1-2023

Defeasible semantics for L4

Guido GOVERNATORI

Meng Weng (HUANG Mingrong) WONG
Singapore Management University, mwwong@smu.edu.sg

Defeasible Semantics for L4

GUIDO GOVERNATORI and MENG WENG WONG, Singapore Management University, Singapore

Reference Format:
Guido Governatori and Meng Weng Wong. 2023. Defeasible Semantics for L4. In *ProLaLa 2023.* 6 pages.

1 INTRODUCTION

The importance of defeasibility for legal reasoning has been investigated for a long time (see among other [10, 3, 11]). This notion mostly concerns the issue that textual provisions of (legal) norms typically provide *prima facie* conditions for their applicability, but to understand a norm in full, we have to evaluate the norms in the context in which the norm is used and to see if other norms prevent it either to apply or to be effective. In other words, when evaluating norms, we must account for possible (*prima facie*) conflicts and exceptions. Indeed, in general, norms first provide the basic conditions for their applicability. Then, they give the exceptions and exclusions (and they can go on, with exceptions/exclusions of the exceptions/exclusions and so on).

The first issue to address to model legal reasoning is how to model norms. Here, we follow the approach of [12, 4] and stipulate that a norm is represented by an "IF $\cdots$ THEN $\ldots$" rule, where the IF part establishes the conditions of applicability of the norm and the THEN part specifies the legal effect of the norm. Where the legal effect of the norm is either that a proposition is taken to hold legally or that a legal requirement (obligation, prohibition, permission) is in force. Moreover, as we have alluded to, the norms are defeasible; thus, the IF/THEN conditional used to model legal norms does not correspond to the material implication of classical logic, and it has a non-monotonic nature. Several approaches have been proposed to reduce or compile the normative IF/THEN conditional. However, in general, as discussed by [13, 8], they suffer from some limitations; for example, the translation to classical propositional logic requires complete knowledge (for any atomic proposition, we have to determine whether it is true or not), it is not resilient to contradictions, and changes to the norms might require a complete rewriting of the translation.

In this work, we are going to examine how to provide an effective and constructive non-monotonic interpretation of (a restricted version of) L4 based on Answer Set Programming (ASP) meta-program. The meta-program gives the semantics of the underlying L4 constructs as well as a computational framework for them.

2 A QUICK OVERVIEW OF L4

L4 is a Domain Specific Language (DSL) under development in the context of the Centre for Computational Law[1] flagship project Research Programme in Computational Law, supported by the National Research Foundation of Singapore. The Programme's ultimate goal is to develop a Domain-Specific Language for expressing laws, contracts, and other rules. Current investigation of L4 has shown that the language is sufficiently precise to avoid ambiguities of natural languages and, at the same time, sufficiently close to a traditional law text with its characteristic elements such as cross-references,

[1] https://cclaw.smu.edu.sg/

ProLaLa 2023, January 15, 2023, Boston

prioritisation of rules and defeasible reasoning. Moreover, once a law has been coded in L4, it can be further processed for different tasks for applications involving some form of legal reasoning.

This paper focuses on one aspect of L4 (and in a restricted form). More specifically, we concentrate on the basic notion of a rule. A basic rule in L4 has the following form.

```
rule <r> if Preconditions then Conclusion
         {restrict: {subject to <s_1>,...,<s_n>} {despite <d_1>,...,<d_m>} }
```

where `r`, a unique identified, is the label (name) or the rule; `Preconditions` is a (possibly empty) conjunction of propositions, and, accordingly, we consider it as the set of the propositions; `Conclusion` is a single proposition. The propositions in a rule can be prefixed by one of the following expressions: `MUST`, `MAY`, `SHANT` indicating the deontic modifier (operator) that applies to the proposition. Finally, the keyword `restrict` specifies what rules are either stronger (`subject to`) or weaker `despite` than the current rule.

3 DEFEASIBILITY

Legal rules can be classified either as *constitutive rules* (also known as *counts as* rules) or *regulative rules*. In turn, a normative rule can either be a *prescriptive rule* or a *permissive rule*. A constitutive rule gives the meaning or defines a term; regulative rules specify what normative positions (obligations and prohibitions for prescriptive rules and permissions for permissive rule) and the conditions under which such normative positions hold. Defeasibility applies to both constitutive rules and regulative rules. Consider the following two real-life examples from Australian regulations and Acts.

Example 3.1 (Telecommunications Consumer Protections Code (C628:2012). Section 2.1. Definitions).

> **Complaint** means an expression of dissatisfaction made to a Supplier in relation to its Telecommunications Products or the complaints handling process itself, where a response or Resolution is explicitly or implicitly expected by the Consumer.
>
> An initial call to a provider to request a service or information or to request support is not necessarily a Complaint. An initial call to report a fault or service difficulty is not a Complaint. However, if a Customer advises that they want this initial call treated as a Complaint, the Supplier will also treat this initial call as a Complaint.

Example 3.2 (National Consumer Credit Protection Act 2009 (Act No. 134 of 2009). Section 29).

(1) A person must not engage in a credit activity if the person does not hold a licence authorising the person to engage in the credit activity.
(3) For the purposes of subsections (1) and (2), it is a defence if:
  (a) the person engages in the credit activity on behalf of another person (the principal); and
  (b) the person is:
    (i) an employee or director of the principal or of a related body corporate of the principal; or
    (ii) a credit representative of the principal; and ...

The semantics/computation for L4 rules we are going to present is based on Defeasible (Deontic) Logic [2, 7]. In Defeasible Logic, a proposition p holds (defeasibly) if:

- p is a fact; or

- there is a rule r such $Conc(r) = p$, and
  - for all $q \in Pre(r)$, q (defeasibly) holds (the rule is applicable), and
  - for any rule s such that $Conc(s) = \sim p$, s is either discarded or defeated

A rule s is *discarded* if there is proposition $q \in Pre(s)$ such that q is refuted, where refuted is the (constructive) failure to show that it holds. A rule s is *defeated* if there is an applicable rule t whose conclusion is the opposite of the conclusion of s and t is stronger than s.

The logic is sceptical in the sense that if there are two (applicable) rules for opposite conclusions and there is no means to solve the conflict, the logic prevents the conclusion of contradictions. Still, at the same time, it discards the conclusion of both conclusions. Hence, none of the two opposite conclusions holds. In other words, there is some ambiguity about which of the two conclusions hold. However, the opposite conclusions can be part of the preconditions of other rules. Consider, for example, the scenario where there are two equally compelling pieces of evidence, one supporting the case that a person was legally responsible for A and the second that the person was not responsible for A. Moreover, if the person was responsible for A, then the person is found guilty. However, according to the presumption of innocence, a person is assumed to be not guilty. This situation can be represented by

```
rule <r1> if evidence1 then responsible
rule <r2> if evidence2 then not responsible
rule <r3> if responsible then guilty
          {restrict: {despite <r4>}}
rule <r4> if true then not guilty
```

In this scenario, given the two pieces of evidence, we cannot assert whether `responsible` or `not responsible` holds; thus, the proposition `responsible` is ambiguous. A statement is ambiguous when there is an argument supporting it and an argument for its opposite; moreover, there is no way to determine if one of the two arguments is stronger/defeats the other. However, in this situation, we can go on since we cannot assert that responsible holds, but rule r_4 (encoding the so-called presumption of innocence) vacuously holds, and we can conclude `not guilty`.

However, Suppose that, in addition to the conditions stipulated above, if a person was wrongly accused, then the person is entitled to some compensation. This can be encoded in L4 by the following two rules:

```
rule <r5> if not guilty then compensation
rule <r6> if true then not compensation
          {restrict: {subject to <r5>}}
```

If we continue our reasoning, rule r_5 is applicable; it defeats r_6, allowing us to establish that `compensation` holds. However, the two pieces of evidence were equally reliable; it does not sound right that the person was wrongly accused. Indeed, it was ambiguous whether the accused was responsible or not. This scenario illustrates that we have to account for two forms of defeasibility: *ambiguity blocking* and *ambiguity propagation*. Governatori [5] argues that these two forms of defeasibility account for different (legal) proof standards. However, Defeasible Logic can accommodate the two variants.

The semantics we gave above is for the ambiguity blocking case. For the ambiguity propagating case, a few changes are needed [1]. First, a conclusion is *supported* if it is a fact or there is a rule such that all the preconditions are supported and the rule s not weaker than an applicable rule for the opposite. Second, rules attaching a conclusion are discarded if they are not supported (instead of applicable). These changes simplify attacking a conclusion, and it is easy to verify that both `guilty` and ‘`not guilty` are supported, and we prevent the conclusion of `compensation`.

4 DEFEASIBLE ENCODING OF L4 IS ANSWER SET PROGRAMMING

In this section, we give a meta-program in Answer Set Programming to encode the reasoning mechanism we presented in the previous section to model defeasibility with both ambiguity blocking and ambiguity propagation. The ASP meta-program clauses to capture ambiguity blocking and ambiguity propagation are based on Defeasible Logic variants and meta-program given in [1]. The deontic extension is based on the Defeasible Deontic Logic of [7] and the meta-program techniques of [9]. In addition to the meta-program clauses to model and compute the logic aspects, we discuss how to encode an L4 theory in the meta-program to compute the extension of the theory.

The first step for encoding Defeasible Deontic Logic in ASP is to provide the predicates and the clauses defining the language and the basic notions of the logic.

```
negation(non(X),X) :- atom(X).
negation(X,non(X)) :- atom(X).
```

The predicate `negation/2` takes two arguments (meant to correspond to the positive and negative literals for an atomic proposition, i.e., `atom/1` in the encoding parlance), and it establishes that `non(X)` and `X` are the negation of each other, and thus they cannot be true at the same time. It is worth noting that we use a meta-encoding; thus, we do not use the negations (classical and negation as failure) of ASP to represent the negation of literals in Defeasible Deontic Logic.

```
conflict(X,Y) :- strongConflict(X,Y).
conflict(Y,X) :- strongConflict(X,Y).
```

We need the following clauses to model the computation for ambiguity blocking aspect of defeasibility.

```
defeasible(X) :-  fact(X).
defeasible(X) :-  opposes(X,X1), not fact(X1), rule(R,X), applicable(R), not overruled(R,X).
overruled(R,X) :- opposes(X,X1), rule(R,X), rule(R1,X1), applicable(R1), not defeated(R1,X1).
defeated(R,X) :- opposes(X,X2), rule(R2,X2), superior(R2,R), applicable(R2).
```

The predicate `opposes/2` establishes that two propositions `X` and `Y` cannot hold simultaneously. The predicate `applicable/1` takes as its argument a rule in L4, and its truth is determined by the ASP encoding of the L4 rule (we discuss the full procedure of how to encode an L4 rule below).

For ambiguity propagation, we need the following clauses (notice that the clauses below offer an alternative version of the `defeasible` predicate to the definition given above).

```
support(X) :- fact(X).
support(X) :- rule(R,X), supported(R), not beaten(R,X).
beaten(R,X) :- rule(R,X), opposes(X,X1), fact(X1).
beaten(R,X) :- rule(R,X), opposes(X,X1), rule(R1,X1), supported(R1), superior(R1,R).
defeasible(X) :-  fact(X).
defeasible(X) :-  opposes(X,X1), not fact(X1), rule(R,X), applicable(R),  not overruled(R,X).
overruled(R,X) :- opposes(X,X1), rule(R,X), rule(R1,X1), supported(R1), not defeated(R1,X1).
defeated(R,X) :- opposes(X,X2), rule(R2,X2), superior(R2,R), applicable(R2).
support(X) :- defeasible(X).
supported(X) :- applicable(X).
```

Finally, to model the Defeasible Deontic Logic proposed in [7], we can use the following clauses (due to space reasons, we refer the readers to [7] for the description of the logic and its motivation. However, the predicate names are self-describing, and the clauses provide an alternative description of the logic).

```
deonticRule(R,X) :- prescriptiveRule(R,X).
deonticRule(R,X) :- permissiveRule(R,X).
obligation(X) :- prescriptiveRule(R,X), applicable(R), not deonticOverruled(R,X).
obligation(X) :- constitutiveRule(R,X), obligationApplicable(R), not deonticOverruled(R,X).
permission(X) :- permissiveRule(R,X), applicable(R), not deonticOverruled(R,X).
permission(X) :- constitutiveRule(R,X), permissionApplicable(R), not deonticOverruled(R,X).
deonticOverruled(R,X) :- prescriptiveRule(R,X), deonticRule(R1,X1), opposes(X,X1),
    applicable(R1), not deonticDefeated(R1,X1).
deonticOverruled(R,X) :- permissiveRule(R,X), prescriptiveRule(R1,X1), opposes(X,X1),
    applicable(R1), not deonticDefeated(R1,X1).
deonticDefeated(R,X) :- opposes(X,X1), prescriptiveRule(R,X), deonticRule(R1,X1),
    applicable(R1), superior(R1,R).
deonticDefeated(R,X) :- opposes(X,X1), permissiveRule(R,X), prescriptiveRule(R1,X1),
    applicable(R1), superior(R1,R).
```

The process of encoding an L4 rule in the meta-program has the following step. First, we rewrite each `SHANT p` as `MUST not p`. Then we group all propositions in `Preconditions` in three groups (the groups can be empty): the first group is the set of propositions that do not occurs in the scope of a deontic operator (`MUST`, `MAY`). The second and third groups are, respectively, the sets of propositions in the scope of ‘`MUST` and `MAY`. Thus a rule *r* has the generic form

```
rule <r>
    if a_1 && ... && a_n && MUST o_1 && ... && MUST o_m && MAY p_1 && ... && MAY p_k
    then [MUST|MAY] c
    {restrict: {subject to <s_1>...<s_l>} {despite <d_1>...<d_w>}}
```

A rule *r* is encoded as

```
prescriptiveRule(r,c). % if Conc(r) = MUST c
permissiveRule(r,c).   % if Conc(r) = MAY c
constitutiveRule(r,c). % otherwise

applicable(r) :-
    defeasible(a_1), ... , defeasible(a_n), % for each a_i not in the scope of MUST|MAY
    obligation(o_1), ... , obligation(o_m), % for each o_i in the scope of MUST
    permission(p_1), ... , permission(p_k). % for each p_i in the scope of MAY
```

finally, we add

```
superior(r,d). % for each d_i in restrict {despite <d_i>}
superior(s,r). % for each r_i in restrict {subject to <s_i>}
```

As we discussed, ambiguity propagation and ambiguity blocking intuition could be seen as different legal proof standards. A decision in a legal proceeding can use conclusions with different proof standards. The meta-programming

approach presented in this paper allows us to accommodate them. For example, instead of using `defeasible/1`, we can replace it with `defeasible/2` where the first argument is the type of defeasible conclusion (ambiguity blocking or ambiguity propagation) and the second is the proposition. In addition, the encoding of a rule can specify what type of defeasibility is required for a precondition in given rules. Thus we can have `defeasible(propagation,a_1), defeasible(blocking,a_2)` in one rule and `defeasible(blocking,a_1), defeasible(_,a_2)` in another rule. [6] proved that this combination is sound and complete and is a conservative extension of the individual variants. This shows that the defeasible deontic logic meta-programming encoding of L4 rules offers an efficient, flexible and powerful environment for modelling legal rules and a feasible and viable Rules as Code framework.

ACKNOWLEDGMENTS

This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. 2000. A flexible framework for defeasible logics. In *Proc. American National Conference on Artificial Intelligence (AAAI-2000)*. AAAI/MIT Press, Menlo Park, CA, 401–405. https://aaai.org/Papers/AAAI/2000/AAAI00-062.pdf.

[2] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. 2001. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2, 2, 255–287. DOI: 10.1145/371316.371517.

[3] Brian H. Bix. 2012. Defeasibility and Open Texture. In *The Logic of Legal Requirements: Essays on Defeasibility*. Oxford University Press, 193–201. DOI: 10.1093/acprof:oso/9780199661640.003.0011.

[4] Thomas F. Gordon, Guido Governatori, and Antonino Rotolo. 2009. Rules and norms: requirements for rule interchange languages in the legal domain. In *RuleML 2009* (LNCS) number 5858 (Las Vegas, Nevada, Nov. 5–7, 2009). Guido Governatori, John Hall, and Adrian Paschke, (Eds.) Springer, Heidelberg, 282–296. DOI: 10.1007/978-3-642-04985-9_26.

[5] Guido Governatori. 2011. On the relationship between Carneades and defeasible logic. In *The 13th International Conference on Artificial Intelligence and Law* (Pittsburgh, PA, USA, June 6–10, 2011). Kevin D. Ashley and Tom M. van Engers, (Eds.) ACM, 31–40. DOI: 10.1145/2018358.2018362.

[6] Guido Governatori and Michael J. Maher. 2017. Annotated defeasible logic. *Theory and Practice of Logic Programming*, 17, 5–6, 819–836. DOI: 10.1017/S1471068417000266.

[7] Guido Governatori, Francesco Olivieri, Antonino Rotolo, and Simone Scannapieco. 2013. Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, 42, 6, 799–829. DOI: 10.1007/s10992-013-9295-1.

[8] Guido Governatori, Vineet Padmanabhan, Antonino Rotolo, and Abdul Sattar. 2009. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17, 3, 227–265. DOI: 10.1093/jigpal/jzp006.

[9] Guido Governatori and Antonino Rotolo. 2004. Defeasible logic: agency, intention and obligation. Deontic logic in computer science. In *7th International Workshop on Deontic Logic in Computer Science* (LNAI) number 3065. Alessio Lomuscio and Donald Nute, (Eds.) Springer, Berlin, 114–128. DOI: 10.1007/978-3-540-25927-5_8.

[10] H. L. A. Hart. 1948. The ascription of responsibility and rights. *Proceedings of the Aristotelian Society*, 49, 171–194.

[11] Neil MacCormick. 1995. Defeasibility in law and logic. In *Informatics and the Foundations of Legal Reasoning*. Zenon Bankowski, Ian White, and Ulrike Hahn, (Eds.) Springer Netherlands, Dordrecht, 99–117. DOI: 10.1007/978-94-015-8531-6_3.

[12] Giovanni Sartor. 2005. *Legal Reasoning*. Springer, Dordrecht.

[13] Giovanni Sartor. 1992. Normative conflicts in legal reasoning. *Artificial Intelligence and Law*, 1, 209–235.