, Argumentation Semantics for Defeasible Logic, *Journal of Logic and Computation* **14**(5) (2004), 675–702.
[53] B.N.. Grosof, Prioritized Conflict Handling for Logic Programs, in: *Logic Programming, Proceedings of the 1997 International Symposium, Port Jefferson, Long Island, NY, USA, October 13-16, 1997*, J. Maluszynski, ed., MIT Press, 1997, pp. 197–211.
[54] S. Guy and R. Schwitter, The PENG^{ASP} system: Architecture, language and authoring tool, *Language Resources and Evaluation* **51**(1) (2017), 67–92.
[55] J. Heyninck and C. Straßer, Relations between assumption-based approaches in nonmonotonic logic and formal argumentation, in: *Proceedings of NMR 2016*, 2016.
[56] K. Kaljurand, Paraphrasing Controlled English Texts, in: *CNL (Pre-Proceedings)*, CEUR Workshop Proceedings, Vol. 448, CEUR-WS.org, 2009.
[57] H. Kamp and U. Reyle, *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language: Formal Logic and Discourse Representation Theory*, Springer, 1993.
[58] C. Kennedy, Adjectives, in: *Routledge Companion to Philosophy of Language*, Routledge, 2012, pp. 324–441.
[59] A. Kratzer, Modals and Conditionals, Oxford University Press, 2012.
[60] T. Kuhn, AceRules: Executing Rules in Controlled Natural Language, in: *Web Reasoning and Rule Systems, First International Conference, RR 2007, Innsbruck , Austria, June 7-8, 2007, Proceedings*, Lecture Notes in Computer Science, Vol. 4524, Springer, 2007, pp. 299–308.
[61] T. Kuhn, AceWiki: A Semantic Wiki Using Controlled English, in: *Proceedings of the Poster Session at the 6th European Semantic Web Conference (ESWC09)*, 2009.
[62] T. Kuhn, An Evaluation Framework for Controlled Natural Languages, in: *Proceedings of the Workshop on Controlled Natural Language (CNL 2009)*, N.E. Fuchs, ed., Lecture Notes in Computer Science, Vol. 5972, Springer, Berlin / Heidelberg, Germany, 2010, pp. 1–20. ISBN 978-3-642-14417-2.
[63] T. Kuhn, A Survey and Classification of Controlled Natural Languages, *Computational Linguistics* **40**(1) (2014a), 121–170.
[64] T. Kuhn, A Survey and Classification of Controlled Natural Languages, *Computational Linguistics* **40**(1) (2014b), 121–170.
[65] S.-J. Leslie and A. Lerner, Generic Generalizations, in: *The Stanford Encyclopedia of Philosophy*, Winter 2016 edn, E.N. Zalta, ed., Metaphysics Research Lab, Stanford University, 2016.
[66] D. Lewis, Adverbs of Quantification, in: *Formal Semantics of Natural Language*, Cambridge University Press, 1975a, pp. 178–188.

[67] D. Lewis, Adverbs of Quantification, in: *Formal Semantics of Natural Language*, Cambridge University Press, 1975b, pp. 178–188.
[68] Y. Lierler and V. Lifschitz, Logic Programs vs. First-Order Formulas in Textual Inference, in: *IWCS*, The Association for Computer Linguistics, 2013a, pp. 340–346.
[69] Y. Lierler and V. Lifschitz, Logic programs vs. first-order formulas in textual inference, *Computer Science Faculty Proceedings & Presentations* (2013b), Paper 16..
[70] M. Lippi and P. Torroni, Argumentation Mining: State of the Art and Emerging Trends, *ACM Transactions on Internet Technology* **16**(2) (2016), 10–11025.
[71] J. McCarthy, Circumscription - A Form of Non-Monotonic Reasoning, *Artificial Intelligence* **13**(1–2) (1980), 27–39.
[72] J. McCarthy, Applications of Circumscription to Formalizing Common-Sense Knowledge, *Artificial Intelligence* **28**(1) (1986), 89–116.
[73] S. Modgil and H. Prakken, A general account of argumentation with preferences, *Artificial Intelligence* **195** (2013), 361–397.
[74] D. Mott, The ITA Controlled English Report (Prolog version), Technical Report, Emerging Technology Services, Hursley, IBM UK, 2016.
[75] D. Nute, Defeasible Logic, in: *Handbook of Logic in Artificial Intelligence and Logic Programming*, Vol. 3, Oxford University Press, 1994, pp. 353–395.
[76] T. Parsons, *Events in the Semantics of English: A Study of Subatomic Semantics*, 1s edn, MIT Press, Cambridge, MA, USA, 1990.
[77] A. Peldszus and M. Stede, From Argument Diagrams to Argumentation Mining in Texts: A Survey, *IJCINI* **7**(1) (2013), 1–31. doi:10.4018/jcini.2013010101. `https://doi.org/10.4018/jcini.2013010101`.
[78] J.L.. Pollock, Reasoning and probability, *Law, Probability and Risk* **6** (2007), 43–58.
[79] D. Poole, A Logical Framework for Default Reasoning, *Artificial Intelligence* **36**(1) (1988), 27–47.
[80] H. Prakken, An Abstract Framework for Argumentation with Structured Arguments, *Argument & Computation* **1**(2) (2010), 93–124.
[81] I. Rahwan and G. Simari (eds), *Argumentation in Artificial Intelligence*, Springer, 2009.
[82] R. Reiter, On Reasoning by Default, in: *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, Association for Computational Linguistics, Morristown, NJ, USA, 1978, pp. 210–218.
[83] R. Reiter, A Logic for Default Reasoning, *Artificial Intelligence* **13** (1980), 81–132.
[84] R. Reiter and G. Criscuolo, On Interacting Defaults, in: *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, William Kaufmann, 1981, pp. 270–276.
[85] N. Rescher and R. Manor, On Inferences from Inconsistent Premises, *Theory and Decision* **1**(2) (1970), 179–217.
[86] C. Schulz and F. Toni, Justifying answer sets using argumentation, *Theory and Practice of Logic Programming* **16**(1) (2016), 59–110.
[87] S. Schulz, A Comparison of Different Techniques for Grounding Near-Propositional CNF Formulae, in: *FLAIRS*, 2002, pp. 72–76.
[88] G. Sileno, A. Boer and T. van Engers, Implementing explanation-based argumentation using answer set programming, in: *Proceedings of the 11th Workshop on Argumentation in Multi-agent Systems, (ArgMass 2014), May 5, 2014, Paris, France*, 2015.
[89] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel and D. Hassabis, Mastering the game of Go with deep neural networks and tree search, *Nature* **529** (2016), 484–489.
[90] G.R. Simari and I. Rahwan (eds), *Argumentation in Artificial Intelligence*, Springer, 2009.
[91] H. Strass, Approximating Operators and Semantics for Abstract Dialectical Frameworks, *Artificial Intelligence* **205** (2013a), 39–70.
[92] H. Strass, Instantiating Knowledge Bases in Abstract Dialectical Frameworks, in: *Proceedings of the Fourteenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XIV)*, LNCS, Vol. 8143, Springer, 2013b, pp. 86–101.
[93] H. Strass, Instantiating rule-based defeasible theories in abstract dialectical frameworks and beyond, *Journal of Logic and Computation* (2015), Preprint available online at `http://dx.doi.org/10.1093/logcom/exv004`..
[94] H. Strass and A. Wyner, On Automated Defeasible Reasoning with Controlled Natural Language and Argumentation, in: *Proceedings of the Second International Workshop on Knowledge-based Techniques for Problem Solving and Reasoning (KnowProS)*, R. Barták, T.L. McCluskey and E. Pontelli, eds, 2017.
[95] F. Toni, A tutorial on assumption-based argumentation, *Argument & Computation* **5**(1) (2014), 89–117.
[96] F. Toni and P. Torroni, Bottom-Up Argumentation, in: *Theorie and Applications of Formal Argumentation - First International Workshop, TAFA 2011. Barcelona, Spain, July 16-17, 2011, Revised Selected Papers*, 2011, pp. 249–262.

[97] A. Toniolo, A.D. Preece, W. Webberley, T.J. Norman, P. Sullivan and T. Dropps, Conversational intelligence analysis, in: *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*, 2016, pp. 42–1426.

[98] K. Van Belleghem, M. Denecker and D. Theseider-Dupré, A Constructive Approach to the Ramification Problem, in: *Reasoning about Actions; Foundations and Applications*, 1998, pp. 1–17.

[99] F. van Eemeren, P. Houtlosser and A.F.S. Henkemans, *Argumentative Indicators in Discourse*, Springer, 2007.

[100] B. Verheij, DefLog: On the Logical Interpretation of Prima Facie Justified Assumptions, *Journal of Logic and Computation* **13**(3) (2003), 319–346.

[101] A. Wyner and H. Strass, dARe - Using Argumentation to Explain Conclusions from a Controlled Natural Language Knowledge Base, in: *Proceedings of the Thirtieth International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2017*, S. Benferhat, K. Tabia and M. Ali, eds, Lecture Notes in Computer Science, Vol. 10351, Springer, 2017, pp. 328–338.

[102] A. Wyner, T. Bench-Capon and P. Dunne, On the Instantiation of Knowledge Bases in Abstract Argumentation Frameworks, in: *Proceedings of CLIMA XIV*, 2013, pp. 34–50.

[103] A. Wyner, W. Peters and D. Price, Argument Discovery and Extraction with the Argument Workbench, in: *Proceedings of the 2nd Workshop on Argumentation Mining, ArgMining@HLT-NAACL 2015, June 4, 2015, Denver, Colorado, USA*, 2015, pp. 78–83.

[104] A. Wyner, T. van Engers and K. Bahreini, From Policy-Making Statements to First-Order Logic, in: *EGOVIS*, 2010, pp. 47–61.

[105] A. Wyner, T. van Engers and A. Hunter, Working on the argument pipeline: Through flow issues between natural language argument, instantiated arguments, and argumentation frameworks, *Argument & Computation* **7**(1) (2016), 69–89.

[106] A.Z. Wyner, J. Bos, V. Basile and P. Quaresma, An Empirical Approach to the Semantic Representation of Laws, in: *Legal Knowledge and Information Systems - JURIX 2012: The Twenty-Fifth Annual Conference, University of Amsterdam, The Netherlands, 17-19 December 2012*, 2012, pp. 177–180. doi:10.3233/978-1-61499-167-0-177. `https://doi.org/10.3233/978-1-61499-167-0-177`.

[107] A. Wyner, T. Bench-Capon, P. Dunne and F. Cerutti, Senses of 'argument' in instantiated argumentation frameworks, *Argument & Computation* **6**(1) (2015), 50–72.