Computing Strong and Weak Permissions in Defeasible Logic

Guido GOVERNATORI [a], Francesco OLIVIERI [a,b,c]
Antonino ROTOLO [d] and Simone SCANNAPIECO [a,b,c]
[a] *NICTA, Queensland Research Laboratory, Australia*
[b] *Department of Computer Science, University of Verona, Italy*
[c] *Institute for Integrated and Intelligent Systems, Griffith University, Australia*
[d] *CIRSFID, University of Bologna, Italy*

Abstract In this paper we propose an extension of Defeasible Logic to represent and compute different concepts of defeasible permission. In particular, we discuss some types of explicit permissive norms that work as exceptions to opposite obligations or encode permissive rights. Moreover, we show how strong permissions can be represented both with, and without introducing a new consequence relation for inferring conclusions from explicit permissive norms. Finally, we illustrate how a preference operator applicable to contrary-to-duty obligations can be combined with a new operator representing ordered sequences of strong permissions. The logical system is studied from a computational standpoint and is shown to have linear computational complexity.

1. Introduction

The concept of permission plays an important role in many normative domains in that it may be crucial in characterising notions such as those of authorisation and derogation [8,25,27]. For example, consider when we subscribe to an on-line sale agreement accepting to enter our personal data on the condition that this information is only used for shipping, and other necessary purposes to communicate with us or deliver the products to us. Here, the permission to use our personal data is an exception to a general prohibition.

Despite this fact, the concept of permission is still elusive and has not been extensively investigated in deontic logic as the notion of obligation. For a long time, deontic logicians mostly viewed permission as the dual of obligation: $\mathsf{P}a \equiv \neg\mathsf{O}\neg a$. That is to say, a is permitted if $\neg a$ is not provable as mandatory. This view is unsatisfactory, as it hardly allows us to grasp the meaning of examples like the one previously mentioned. As a consequence, the attempt to reduce permissions to duals of obligations has been criticised (see [2,1]).

One important distinction that has traditionally contributed to a richer account of this concept is the one between *weak* (or *negative*) and *strong* (or *positive*) permission [28]. The former corresponds to saying that something is allowed by a code only when it is not prohibited by that code. At least when dealing with unconditional obligations, the notion of weak permission is trivially equivalent to the dual of obligation [24].

The concept of strong permission is more complicated, as it amounts to saying that some a is permitted by a code iff such a code explicitly states that a is permitted. It follows

Accepted on Journal of Philosophical Logic

that a strong permission is not derived from the absence of a prohibition, but is explicitly formulated in a permissive norm. The complexities of this concept depend on the fact that, besides "the items that a code explicitly pronounces to be permitted, there are others that in some sense follow from the explicit ones". The problem is hence "to clarify the inference from one to the other" [24, p. 391–2]. For example, if some b logically follows from a, which is strongly permitted, is b strongly permitted as well?

Features such as the distinction between strong and weak permission show the multi-faceted nature of permission and permissive norms, which has been overlooked by most logicians for a long time, even though a new interest has emerged in the last few years [24,6,7,9,27,26].

This paper assumes to work on two types of permission that emerge especially in the *legal* domain. *Firstly*, explicit permissive norms often state exceptions to obligations [5]: derogating with a permission, for example, to a general prohibition to use private protected personal data provides an exception to such a prohibition. Hence, the distinction between weak and strong permission is needed, otherwise it is rather hard to account for the fact that certain permissions explicitly derogate to *existing* prohibitions while other permissions are not explicit and occur *precisely* because opposite prohibitions *do not exist*. *Secondly*, permissive norms are sometimes used in the law to express the so-called permissive rights, which are directed permissions aimed at satisfying an interest of the person being permitted [25]. Permissive rights do not necessarily correspond to derogations: for instance, U.S. Copyright Act states that the copyright owner has the permission to elect, at any time before final judgment is rendered, to recover, instead of actual damages and profits, an award of statutory permissions are not explicit, and occur *precisely* because opposite damages for all infringements involved in the action (see Example 2). While a complete model of the idea of permissive rights may require to devise a rich formal machinery, which labels deontic operators with agents and combines deontic and action logics (see the discussion in [25]), this idea relies on developing at least a suitable deontic treatment of such permissions.

This paper moves from the above perspective with the specific purpose of studying the different conceptual and computational aspects of weak and strong permission. More precisely, the present contribution works in the following directions:

Permissions and defeasibility. The concept of permission exhibits strong connections with the idea of defeasibility. Indeed, an example of strong permissions acting as exceptions to obligations is when permissions rebut the conclusions of incompatible prescriptive norms [24,6,12,27] or undercut them (i.e., challenge an inference rule of an argument supporting an opposite obligation [7]).

Permissions and preferences. Sometimes permissive rights and explicit derogations of (existing or possible) prohibitions may be ranked according to some preference orderings. For example, in the case of derogations, given any prescriptive norm prohibiting a, more exceptions to this norm can be stated and ranked in a certain preference sequence. An example in the law domain is when the lawmaker, imposing duties for citizens, establishes conditions to lessen the effect of violating such duties to different degrees, or exempt people to comply with the duties. We study these mechanisms and see that ordered sequences of strong or explicit permissions have interesting similarities with ordered sequences of contrary-to-duty obligations [13,14]. This is a specific novelty of our contribution, as such sequences regard permissions which are not necessarily incompatible with each other.

Permissions and computation. To the best of our knowledge, no work in deontic logic has extensively explored the computational complexity of reasoning about different types of permission. In this work, we attempt an analysis of the problem in the context of a modal extension of Defeasible Logic [4] (DL hereafter). Modal DL is a computationally efficient logical framework able to capture various aspects of non-monotonic and modal reasoning, as well as the defeasible character of permissive norms, and recently a possible-world semantics for it has been proposed [15]. We study how to compute weak and strong permissions with and without introducing a new non-monotonic consequence relation for permission.

The layout of the paper is as follows. Section 2 introduces different types of defeasible permission in DL. Section 3 presents the technical machinery and states coherency and consistency results. In Section 4 we develop the algorithmic means to state what is mandatory and what is permitted in a given theory, along with the corresponding computational results in Section 5. Section 6 discusses the system and illustrates how the logical framework presented in Section 3 is able to capture the different types of permission. Section 7 discusses some related work and provides a summary of the paper.

2. Types of defeasible permission

This section is meant to offer a brief and gentle introduction to our formal language and logic, and to discuss different types of permission and their relation with the concept of normative defeasibility. Moreover, we illustrate the idea of preference over permissive rights or permissions that explicitly derogate to prohibitions.

These aspects are formally handled in Section 3. The whole discussion of the computational aspects of permission in DL is postponed to Sections 4 and 5.

2.1. Informal presentation of the logic

Let us summarise the basic logical intuitions behind our framework.

1. Permissive and prescriptive norms are represented by defeasible rules, whose conclusions normally follow unless defeated by contrary evidence. For example, the rule

   $$Order \Rightarrow_{\mathsf{O}} Pay$$

   says that, if we send a purchase order, then we are defeasibly obliged to pay; the rule

   $$Order, Creditor \Rightarrow_{\mathsf{P}} \neg Pay$$

   states that if we send an order, in general we are not obliged to pay if we are creditors for the same amount.
2. Rules introduce modalities: if we have the rule $a \Rightarrow_{\mathsf{O}} b$ and a holds, then we obtain $\mathsf{O}b$. That is to say, in the scenario where conditions described by a hold, the obligation of doing b is active as well. The advantage is that explicitly deriving modal literals such as $\mathsf{O}b$ adds expressive power to the language, since $\mathsf{O}b$ may appear in the antecedent of other rules which, in turn, can be triggered.
3. Modal literals can only occur in the antecedent of rules. In other words, we do not admit nested modalities, i.e., rules such as $a \Rightarrow_{\mathsf{O}} \mathsf{P}b$. This is in line with our idea that the applicability of rules labeled with mode $\Box$ (where $\Box$ can be O for obligation, or P for permission) is the condition for deriving literals modalised with $\Box$.

4. The symbols O and P are not simple labels: they are modalities. O is non-reflexive[1]: consequently, we do not have a conflict within the theory when $\neg a$ is the case and we derive that a is mandatory ($\mathsf{O}a$); this amounts to having a violation. The modality P works in such a way that two rules for P supporting a and $\neg a$ do not clash, but a rule like $\Rightarrow_{\mathsf{P}} b$ attacks a rule such as $\Rightarrow_{\mathsf{O}} \neg b$ and vice versa.
5. Like standard DL, our extension is able to establish the relative strength of any rule (thus to solve rule conflicts) and has two types of attackable rules: defeasible rules and defeaters. Defeaters in DL are a special kind of rules; they are used to prevent conclusions but not to support them. For example, the defeater

$$SpecialOrder, PremiumCustomer \rightsquigarrow_{\mathsf{O}} \neg PayBy7Days$$

can prevent the derivation of the obligation for premium customers placing special orders to pay within 7 days, but cannot be used to directly derive any conclusion.

2.2. Permissions and defeasibility

The above framework, though simple, allows us to express some basic types of permissions as well as illustrate interesting connections with the idea of defeasibility.

Weak permission. A first way to define permissions in DL is by simply considering weak permissions and stating that the opposite of what is permitted is not provable as obligatory. Let us consider a normative system consisting of the following two rules:

$$r_1 : Park, Vehicle \Rightarrow_{\mathsf{O}} \neg Enter$$
$$r_2 : Park, Emergency \Rightarrow_{\mathsf{O}} Enter.$$

Here, the normative system does not contain any permissive norm. However, since DL is a sceptical non-monotonic logic, in case both r_1 and r_2 fire we neither conclude that it is prohibited nor that it is obligatory to enter, because we do not know which rule is stronger. Consequently, in this context, both $\neg Enter$ and $Enter$ are weakly permitted.

As already argued, this is the most direct way to define the idea of weak permission: some q is permitted by a code iff q is not prohibited by that code. Accordingly, saying that any literal q is weakly permitted corresponds to the failure of deriving $\neg q$ using rules for O.

Explicit permissions are defeaters. In DL any rule can be used to prevent the derivation of a conclusion. For instance, suppose there exists a norm that prohibits to U-turn at traffic lights unless there is a "U-turn permitted" sign:

$$r_1 : AtTrafficLight \Rightarrow_{\mathsf{O}} \neg Uturn$$
$$r_2 : AtTrafficLight, UturnSign \Rightarrow_{\mathsf{O}} Uturn.$$

We use a defeasible rule for obligation to block the prohibition to U-turn. However, this is not satisfactory: if we do not know whether r_2 is stronger than r_1, then the best we can say is that U-turn is weakly permitted. Furthermore, if r_2 prevails over r_1, we derive that U-turn is obligatory.

Thus, when permissions derogate to prohibitions, there are good reasons to argue that defeaters for O are suitable to express an idea of strong permission[2]. Explicit rules such as $r : a \rightsquigarrow_{\mathsf{O}} q$ state that a is a specific reason for blocking the derivation of $\mathsf{O}\neg q$ (but not for

[1] As it is well-known, in a non-reflexive modal logic, $\Box a$ does not imply a, where $\Box$ is a modal operator.
[2] The idea of using defeaters to introduce permissions was introduced in [16].

proving $\mathsf{O}q$), i.e., this rule does not support any conclusion, but states that $\neg q$ is deontically undesirable. Consider this example:

$$r_1 : \mathit{Weekend}, \mathit{AirPollution} \Rightarrow_{\mathsf{O}} \neg \mathit{UseCar}$$
$$r_2 : \mathit{Weekend}, \mathit{Emergency} \rightsquigarrow_{\mathsf{O}} \mathit{UseCar}.$$

Rule r_1 states that on weekends it is forbidden to use private cars if a certain air pollution level is exceeded. Defeater r_2 is in fact an exception to r_1, and so it seems to capture the above idea that explicit permissive norms provide exceptions to obligations.

Explicit permissions using permissive rules. Another approach is based on introducing specific rules for deriving permissions [24,6]. Let us consider the following situation:

$$r_1 : \mathit{Weekend}, \mathit{AirPollution} \Rightarrow_{\mathsf{O}} \neg \mathit{UseCar}$$
$$r_2' : \mathit{Emergency} \Rightarrow_{\mathsf{P}} \mathit{UseCar}.$$

As r_2 in the previous scenario, r_2' looks like an exception to r_1. The apparent difference between r_2 and r_2' is that the latter is directly used to prove that the use of the car is permitted ($\mathsf{P}\mathit{UseCar}$) in case of emergencies. Does it amount to a real difference?

Although r_2 is a defeater, it is specifically used to derive the strong permission to use the car, like r_2'. In addition, rules such as r_2' do not attack other permissive rules, but are in conflict only with rules for obligation intended to prove the opposite conclusion. This precisely holds for defeaters.

Moreover, let us suppose to have the defeater $s : a \rightsquigarrow_{\mathsf{P}} b$. Does s attack a rule like $\Rightarrow_{\mathsf{P}} \neg b$? If this is the case, s would be close to an obligation. The fact that $\mathsf{P}b$ does not attack $\mathsf{P}\neg b$ makes it pointless for s to introduce defeaters for P. But, if this is not the case, s could only attack $\Rightarrow_{\mathsf{O}} \neg b$, thus being equivalent to $s' : a \rightsquigarrow_{\mathsf{O}} b$. although it is admissible to have defeaters, we do not need to distinguish defeaters for O from those for P.

We see two arguments to differentiate the approaches that model permissions via $\rightsquigarrow$ and via $\Rightarrow_{\mathsf{P}}$. The first argument is purely conceptual in that defeaters act only to block opposite conclusions. Hence, they are suitable for modeling only derogations but not permissive rights, which are incompatible with opposite obligations but are not primarily designed to be exceptions to prohibitions (see Example 2). Another way to mark the difference between $\rightsquigarrow$ and $\Rightarrow_{\mathsf{P}}$ is by stating that only the latter rule type admits ordered sequences of strong permissions in the head of a rule, which are either permissive rights or derogations to prohibitions. This second matter will be discussed in the next subsection.

2.3. Permissions, obligations, and preferences

Ordered sequences of strong permissions in the head of a rule can be logically modelled by enriching the formal language and following these guidelines:

1. In many domains, such as the law, norms often specify mandatory actions to be taken in case of their violation. In general, obligations in force after the violation of some other obligations correspond to contrary-to-duty (CTD) obligations. These constructions affect the formal characterisation of compliance since they identify situations that are not ideal, but still acceptable. A compact representation of CTDs may resort to the non-boolean connective $\otimes$ [13]: a formula like $x \Rightarrow_{\mathsf{O}} a \otimes b$ means that if x is the case, then a is obligatory, but if the obligation a is not fulfilled, then the obligation b is activated and becomes in force until it is satisfied, or violated.

2. Concepts introduced at point 1 can be extended to permissive rules with the subscripted arrow $\Rightarrow_{\mathsf{P}}$ by introducing the non-boolean connective $\odot$ for sequences of permissions. As in the case of $\otimes$, given a rule like $\Rightarrow_{\mathsf{P}} a \odot b$, we can proceed through the $\odot$-chain to obtain the derivation of $\mathsf{P}b$. However, permissions cannot be violated, and consequently it does not make sense to obtain $\mathsf{P}b$ from $\Rightarrow_{\mathsf{P}} a \odot b$ and $\neg a$. In this case, the reason to proceed in the chain is rather that the normative system allows us to prove $\mathsf{O}\neg a$. Hence, $\odot$ still establishes a preference order among strong permissions and, in case the opposite obligation is in force, another permission holds. This is significant especially when strong permissions are permissive rights or exceptions to obligations.

In this paper we take a neutral approach as to whether ordered sequences of obligations or permissions are either given explicitly, or inferred from other rules. However, we point out that, in domains such as the law, normative documents often explicitly contain provision with such structures. A clear example is provided by the Australian "National Consumer Credit Protection Act 2009" (Act No. 134 of 2009) which is structured in such a way that for every section establishing an obligation or a prohibition, the penalties for violating the provision are given in the section itself.

Example 1 (National Consumer Credit Protection Act 2009)**.** *Section 29 (Prohibition on engaging in credit activities without a licence) of the act recites:*

> *(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.*
> *Civil penalty: 2,000 penalty units.*
> *[…]*
> *Criminal penalty: 200 penalty units, or 2 years imprisonment, or both.*

This norm can be represented as

$$r_1 : \Rightarrow_{\mathsf{O}} \neg \mathit{CreditActivity} \otimes \mathit{2000CivilPenaltyUnits}$$
$$r_2 : \mathit{CreditLicence} \Rightarrow_{\mathsf{P}} \mathit{CreditActivity}$$

where $r_2 > r_1$. The first rule states that, in absence of other information, a person is forbidden to engage in credit activities ($\mathsf{O}\neg$CreditActivity), and then the second rule establish an exception to the prohibition, or in other terms, it recites a condition under which such activities are permitted. The section continues by giving explicit exceptions (permissions) to the prohibition to engage in credit activity, even without a valid licence.

Remark 1. This kind of structure has been successfully used for applications in the area of business process compliance [18]. In a situation governed by the rule $\Rightarrow_{\mathsf{O}} a \otimes b$ and where $\neg a$ and b hold, the norm has been complied with (even if to a lower degree than if we had a). On the contrary, if we had two rules $\Rightarrow_{\mathsf{O}} a$ and $\neg a \Rightarrow_{\mathsf{O}} b$, then the first norm would have been violated, while the second would have been complied with. But the whole case would be not compliant [17]. Consider the following example:

$$r_1 : \mathit{Invoice} \Rightarrow_{\mathsf{O}} \mathit{PayWithin7days}$$
$$r_2 : \mathsf{O}\mathit{PayWithin7days}, \neg \mathit{PayWithin7days} \Rightarrow_{\mathsf{O}} \mathit{Pay5\%Interest}$$
$$r_3 : \mathsf{O}\mathit{Pay5\%Interest}, \neg \mathit{Pay5\%Interest} \Rightarrow_{\mathsf{O}} \mathit{Pay10\%Interest}.$$

What happens if a customer violates both the obligation to pay within 7 days after the invoice and the obligation to pay the 5% of interest, but she pays the total amount plus the 10% of interest? In the legal perspective the customer should be still compliant, but in this

representation, contract clauses r_1 and r_2 have been violated. However, if we represent the whole scenario with the single rule

$$Invoice \Rightarrow_{\mathsf{O}} PayBy7days \otimes Pay5\%Interest \otimes Pay10\%Interest,$$

then the rule is not violated, and the customer is compliant with the contract.

Section 29 shows that there are cases where the textual provisions of norms themselves suggest to represent the norms using sequences of obligations. However, there are other cases, witnessed by the invoice scenario above, where contrary-to-duties, e.g., sequences of obligations and their penalties, are stated in different norms. The framework proposed in this paper is agnostic about the format in which norms are modelled. It offers syntactic structures and reasoning mechanisms suitable to handle both cases. It is up to a legal knowledge engineer to decide which format is the most suitable for the needs of the application at hand. [13] presents a sequent-style calculus to obtain rules with sequences of obligations (for example, to be used in a compliance application) from rules without them.

Sequences of permissions are a natural fit for expressions like "the subject is authorised, in order of preference, to do the following: (list)" or "the subject is entitled, in order of preference, to one of the following: (list)". This is illustrated in the next example, which offers a case of permissive right[3]:

Example 2 (U.S. Copyright Act)**.** *A concrete instance of sequences of permissions is given by Section 504(c)(1) (Remedies for infringement: Damages and profits) of the U.S. Copyright Act (17 USC §504).*

> *Except as provided by clause (2) of this subsection, the copyright owner may elect, at any time before final judgment is rendered, to recover, instead of actual damages and profits, an award of statutory damages for all infringements involved in the action, with respect to any one work, for which any one infringer is liable individually, or for which any two or more infringers are liable jointly and severally, in a sum of not less than $750 or more than $30,000 as the court considers just. [. . .]*

The above provision can be modelled as

$$Infringment, BeforeJudgment \Rightarrow_{\mathsf{P}} ActualDamages \odot StatutoryDamages$$

The above rendering of the textual provision is based on the interpretation of the term 'instead', which suggests that the copyright owners are entitled by default the award of the actual damages and profits, but they may elect to recover statutory damages, which is then the second option if exercised by the relevant party.

We conclude this section with an example showing a case where a sequence of permissions is used to derogate other obligations and prohibitions.

Example 3 (Italian Law 68/1999)**.** *Consider Article 4, Comma 4 of the Law 68/1999 (Right to work for people with disabilities).*

[3]Hence, we speak here of *entitlements* or *rights*, as corresponding to options for exercising the same general permissive right to compensation. In this perspective, we can model them as permissions on one party (in this case the copyright owner) generating an obligation on another party (in this case the infringer). This is in line with the classic conception of rights proposed, for instance, in [19], which does not properly view them as powers: a power is typically required there to generate further normative effects (such as duties, juridical relations, etc.). For a more detailed discussion on these issues, see [25].

> *For [...] workers [who suffered minor disabilities as a result of injury or illness, when complying with their duties] the injury or illness does not allow for justified dismissal in case they can be employed for, and assigned to equivalent or job duties and tasks or, when this is not applicable, even to lower job tasks. In the case of allocation to lower tasks they have the right to retain the more favorable treatment corresponding to the original tasks. If any such employees cannot be assigned to equivalent or lower tasks, they are* ex officio *relocated, by the competent authorities [...], in other companies and assigned to activities compatible with their work capacity [...].*

This provision prohibits the termination of employment for workers suffering from a work related injury resulting in a minor disability. In case the employees are no longer able to carry out their normal job function the comma permits to assign them to a different job position[4]*, with the option to terminate the employment if suitable job positions are not available. In addition it prescribes mechanisms to relocate workers whose employment has been terminated according to the previous condition.*

The comma can be formalised as follows:

$$r_1\colon \mathit{Injury}, \mathit{MinorDisability} \Rightarrow_{\mathsf{O}} \neg \mathit{TerminateEmployment}$$
$$r_2\colon \mathit{Injury}, \mathit{MinorDisability}, \neg \mathit{CurrentJob} \Rightarrow_{\mathsf{P}} \mathit{ChangeJob} \odot \mathit{TerminateEmployment}$$
$$r_3\colon \mathit{Injury}, \mathit{MinorDisability}, \mathit{TerminateEmployment} \Rightarrow_{\mathsf{O}} \mathit{Relocation}$$

where $r_2 > r_1$.

In case there is an available alternative position, the employer cannot terminate the employment, and has to assign the available position to the injured employee.

Suppose, now that the alternative (equivalent or lower) job position requires a licence, meaning that it is forbidden to perform the activities required without a licence:

$$r_4\colon \neg \mathit{License} \Rightarrow_{\mathsf{O}} \neg \mathit{ChangeJob}$$

with $r_4 > r_2$. *If the employee does not posses the required licence, then the employer can terminate the employment and has to refer the employee to the relevant authority for relocation.*

3. Defeasible Deontic Logic with strong permission

This section introduces the language adopted to formalise obligations and strong permissions in DL, and describes the inferential mechanism in the form of proof conditions defining the logic. Finally, we show that the proposed formalisation enjoys properties appropriate to model the notion of strong permission. We will proceed incrementally: this section, as well as Section 4, works only with obligations and strong permissions expressed by rules for P. In Section 6 we will show how weak permissions and strong permissions based on defeaters can be easily captured in the framework.

We consider a logic whose language is defined as follows.

Definition 1. *Let* PROP *be a set of propositional atoms,* MOD = $\{\mathsf{O}, \mathsf{P}\}$ *the set of modal operators where* O *is the modality for the obligation and* P *for permission.*

- *The set* Lit = PROP $\cup \{\neg p \mid p \in \text{PROP}\}$ *denotes the set of* literals.
- *The* complementary *of a literal* q *is denoted by* $\sim q$*; if* q *is a positive literal* p*, then* $\sim q$ *is* $\neg p$*, and if* q *is a negative literal* $\neg p$*, then* $\sim q$ *is* p.
- *The set of* modal literals *is* ModLit = $\{\Box l, \neg \Box l \mid l \in \text{Lit}, \Box \in \text{MOD}\}$.

[4] This in general might be prohibited based on contractual conditions of by other laws.

We introduce two preference operators, $\otimes$ for obligations and $\odot$ for permissions, and use $\oslash$ when we refer to one of them generically. These operators are used to build chains of preferences, called $\oslash$-expressions. The formation rules for well-formed $\oslash$-expressions are:

(a) every literal $l \in \text{Lit}$ is an $\oslash$-expression;
(b) if A is an $\otimes$-expression, B is an $\odot$-expression and $c_1, \ldots, c_k \in \text{Lit}$, then $A \otimes c_1 \otimes \cdots \otimes c_k$ is an $\otimes$-expression, $B \odot c_1 \odot \cdots \odot c_k$ is an $\odot$-expression, $A \odot B$ is an $\oslash$-expression;
(c) every $\otimes$-expression and $\odot$-expression is an $\oslash$-expression;
(d) nothing else is an $\oslash$-expression.

In addition we stipulate that $\otimes$ and $\odot$ obey the following properties:

1. $a \oslash (b \oslash c) = (a \oslash b) \oslash c$ (associativity);
2. $\bigoslash_{i=1}^{n} a_i = (\bigoslash_{i=1}^{k-1} a_i) \oslash (\bigoslash_{i=k+1}^{n} a_i)$ where there exists j such that $a_j = a_k$ and $j < k$ (duplication and contraction on the right).

Given an $\oslash$-expression A, the *length* of A is the number of literals in it. Given an $\oslash$-expression $A \oslash b \oslash C$ (where A and C can be empty), the *index* of b is the length of $A \oslash b$. We also say that b appears at index n in $A \oslash b$ if the length of $A \oslash b$ is n.

We adopt the standard DL definitions of *strict rules*, *defeasible rules*, and *defeaters* [4]. However for the sake of simplicity, and to better focus on the non-monotonic aspects that DL offers, in the remainder we use only defeasible rules and defeaters. In addition, we have to take the modal operators into account.

Definition 2. *Let* Lab *be a set of arbitrary labels. Every rule is of the type*

$$r : A(r) \hookrightarrow C(r)$$

where

1. $r \in \text{Lab}$ *is the name of the rule;*
2. $A(r) = \{a_1, \ldots, a_n\}$*, the* antecedent *(or* body*) of the rule, is the set of the premises of the rule (alternatively, it can be understood as the conjunction of all the literals in it). Each a_i is either a literal, or a modal literal;*
3. $\hookrightarrow \in \{\Rightarrow_\Box, \rightsquigarrow\}$ *denotes the type of the rule. If $\hookrightarrow$ is $\Rightarrow_\Box$, the rule is a* defeasible rule*, while if $\hookrightarrow$ is $\rightsquigarrow$, the rule is a* defeater*. The subscript $\Box \in$ MOD in defeasible rules represents the modality introduced by the rule: the mode of a rule tells what kind of conclusion we obtain from the rule. As argued in Section 2, we do not need to label $\rightsquigarrow$ with any mode;*
4. $C(r)$ *is the* consequent *(or* head*) of the rule, which is an $\oslash$-expression such that: (a) if $\hookrightarrow$ is $\rightsquigarrow$, then $C(r)$ is a single literal; (b) if $\Box = \mathsf{P}$, then $C(r)$ must be an $\odot$-expression.*

Given a set of rules R, we use the following abbreviations for specific subsets of rules:

- R_{def} denotes the set of all defeaters in the set R;
- $R[q, n]$ is the set of rules where q appears at index n in the consequent. The set of (defeasible) rules where q appears at any index n is denoted by $R[q]$;
- $R^\Box$ with $\Box \in$ MOD denotes the set of all rules in R introducing modality $\Box$;
- $R^{\mathsf{O}}[q, n]$ is the set of (defeasible) rules where q appears at index n and the operator preceding it is $\otimes$ for $n > 1$ or the mode of the rule is O for $n = 1$. The set of (defeasible) rules where q appears at any index n is denoted by $R^{\mathsf{O}}[q]$;
- similarly $R^{\mathsf{P}}[q, n]$ is the set of rules where q appears at index n, and the operator preceding it is $\odot$ for $n > 1$ or the mode of the rule is P for $n = 1$. The set of (defeasible) rules where q appears at any index n is denoted by $R^{\mathsf{P}}[q]$.

Definition 3. *A* Defeasible Theory *is a structure* $D = (F, R, >)$*, where* F*, the set of facts, is a set of literals and modal literals,* R *is a set of rules and* $>$*, the superiority relation, is a binary relation over* R*.*

A theory corresponds to a normative system, i.e., a set of norms, where every norm is modelled by rules. The superiority relation is used for conflicting rules, i.e., rules whose conclusions are complementary literals, in case both rules fire. We do not impose any restriction on the superiority relation: it just determines the relative strength of two rules.

Definition 4. *A* proof P *in a defeasible theory* D *is a linear sequence* $P(1) \dots P(n)$ *of* tagged literals *in the form of* $+\partial_{\Box} q$ *and* $-\partial_{\Box} q$ *with* $\Box \in \mathrm{MOD}$*, where* $P(1) \dots P(n)$ *satisfy the proof conditions given in Definitions 9–12.*

The tagged literal $+\partial_{\Box} q$ means that q is *defeasibly provable* in D with modality $\Box$, while $-\partial_{\Box} q$ means that q is *defeasibly refuted* with modality $\Box$. The initial part of length i of a proof P is denoted by $P(1..i)$.

The first thing to do is to define when a rule is applicable or discarded. A rule is *applicable* for a literal q if q occurs in the head of the rule, all non-modal literals in the antecedent are given as facts and all the modal literals have been defeasibly proved (with the appropriate modalities). On the other hand, a rule is *discarded* if at least one of the modal literals in the antecedent has not been proved (or is not a fact in the case of non-modal literals). However, as literal q might not appear as the first element in an $\oslash$-expression in the head of the rule, some additional conditions on the consequent of rules must be satisfied. Defining when a rule is applicable or discarded is essential to characterise the notion of provability for obligations ($\pm\partial_{\mathrm{O}}$) and permissions ($\pm\partial_{\mathrm{P}}$).

Definition 5. *A rule* $r \in R[q, j]$ *is* body-applicable *iff for all* $a_i \in A(r)$*:*

1. *if* $a_i = \Box l$ *then* $+\partial_{\Box} l \in P(1..n)$ *with* $\Box \in \mathrm{MOD}$*;*
2. *if* $a_i = \neg \Box l$ *then* $-\partial_{\Box} l \in P(1..n)$ *with* $\Box \in \mathrm{MOD}$*;*
3. *if* $a_i = l \in \mathrm{Lit}$ *then* $l \in F$*.*

A rule $r \in R[q, j]$ *is* body-discarded *iff* $\exists a_i \in A(r)$ *such that*

1. *if* $a_i = \Box l$ *then* $-\partial_{\Box} l \in P(1..n)$ *with* $\Box \in \mathrm{MOD}$*;*
2. *if* $a_i = \neg \Box l$ *then* $+\partial_{\Box} l \in P(1..n)$ *with* $\Box \in \mathrm{MOD}$*;*
3. *if* $a_i = l \in \mathrm{Lit}$ *then* $l \notin F$*.*

Definition 6. *A rule* $r \in R[q, j]$ *such that* $C(r) = c_1 \otimes \cdots \otimes c_{l-1} \odot c_l \odot \cdots \odot c_n$ *is* applicable *for literal* q *at index* j*, with* $1 \leq j < l$*, in the condition for* $\pm\partial_{\mathrm{O}}$ *iff*

1. r *is body-applicable; and*
2. *for all* $c_k \in C(r)$*,* $1 \leq k < j$*,* $+\partial_{\mathrm{O}} c_k \in P(1..n)$ *and* ($c_k \notin F$ *or* $\sim c_k \in F$*).*

Conditions (1) represents the requirements on the antecedent stated in Definition 5; condition (2) on the head of the rule states that each element c_k prior to q must be derived as an obligation, and a violation of such obligation has occurred.

Definition 7. *A rule* $r \in R[q, j]$ *such that* $C(r) = c_1 \otimes \cdots \otimes c_{l-1} \odot c_l \odot \cdots \odot c_n$ *is* applicable *for literal* q *at index* j*, with* $l \leq j \leq n$ *in the condition for* $\pm\partial_{\mathrm{P}}$ *iff*

1. r *is body-applicable; and*
2. *for all* $c_k \in C(r)$*,* $1 \leq k < l$*,* $+\partial_{\mathrm{O}} c_k \in P(1..n)$ *and* ($c_k \notin F$ *or* $\sim c_k \in F$*); and*
3. *for all* $c_k \in C(r)$*,* $l \leq k < j$*,* $-\partial_{\mathrm{P}} c_k \in P(1..n)$*.*

The only difference with respect to $\pm\partial_{\mathrm{O}}$ is the presence of an additional condition, stating that all permissions prior to q must be refuted (condition (3)).

Definition 8. *A rule $r \in R[q, j]$ such that $C(r) = c_1 \otimes \cdots \otimes c_{l-1} \odot c_l \odot \cdots \odot c_n$ is* discarded *for literal q at index j, with $1 \leq j \leq n$ in the condition for $\pm\partial_{\mathsf{O}}$ or $\pm\partial_{\mathsf{P}}$ iff*

1. *r is body–discarded; or*
2. *there exists $c_k \in C(r)$, $1 \leq k < l$, such that either $-\partial_{\mathsf{O}} c_k \in P(1..n)$ or $c_k \in F$; or*
3. *there exists $c_k \in C(r)$, $l \leq k < j$, such that $+\partial_{\mathsf{P}} c_k \in P(1..n)$.*

In this case, condition (2) ensures that an obligation prior to q in the chain is not in force or has already been fulfilled (thus, no reparation is required), while condition (3) states that there exists at least one explicit derived permission prior to q.

We now introduce the proof conditions for $\pm\partial_{\mathsf{O}}$ and $\pm\partial_{\mathsf{P}}$.

Definition 9. *The proof condition of* defeasible provability for obligation *is*

$+\partial_{\mathsf{O}}$: If $P(n+1) = +\partial_{\mathsf{O}} q$ then
(1) $\mathsf{O}q \in F$ or
(2.1) $\mathsf{O}{\sim}q \notin F$ and $\neg\mathsf{O}q \notin F$ and $\mathsf{P}{\sim}q \notin F$ and
(2.2) $\exists r \in R^{\mathsf{O}}[q, i]$ such that r is applicable for q, and
(2.3) $\forall s \in R[{\sim}q, j]$, either
(2.3.1) s is discarded, or either
(2.3.2) $s \in R^{\mathsf{O}}$ and $\exists t \in R[q, k]$ such that t is applicable for q and $t > s$, or
(2.3.3) $s \in R^{\mathsf{P}} \cup R_{def}$ and $\exists t \in R^{\mathsf{O}}[q, k]$ such that t is applicable for q and $t > s$.

To show that q is defeasibly provable as an obligation, there are two ways: (1) the obligation of q is a fact, or (2) q must be derived by the rules of the theory. In the second case, three conditions must hold: (2.1) q does not appear as not obligatory as a fact, and ${\sim}q$ is neither provable as an obligation nor as a permission using the set of modal facts at hand; (2.2) there must be a rule introducing the obligation for q which can apply; (2.3) every rule s for ${\sim}q$ is either discarded or defeated by a stronger rule for q. If s is an obligation rule, then it can be counterattacked by any type of rule; if s is a defeater or a permission rule, then only an obligation rule can counterattack it.

The strong negation of Definition 9 gives the negative proof condition for obligation.

Definition 10. *The proof condition of* defeasible refutability for obligation *is*

$-\partial_{\mathsf{O}}$: If $P(n+1) = -\partial_{\mathsf{O}} q$ then
(1) $\mathsf{O}q \notin F$ and either
(2.1) $\mathsf{O}{\sim}q \in F$ or $\neg\mathsf{O}q \in F$ or $\mathsf{P}{\sim}q \in F$ or
(2.2) $\forall r \in R^{\mathsf{O}}[q, i]$ either r is discarded for q, or
(2.3) $\exists s \in R[{\sim}q, j]$ such that
(2.3.1) s is applicable for ${\sim}q$, and
(2.3.2) if $s \in R^{\mathsf{O}}$ then $\forall t \in R[q, k]$, either t is discarded or $t \not> s$, and
(2.3.3) if $s \in R^{\mathsf{P}} \cup R_{def}$ then $\forall t \in R^{\mathsf{O}}[q, k]$, either t is discarded or $t \not> s$.

We now introduce and briefly explain the proof conditions for permission.

Definition 11. *The proof condition of* defeasible provability for permission *is*

$+\partial_{\mathsf{P}}$: If $P(n+1) = +\partial_{\mathsf{P}} q$ then
(1) $\mathsf{P}q \in F$ or
(2.1) $\mathsf{O}{\sim}q \notin F$ and $\neg\mathsf{P}q \notin F$ and
(2.2) $\exists r \in R^{\mathsf{P}}[q, i]$ such that r is applicable for q, and
(2.3) $\forall s \in R^{\mathsf{O}}[{\sim}q, j]$, either
(2.3.1) s is discarded for ${\sim}q$, or
(2.3.2) $\exists t \in R[q, k]$ such that t is applicable for q and $t > s$.

This proof condition differs from its counterpart for obligation in two aspects: we allow scenarios where both $+\partial_{\mathsf{P}} q$ and $+\partial_{\mathsf{P}} {\sim} q$ hold, but $+\partial_{\mathsf{O}} {\sim} q$ must not hold (clause 2.1); any applicable rule s supporting ${\sim} q$ can be counterattacked by any type of rule t supporting q, as s must be an obligation rule, and permission rules can only be attacked by obligation rules (clause 2.3).

As above, the negative proof conditions for P are the strong negation of those for $+\partial_{\mathsf{P}}$.

Definition 12. *The proof condition of* defeasible refutability for permission *is*

$-\partial_{\mathsf{P}}$*: If* $P(n+1) = -\partial_{\mathsf{P}} q$ *then*
(1) $\mathsf{P} q \notin F$ *and either*
  (2.1) $\mathsf{O}{\sim} q \in F$ *or* $\neg \mathsf{P} q \in F$*, or*
  (2.2) $\forall r \in R^{\mathsf{P}}[q, i]$*, either* r *is discarded, or*
  (2.3) $\exists s \in R^{\mathsf{O}}[{\sim} q, j]$ *such that*
    (2.3.1) s *is applicable for* ${\sim} q$*, and*
    (2.3.2) $\forall t \in R[q, k]$*, either* t *is discarded or* $t \not> s$*.*

The logic resulting from the above proof conditions enjoys properties describing the appropriate behaviour of the modal operators.

Definition 13. *A Defeasible Theory* $D = (F, R, >)$ *is* consistent *iff* $>$ *is acyclic and* F *does not contain pairs of complementary (modal) literals, that is if* D *does not contain pairs like* $\mathsf{O} l$ *and* $\neg \mathsf{O} l$*,* $\mathsf{P} l$ *and* $\neg \mathsf{P} l$*, or* l *and* ${\sim} l$*. The theory* D *is* O-consistent *iff* $>$ *is acyclic and for any literal* l*,* F *does not contain any of the following pairs:* $\mathsf{O} l$ *and* $\mathsf{O}{\sim} l$*,* $\mathsf{O} l$ *and* $\mathsf{P}{\sim} l$*.*

As usual, given a Defeasible Theory D, we will use $D \vdash \pm\partial_{\Box} l$ iff there is a proof P in D such that $P(n) = \pm\partial_{\Box} l$ for an index n.

Proposition 14. *Let* D *be a consistent Defeasible Theory, and* $\Box \in$ MOD*. For any literal* l*, it is not possible to have both* $D \vdash +\partial_{\Box} l$ *and* $D \vdash -\partial_{\Box} l$*.*

Proof. It straightforwardly follows from the principle of strong negation proposed in [3,12]:[5] indeed, the negative proof tags proposed in this work are defined as the strong negation of the positive ones. □

The meaning of the above proposition is that it is not possible to prove that a literal is at the same time obligatory and not obligatory, or permitted and not permitted.

Proposition 15. *Let* D *be an* O*-consistent Defeasible Theory. For any literal* l*, it is not possible to have both* $D \vdash +\partial_{\mathsf{O}} l$ *and* $D \vdash +\partial_{\mathsf{O}} {\sim} l$*.*

Proof. We have two cases: (i) at least one of $\mathsf{O} l$ and $\mathsf{O}{\sim} l$ is in F; (ii) none of them is in F.

For (i) the proposition immediately follows by the assumption of O-consistency. Suppose that $\mathsf{O} l \in F$. Then clause (1) of $+\partial_{\mathsf{O}}$ holds for l. By O-consistency $\mathsf{O}{\sim} l \notin F$, thus clause (1) of $+\partial_{\mathsf{O}}$ does not hold for ${\sim} l$. Since $\mathsf{O} l \in F$, clause (2.1) of $+\partial_{\mathsf{O}}$ is always falsified for ${\sim} l$, and the thesis is proved.

For (ii): First of all, it is easy to verify that no rule can be at the same time applicable and discarded for the derivation of $\pm\partial_{\mathsf{O}} l({\sim} l)$. Then, since both $+\partial_{\mathsf{O}} l$ and $+\partial_{\mathsf{O}} {\sim} l$ hold, we have that there are applicable obligation rules for both l and ${\sim} l$. This means that clause (2.3.2) holds for both $+\partial_{\mathsf{O}} l$ and $+\partial_{\mathsf{O}} {\sim} l$. Therefore, for every applicable rule for l there is an

[5] The *strong negation* of a formula is closely related to the function that simplifies a formula by moving all negations to an inner most position in the resulting formula, and replaces the positive tags with respective negative tags, and vice versa.

applicable rule for $\sim l$ stronger than the rule for l, and symmetrically, for every applicable rule for $\sim l$ there is an applicable rule for l stronger than the rule for $\sim l$. Since the set of rules in a theory is finite, the situation we have just described is possible only if there is a cycle in the transitive closure of the superiority relation. Therefore, we have a contradiction because the superiority relation is assumed to be acyclic. □

The meaning of the proposition is that no formula is both obligatory and forbidden at the same time. However, the proposition does not hold for permission. It is possible to have both the explicit permission of l and the explicit permission of $\sim l$.

The relationships between permissions and obligations are governed by the following.

Proposition 16. *Let D be an O-consistent Defeasible Theory. For any literal l:*

1. *if $D \vdash +\partial_{\mathsf{O}} l$, then $D \vdash -\partial_{\mathsf{O}} \sim l$;*
2. *if $D \vdash +\partial_{\mathsf{O}} l$, then $D \vdash -\partial_{\mathsf{P}} \sim l$;*
3. *if $D \vdash +\partial_{\mathsf{P}} l$, then $D \vdash -\partial_{\mathsf{O}} \sim l$.*

Proof. 1. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_{\mathsf{O}} l$. Literal $\sim l$ can be in only one of the following mutually exclusive situations: (i) $D \vdash +\partial_{\mathsf{O}} \sim l$; (ii) $D \vdash -\partial_{\mathsf{O}} \sim l$; (iii) $D \not\vdash \pm\partial_{\mathsf{O}} \sim l$. Proposition 15 allows us to exclude case (i) since $D \vdash +\partial_{\mathsf{O}} l$ by hypothesis. Case (iii) denotes situations where there are loops in D involving literal $\sim l$,[6] but inevitably this would affect also the provability of literal l, i.e., we would not be able to give a proof for $+\partial_{\mathsf{O}} l$ as well. This contradicts the hypothesis; thus, situation (ii) must be the case.

2. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_{\mathsf{O}} l$. By definition of proof conditions for $+\partial_{\mathsf{O}}$, statements (1)–(2.3.3) hold.

If $\mathsf{O}l \in F$, then condition (2.1) of $-\partial_{\mathsf{P}}$ holds for $\sim l$. Furthermore, by O-consistency of D, $\mathsf{O}l \in F$ implies condition (1) of $-\partial_{\mathsf{P}}$ for $\sim l$, and we obtain the thesis.

Otherwise, from condition (2.2) of $+\partial_{\mathsf{O}}$, conditions (2.3) and (2.3.1) of $-\partial_{\mathsf{P}}$ follow. Again, we can iterate the same reasoning for condition (2.3.1) of $+\partial_{\mathsf{O}}$ implying condition (2.2) of $-\partial_{\mathsf{P}}$. It remains to consider conditions (2.3.2) and (2.3.3) of $+\partial_{\mathsf{O}}$. In the first case, the attacking rule s is an obligation rule, thus it is of no interest in this proof since in condition (2.2) of $-\partial_{\mathsf{P}}$ we consider only permission rules. Thus, for the latter case, we know there exists a rule t for obligation that is stronger than s, and this tuple of rules (t, s) in condition (2.3.3) for $+\partial_{\mathsf{O}}$ is the equivalent but opposite tuple of the rules used in condition (2.3.2) for $-\partial_{\mathsf{P}}$. But, to analyse in an exhaustive way condition (2.3.2) of $-\partial_{\mathsf{P}}$, we have to take into consideration the whole set of rules $S = \{s_1, \ldots, s_n\}$ for permission leading to $\sim l$, against the set of rules $T = \{t_1, \ldots, t_m\}$ for obligation leading to l.[7] For each subset S' of S, every rule $s' \in S'$ is either discarded, or there exists a rule $t' \in T$ that is stronger (the existence of t' is guaranteed by the fact that $+\partial_{\mathsf{O}} l$ holds by hypothesis). We can now remove S' from S (as it cannot be used for proving $+\partial_{\mathsf{P}} \sim l$), reducing the number of elements in S. The iteration of this procedure eventually empties the set S since the number of rules in D is finite, and the superiority relation is acyclic.

3. Let D be an O-consistent Defeasible Theory, and $D \vdash +\partial_{\mathsf{P}} l$. By definition of proof conditions for $+\partial_{\mathsf{P}}$, statements (1) or (2.1)–(2.3.2) hold. The proof follows step by step that of Part 2 of the proposition. Moreover, steps for conditions (1)–(2.2) are the mere juxtaposition. It remains to analyse the interrelationship between condition (2.3) of $+\partial_{\mathsf{P}}$ and conditions (2.3.2)–(2.3.3) of $-\partial_{\mathsf{O}}$. Since condition (2.3) of $+\partial_{\mathsf{P}}$ takes into account only

[6] Situations like $\mathsf{O}\sim l \Rightarrow_{\mathsf{O}} \sim l$, where the proof conditions will generate a loop without introducing a proof.

[7] Notice that the rules in conditions (2.2) and (2.3.3) are different rules: they form a team that can defeat teams of rules for the opposite.

rules for obligation which are systematically defeated with an analogous process of rule elimination, thus conditions (2.3.2) and (2.3.3) of $-\partial_{\mathsf{O}}$ are satisfied. □

The combination of Part 2 and 3 of Proposition 16 describes the consistency between obligation and permission, while Part 3 gives the relationships between strong and weak permission. As discussed in Section 1, a weak permission is a permission obtained from the failure to derive the opposite obligation. Consequently, we have the weak permission of p when $-\partial_{\mathsf{O}}{\sim}p$ holds and Part 3 guarantees $+\partial_{\mathsf{P}}p$ to hold as well.

4. Algorithms for defeasible extension

We now exhibit procedures and algorithms apt to compute the *extension* of a *finite* Defeasible Theory[8], in order to bound the complexity of the logic presented. The algorithms are based on the algorithm proposed in [21]; the algorithms also incorporate the notion of inferiorly defeated rules proposed by [20] to handle directly the superiority relation.

The present section is divided in three main parts. The first part gives the formal definitions and introduces the notation adopted. The second part describes the required computations: where Algorithms 1, 2, and 3 are but auxiliary procedures, Algorithms 5 and 4 effectively compute the defeasible extension of a Defeasible Theory given as an input. Each algorithm is followed by a technical explanation. In the third part we present formal properties that are meant to prove the computational results proposed in Section 5.

4.1. Notation for the algorithms

Given a Defeasible Theory D, HB_D is the set of literals such that the literal or its complement appears in D, where 'appears' means that it is a sub-formula of a modal literal occurring in the theory. The modal Herbrand Base of D is $HB = \{\Box l | \Box \in \mathrm{MOD}, l \in HB_D\}$. Accordingly, the extension of a Defeasible Theory is defined as follows.

Definition 17. *Given a Defeasible Theory D, the* defeasible extension *of D is defined as*

$$E(D) = (+\partial_{\mathsf{O}}, +\partial_{\mathsf{P}}, -\partial_{\mathsf{O}}, -\partial_{\mathsf{P}}),$$

where $\pm\partial_{\Box} = \{l \in HB_D : D \vdash \pm\partial_{\Box} l\}$ *with* $\Box \in \mathrm{MOD}$. *We define two Defeasible Theories D and* D' *to be* equivalent *(in notation* $D \equiv D'$*) if they have the same extensions, i.e.,* $E(D) = E(D')$.

The next definition introduces two syntactical operations on the consequent of rules.

Definition 18. *Let* $c_1 = a_1 \oslash \cdots \oslash a_{l-1}$ *and* $c_2 = a_{i+1} \oslash \cdots \oslash a_n$ *be two (possibly empty)* $\oslash$*-expressions such that* a_i *does not occur in them, and* $c = c_1 \oslash a_i \oslash c_2$ *is an* $\oslash$*-expression. Let r be a rule with form* $A(r) \hookrightarrow c$. *The operation of* truncation *of the consequent c at* a_i *is:*

$$A(r) \hookrightarrow c!a_i = A(r) \hookrightarrow c_1 \oslash a_i.$$

We define the removal *of* a_i *from the consequent c,* $A(r) \hookrightarrow c \ominus a_i$, *as:*

$$A(r) \hookrightarrow c \ominus a_i = \begin{cases} A(r) \Rightarrow_{\mathsf{O}} c_1 \otimes c_2 & \text{if } r \text{ is } A(r) \Rightarrow_{\mathsf{O}} c_1 \otimes a_i \otimes c_2 \\ A(r) \Rightarrow_{\mathsf{O}} c_1 \odot c_2 & \text{if } r \text{ is } A(r) \Rightarrow_{\mathsf{O}} c_1 \otimes a_i \odot c_2 \\ A(r) \Rightarrow_{\Box \in \mathrm{MOD}} c_1 \odot c_2 & \text{if } r \text{ is } A(r) \Rightarrow_{\Box \in \mathrm{MOD}} c_1 \odot a_i \odot c_2 \\ A(r) \Rightarrow_{\mathsf{P}} c_2 & \text{if } r \text{ is } A(r) \Rightarrow_{\mathsf{O}} a_i \odot c_2. \end{cases}$$

[8] A defeasible theory is *finite* when the sets of facts and rules are finite.

The next definition extends the concept of complement presented in Section 3 for modal literals and establishes the logical connection among proved and refuted literals.

Definition 19. *We define the* complement *of a given literal l, denoted by $\tilde{l}$, as:*

1. *If $l \in \mathrm{Lit}$, then $\tilde{l} = \{\sim l\}$;*
2. *If $l = \mathsf{O}m$, then $\tilde{l} = \{\neg\mathsf{O}m, \mathsf{O}{\sim}m, \mathsf{P}{\sim}m\}$;*
3. *If $l = \neg\mathsf{O}m$, then $\tilde{l} = \{\mathsf{O}m\}$;*
4. *If $l = \mathsf{P}m$, then $\tilde{l} = \{\neg\mathsf{P}m, \mathsf{O}{\sim}m\}$;*
5. *If $l = \neg\mathsf{P}m$, then $\tilde{l} = \{\mathsf{P}m\}$.*

Given $\Box \in \mathrm{MOD}$, the sets $\pm\partial_\Box$ denote the global sets of defeasible conclusions, while $\partial^{\pm}_\Box$ are the corresponding temporary sets. Notice that the complement of $\neg\mathsf{P}m$ does not include $\mathsf{O}m$ (and vice versa) because the failure to derive $\mathsf{P}m$ cannot depend on the derivation of $\mathsf{O}m$, but rather on the fact that $\mathsf{O}{\sim}m$ is the case.

4.2. Algorithms

We begin this subsection by reporting and explaining the three auxiliary procedures used in the two main algorithms for the computation of the extension of a logic.

Algorithm 1 Discard

1: **procedure** Discard($l \in \mathrm{Lit}, \Box \in \mathrm{MOD}$)
2: $\quad\partial^-_\Box \leftarrow \partial^-_\Box \cup \{l\}$
3: $\quad R \leftarrow \{A(r) \setminus \{\neg\Box l\} \hookrightarrow C(r) \mid r \in R\} \setminus \{r \in R \mid \Box l \in A(r)\}$
4: $\quad >\!\!\leftarrow\!\!> \setminus \{(r,s), (s,r) \in >\!\!\leftarrow\!\!> \mid \Box l \in A(r)\}$
5: $\quad HB \leftarrow HB \setminus \{\Box l\}$
6: **end procedure**

Algorithm 1 Discard is invoked when $-\partial_\Box l$ holds for a given literal l. First of all, literal l is placed in the local set of refuted literals with modality $\Box$ (line 2). Furthermore, condition $-\partial_\Box l$ makes literal $\neg\Box l$ provable, therefore it can be safely removed from all rules where it appears as an antecedent, being that its contribution to a rule being applicable or refuted has already been established. Since l cannot be proved with modality $\Box$, every rule containing $\Box l$ in its body is discarded by clauses (1.1) and (1.3) of Definition 8, and thus we can remove such rules without affecting the conclusions of the theory (line 3). The superiority relation and the modal Herbrand Base are updated accordingly (line 4–5).

Algorithms 2 ModifyObl and 3 ModifyPerm behave in a very similar way: both of them modify the theory to accommodate the positive derivation of a modal literal. They only differ on the kind of rules they manipulate (resp. obligation and permission rules).

The input of both procedures is a literal l. As such, we add it to the corresponding set of derived literals (line 2). Since $D \vdash +\partial_\mathsf{O} l$ implies $D \vdash -\partial_\mathsf{O}{\sim}l, -\partial_\mathsf{P}{\sim}l$ by Proposition 16 Part 1 and 2, and $D \vdash +\partial_\mathsf{P} l$ implies $D \vdash -\partial_\mathsf{O}{\sim}l$ by Proposition 16 Part 3, we also remove $\sim l$ from the appropriate sets of refuted literals; then the modal literal along with the set of its complementaries[9] are removed from the Modal Herbrand Base (lines 3–4). Lines 5–8 follow the same reasoning of line 3 in Algorithm 1 Discard. Finally, the rules of the theory are modified on account of the modality the literal is derived with, as well as the applicability conditions given in Definitions 6–8 (resp. lines 9–11 and 9–10).

[9] Notice that we do not remove any negative modal literal from HB by definition of modal Herbrand Base.

Algorithm 2 ModifyObl

procedure ModifyObl($l \in$ Lit)
  $\partial^+_{\mathsf{O}} \leftarrow \partial^+_{\mathsf{O}} \cup \{l\}$
  $\partial^-_{\Box} \leftarrow \partial^-_{\Box} \cup \{\sim l\}$, with $\Box \in$ MOD
  $HB \leftarrow HB \setminus \{\mathsf{O}l, \mathsf{O}\sim l, \mathsf{P}\sim l\}$
  if $\mathsf{O}l \notin F$ **then**
    $R \leftarrow \{A(r) \setminus \{\mathsf{O}l, \neg\mathsf{O}\sim l\} \hookrightarrow C(r) |\ r \in R\} \setminus \{r \in R |\ A(r) \cap \widetilde{\mathsf{O}}l \neq \varnothing\}$
    $>\!\leftarrow\!> \setminus \{(r,s), (s,r) \in >\ |\ A(r) \cap \widetilde{\mathsf{O}}l \neq \varnothing\}$
  end if
  $R \leftarrow \{A(r) \hookrightarrow C(r) \ominus l |\ r \in R^{\mathsf{O}}[l,n]$ for an index $n\}$
  $R \leftarrow \{A(r) \hookrightarrow C(r) \ominus \sim l |\ r \in R^{\mathsf{P}}[\sim l,n]$ for an index $n\}$
  $R \leftarrow \{A(r) \hookrightarrow C(r)!\sim l \ominus \sim l |\ r \in R^{\mathsf{O}}[\sim l,n]$ for an index $n\}$
end procedure

Algorithm 3 ModifyPerm

procedure ModifyPerm($l \in$ Lit)
  $\partial^+_{\mathsf{P}} \leftarrow \partial^+_{\mathsf{P}} \cup \{l\}$
  $\partial^-_{\mathsf{O}} \leftarrow \partial^-_{\mathsf{O}} \cup \{\sim l\}$
  $HB \leftarrow HB \setminus \{\mathsf{P}l, \mathsf{O}\sim l\}$
  if $\mathsf{P}l \notin F$ **then**
    $R \leftarrow \{A(r) \setminus \{\mathsf{P}l, \neg\mathsf{O}\sim l\} \hookrightarrow C(r) |\ r \in R\} \setminus \{r \in R |\ A(r) \cap \widetilde{\mathsf{P}}l \neq \varnothing\}$
    $>\!\leftarrow\!> \setminus \{(r,s), (s,r) \in >\ |\ A(r) \cap \widetilde{\mathsf{P}}l \neq \varnothing\}$
  end if
  $R \leftarrow \{A(r) \hookrightarrow C(r)!\sim l \ominus \sim l |\ r \in R^{\mathsf{O}}[\sim l,n]$ for an index $n\}$
  $R \leftarrow \{A(r) \hookrightarrow C(r)!l |\ r \in R^{\mathsf{P}}[l,n]$ for an index $n\}$
end procedure

Algorithm 4 CheckFacts

procedure CheckFacts
  for $l \in F$ **do**
    $R \leftarrow \{A(r) \setminus \{l\} \hookrightarrow C(r) |\ r \in R\} \setminus \{r \in R |\ A(r) \cap \tilde{l} \neq \varnothing\}$
    $>\!\leftarrow\!> \setminus \{(r,s), (s,r) \in >\ |\ A(r) \cap \tilde{l} \neq \varnothing\}$
    switch (l)
      case $l \in$ Lit**:**
        $R \leftarrow \{A(r) \hookrightarrow C(r)!l |\ r \in R^{\mathsf{O}}[l,n]$ for an index $n\}$
      case $l = \mathsf{O}m$**:**
        ModifyObl(m)
      case $l = \neg\mathsf{O}m$**:**
        $-\partial_{\mathsf{O}} \leftarrow -\partial_{\mathsf{O}} \cup \{m\}$
        $HB \leftarrow HB \setminus \{\mathsf{O}m\}$
        $R \leftarrow \{A(r) \hookrightarrow C(r)!m \ominus m |\ r \in R^{\mathsf{O}}[m,n]$ for an index $n\}$
      case $l = \mathsf{P}m$**:**
        ModifyPerm(m)
      case $l = \neg\mathsf{P}m$**:**
        $-\partial_{\mathsf{P}} \leftarrow -\partial_{\mathsf{P}} \cup \{m\}$
        $HB \leftarrow HB \setminus \{\mathsf{P}m, \mathsf{O}m\}$
        $R \leftarrow \{A(r) \setminus \hookrightarrow C(r) \ominus m |\ r \in R^{\mathsf{P}}[m,n]$ for an index $n\}$
    end switch
  end for
end procedure

Before describing how Algorithm 4 works, let us recall some concepts about the provability of a literal. Given a Defeasible Theory, a modal literal $\Box l \in F$ is trivially proved with the corresponding modality by definition. Furthermore, we also stated that a non-modal literal is proved within the theory if it is a fact.

Based on these facts, the procedure described in Algorithm 4 CheckFacts begins by

removing all factual literals from every rule where they appear as an antecedent; it also removes all rules whose body contains a complementary literal (line 3). The superiority relation is then modified in view of this operation (line 4).

Different operations are now bring about based on which factual literal is considered.

1. If l is a non-modal literal, we truncate the head of all rules at l, where l appears as an obligation (lines 6–7);
2. if l is $\mathsf{O}m$ (lines 8–9) or $\mathsf{P}m$ (lines 14–15), then Algorithm 2 ModifyObl (resp. Algorithm 3 ModifyPerm) properly modifies the theory. Notice that operations in lines 7–8 of Algorithm 2 ModifyObl and 6–7 of Algorithm 3 ModifyPerm are not performed, since they are equivalent to lines 3–4 of Algorithm 4 CheckFacts;
3. if l is $\neg\mathsf{O}m$ (lines 10–13), then $-\partial_{\mathsf{O}}m$ holds, and by clause (2) of Definition 8 all rules with m as an obligation in their heads are discarded for all literals after m. Hence, we truncate all such chains at m, and then remove m (line 14);
4. if l is $\neg\mathsf{P}m$ (lines 16–19), then $-\partial_{\mathsf{P}}m$ holds in the theory. Thus, we remove m in every chain where m appears as a permission (line 22).

We conclude this section by presenting and describing Algorithm 5 ComputeDefeasible, which represents the core for the computation of the defeasible extension of a theory.

Algorithm 5 ComputeDefeasible

Input: A defeasible theory D.
Output: The extension $E(D)$ of D.

1: $\pm\partial_{\square} \leftarrow \varnothing$ with $\square \in$ MOD
2: $R[l]_{infd} \leftarrow \varnothing$ for each $l \in$ Lit
3: checkFacts
4: **repeat**
5: $\quad\partial^{\pm}_{\square} \leftarrow \varnothing$
6: $\quad$**for** $\square l \in HB, \square \in \{\mathsf{O}, \mathsf{P}\}$ **do**
7: $\quad\quad$**if** $R^{\square}[l] = \varnothing$ **then** Discard$(l, \square)$
8: $\quad\quad$**if** there exists $r \in R^{\mathsf{O}}[l, 1]$ such that $A(r) = \varnothing$ **then**
9: $\quad\quad\quad R[\sim l]_{infd} \leftarrow R[\sim l]_{infd} \cup \{s \in R[\sim l] |\ r > s\}$
10: $\quad\quad\quad$**if** $\{s \in R[\sim l] |\ s > r\} = \varnothing$ **then**
11: $\quad\quad\quad\quad$Discard$(\sim l, \square)$
12: $\quad\quad\quad\quad$**if** $R[\sim l] \setminus R[\sim l]_{infd} = \varnothing$ and $\neg\mathsf{O}l \notin F$ **then** ModifyObl(l)
13: $\quad\quad\quad$**end if**
14: $\quad\quad$**end if**
15: $\quad\quad$**if** there exists $r \in R^{\mathsf{P}}[l, 1]$ such that $A(r) = \varnothing$ **then**
16: $\quad\quad\quad R[\sim l]_{infd} \leftarrow R[\sim l]_{infd} \cup \{s \in R^{\mathsf{O}}[\sim l] |\ r > s\}$
17: $\quad\quad\quad$**if** $\{s \in R[\sim l] |\ s > r\} = \varnothing$ **then**
18: $\quad\quad\quad\quad$Discard$(\sim l, \mathsf{O})$
19: $\quad\quad\quad\quad$**if** $R^{\mathsf{O}}[\sim l] \setminus R[\sim l]_{infd} = \varnothing$ **then** ModifyPerm(l)
20: $\quad\quad\quad$**end if**
21: $\quad\quad$**end if**
22: $\quad$**end for**
23: $\quad\pm\partial_{\square} \leftarrow \pm\partial_{\square} \cup \partial^{\pm}_{\square}$
24: **until** $\partial^{+}_{\square} = \varnothing$ and $\partial^{-}_{\square} = \varnothing$
25: **return** $E(D) = (+\partial_{\mathsf{O}}, +\partial_{\mathsf{P}}, -\partial_{\mathsf{O}}, -\partial_{\mathsf{P}})$

Lines 1–2 initialise variables $+\partial_{\mathsf{O}}$, $+\partial_{\mathsf{P}}$ and, for each literal l, a set $R[l]_{infd}$ containing all the rules for l that are defeated by a rule for the opposite. Algorithm 4 CheckFacts is invoked to compute all defeasible conclusions derived from the set of facts (line 3).

The algorithm consists of a main loop (the **repeat** cycle at lines 4–24) that performs a series of transformations to reduce a Defeasible Theory into a simpler equivalent one. The loop ends when no more modifications on the extension are made, i.e., when both variables $\partial^{+}_{\square}$ and $\partial^{-}_{\square}$ are empty at the end of an iteration.

The **for** cycle at lines 6–22 checks all the rules for every literal l of the theory. At line 7, Algorithm 1 DISCARD is invoked for all modal literals with no supporting chains. Lines 8–21 loop over all rules in the theory for the current literal l, and checks if an applicable rule exists with l as first element in its head. If the rule introduces l as an obligation (lines 8–14), then we have to collect all rules for the opposite, and check if they are all defeated by a rule for l. If this is the case, then $+\partial_{\mathsf{O}} l$ holds and Algorithm 2 MODIFYOBL is invoked.

On the other hand, if l is introduced as a permission (lines 15–21), then we have to take into account only obligation rules for $\sim l$, and to check if every rule for $\sim l$ as an obligation is defeated by at least one rule for the opposite. If so, condition $+\partial_{\mathsf{P}} l$ holds, and Algorithm 3 MODIFYPERM is invoked. Finally, all modifications on the extension, due to the execution of the cycle, are stored in the global sets of conclusions (line 23).

4.3. Properties of defeasible theory transformations

The properties we are going to show are related to operations that transform a theory D into an equivalent simpler theory D' (where the term 'simpler' denotes a theory with a minor number of symbols in it).

The transformations operate either by removing some elements from it, or by deleting a rule from the theory. Given the functional nature of the operations, we will refer to the rules in the target theory with the same names/labels as the rules in the source theory. Thus, given a rule $r \in D$, we will refer to the rule corresponding to it in D' (if it exists) with the same label, namely r. The determined reader is referred to [11] for the rigorous proofs of Lemmas 20–29.

Given a non-modal literal $p \in F$, we can obtain an equivalent theory by removing p in every rule where it appears in the antecedent. Moreover, if the rule is for an obligation, Definition 8 clause (2) ensures that the rule will be discarded for every element after p, and therefore we can truncate the reparation chain at p. Instead, if the rule is for a permission, we cannot operate on it. In both cases, we only consider rules where the complement of p does not appear in the antecedent. Finally, the superiority relation can be simplified by removing all tuples with a rule containing $\sim p$ in the antecedent, or an obligation rule for an element after p in its consequent.

Lemma 20. *Let $D = (F, R, >)$ be a theory such that $p \in F \cap \mathrm{Lit}$. Let $D' = (F', R', >')$ be the theory obtained from D where*

$$
\begin{aligned}
F' &= F \setminus \{p\} \\
R' &= \{r : A(r) \setminus \{p\} \Rightarrow_{\mathsf{O}} C(r)!p \mid r \in R,\ A(r) \cap \tilde{p} = \varnothing\} \cup \\
&\quad \{r : A(r) \setminus \{p\} \Rightarrow_{\mathsf{P}} C(r) \mid r \in R,\ A(r) \cap \tilde{p} = \varnothing\} \\
>' &= > \setminus \{(r,s), (s,r) \mid r, s \in R, A(r) \cap \tilde{p} \neq \varnothing\}.
\end{aligned}
$$

Then $D \equiv D'$.

Starting from the modified theory given by the transformations of the previous lemma, we now consider a theory with only modal literals in the set of facts. If a literal p is provable as an obligation, then we can simplify the theory by removing $\mathsf{O}p$ in every antecedent of the rules in R, and erase the rules where at least one element of $\widetilde{\mathsf{O}p}$ appears in the antecedent. Since by hypothesis $F \cap \mathrm{Lit} = \varnothing$, if p is present in the reparation chain of an obligation rule, we simplify the theory by removing p from the consequent. If $\sim p$ is in the consequent, we

can also truncate the reparation chain of the rule since, by Definition 8 clause (2), the rule will be discarded for each element after $\sim p$ (Proposition 16 Part 1 states that $-\partial_{\mathsf{O}}\sim p$ holds as well). Moreover, Proposition 16 Part 2 ensures that $-\partial_{\mathsf{P}}\sim p$ holds. Thus, Definition 7 clause (3) allows us to remove $\sim p$ in the consequent of permission rules for $\sim p$. Finally, the superiority relation can be simplified by removing all tuples with a rule containing at least one element of $\widetilde{\mathsf{O}p}$ in the antecedent.

Lemma 21. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{Lit} = \varnothing$ and $D \vdash +\partial_{\mathsf{O}}p$. Let $D' = (F, R', >')$ be the theory obtained from D where*

$$\begin{aligned} R' = &\{r : A(r) \setminus \{\mathsf{O}p\} \Rightarrow_{\mathsf{O}} C(r)!\sim p \ominus \sim p \mid r \in R,\ A(r) \cap \widetilde{\mathsf{O}p} = \varnothing\} \cup \\ &\{r : A(r) \setminus \{\mathsf{O}p\} \Rightarrow_{\mathsf{O}} C(r) \ominus p \mid r \in R,\ A(r) \cap \widetilde{\mathsf{O}p} = \varnothing\} \cup \\ &\{r : A(r) \setminus \{\mathsf{O}p\} \Rightarrow_{\mathsf{P}} C(r) \ominus \sim p \mid r \in R,\ A(r) \cap \widetilde{\mathsf{O}p} = \varnothing\} \\ >' = &> \setminus \{(r,s), (s,r) \mid r, s \in R, A(r) \cap \widetilde{\mathsf{O}p} \neq \varnothing\}. \end{aligned}$$

Then $D \equiv D'$.

As the previous lemma, we consider a theory with only modal literals in the set of facts. Since the theory proves $-\partial_{\mathsf{O}}p$, also $\neg\mathsf{O}p$ holds. Thus, we obtain an equivalent simpler theory by erasing all rules with $\mathsf{O}p$ as one of the antecedents, and by removing $\neg\mathsf{O}p$ in each rule where it appears in the antecedent. Again, by Definition 8 clause (2), for every obligation rule we can truncate each reparation chain with p in the consequent and eliminate it from such a chain. Finally, the superiority relation can be simplified by removing all the pairs with a rule containing $\mathsf{O}p$ in the antecedent.

Lemma 22. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{Lit} = \varnothing$ and $D \vdash -\partial_{\mathsf{O}}p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' = &\{r : A(r) \setminus \{\neg\mathsf{O}p\} \Rightarrow_{\Box} C(r) \mid r \in R,\ A(r) \cap \{\mathsf{O}p\} = \varnothing\} \cup \\ &\{r : A(r) \Rightarrow_{\mathsf{O}} C(r)!p \ominus p \mid r \in R,\ A(r) \cap \{\mathsf{O}p\} = \varnothing\} \\ >' = &> \setminus \{(r,s), (s,r) \mid r, s \in R, A(r) \cap \{\mathsf{O}p\} \neq \varnothing\}. \end{aligned}$$

Then $D \equiv D'$.

We can defeasibly prove a literal p as a permission. A simpler equivalent theory is one where we remove $\mathsf{P}p$ in each set of antecedents and where we erase all the rules containing at least one element of the complement of $\widetilde{\mathsf{P}p}$ in the antecedent. Proposition 16 Part 3 states that $-\partial_{\mathsf{O}}\sim p$ holds. Thus, by Definition 8 clause (2), if $\sim p$ appears in the reparation chain of an obligation rule, we can remove it after having truncate the chain at $\sim p$. Instead, if we consider permission rules with p in the consequent, by Definition 8 clause (3), we can truncate the corresponding reparation chain at p. Finally, the superiority relation can be simplified by removing all the pairs with a rule with an element of $\widetilde{\mathsf{P}p}$ in the antecedent.

Lemma 23. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{Lit} = \varnothing$ and $D \vdash +\partial_{\mathsf{P}}p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' = &\{r : A(r) \setminus \{\mathsf{P}p\} \Rightarrow_{\mathsf{O}} C(r)!\sim p \ominus \sim p \mid r \in R,\ A(r) \cap \widetilde{\mathsf{P}p} = \varnothing\} \cup \\ &\{r : A(r) \setminus \{\mathsf{P}p\} \Rightarrow_{\mathsf{P}} C(r)!p \mid r \in R,\ A(r) \cap \widetilde{\mathsf{P}p} = \varnothing\} \\ >' = &> \setminus \{(r,s), (s,r) \mid r, s \in R, A(r) \cap \widetilde{\mathsf{P}p} \neq \varnothing\}. \end{aligned}$$

Then $D \equiv D'$.

The theory proves $-\partial_{\mathsf{P}} p$, allowing $\neg \mathsf{P} p$ to hold. Thus, we obtain an equivalent theory if we erase all the rules with $\mathsf{P} p$ in the set of the antecedents and if we remove $\neg \mathsf{P} p$ where it appears in the tail of a rule. Moreover, if the rule is for a permission, we remove p from the reparation chain. Finally, we change the superiority relation by erasing the tuples with a rule with $\mathsf{P} p$ in the antecedent.

Lemma 24. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{Lit} = \varnothing$ and $D \vdash -\partial_{\mathsf{P}} p$. Let $D' = (F, R', >')$ be theory obtained from D where*

$$\begin{aligned} R' = &\{r : A(r) \setminus \{\neg \mathsf{P} p\} \Rightarrow_{\square} C(r) \mid r \in R,\ A(r) \cap \{\mathsf{P} p\} = \varnothing\} \cup \\ &\{r : A(r) \Rightarrow_{\mathsf{P}} C(r) \ominus p \mid r \in R,\ A(r) \cap \{\mathsf{P} p\} = \varnothing\} \\ >' = &> \setminus \{(r, s), (s, r) \mid r, s \in R, A(r) \cap \{\mathsf{P} p\} \neq \varnothing\}. \end{aligned}$$

Then $D \equiv D'$.

The following two lemmas represent conditions under which a literal can be proved as an obligation or as a permission. The transformations dictated by the previous lemmas empty the antecedent of every applicable rule.

Definition 25. *Given a theory $D = (F, R, >)$, and a set of rules S, the subset of S of* inferiorly defeated *rules for a literal p, $S[p]_{infd}$ is thus defined: $r \in S[p]_{infd}$ iff*

1. *$\exists s \in R[\sim p]$ such that $A(r) = \varnothing$ and $s > r$, and*
2. *if $r \in R^{\mathsf{P}}[p]$, then $s \in R^{\mathsf{O}}[\sim p]$.*

Lemma 26. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{ModLit} = \varnothing$, $\exists r \in R^{\mathsf{O}}[p, 1]$, $A(r) = \varnothing$, and $R[\sim p] \subseteq R_{infd}$. Then $D \vdash +\partial_{\mathsf{O}} p$.*

Lemma 27. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{ModLit} = \varnothing$, $\exists r \in R^{\mathsf{P}}[p, 1]$, $A(r) = \varnothing$, and $R^{\mathsf{O}}[\sim p] \subseteq R_{infd}$. Then $D \vdash +\partial_{\mathsf{P}} p$.*

The next two lemmas concern conditions to determine when it is possible to assert that a literal is negatively provable.

Lemma 28. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{ModLit} = \varnothing$ and $R^{\square}[p] = \varnothing$, for $\square \in \{\mathsf{O}, \mathsf{P}\}$. Then $D \vdash -\partial_{\square} p$.*

Lemma 29. *Let $D = (F, R, >)$ be a theory such that $F \cap \mathrm{ModLit} = \varnothing$, and $\exists r \in R[p, 1]$ such that $A(r) = \varnothing$ and $r_{sup} = \varnothing$. Then*

1. *if $r \in R^{\mathsf{O}}$, then $D \vdash -\partial_{\square} \sim p$, $\square \in \{\mathsf{O}, \mathsf{P}\}$;*
2. *if $r \in R^{\mathsf{P}} \cup R_{def}$, then $D \vdash -\partial_{\mathsf{O}} \sim p$.*

5. Computational results

In this section we present the computational properties of the algorithms previously described. Since, as stated above, the first three algorithms are sub-routines of the two main ones, we will present the correctness and completeness results of these algorithms inside theorems for Algorithms 4 CheckFacts and 5 ComputeDefeasible.

We start by introducing the following definition.

Definition 30. *Given a Defeasible Theory D, the* size *S of D is the number of literal occurrences plus the number of the rules in D.*

We also report some key ideas and intuitions behind our implementation.

1. Each operation on global sets $\pm\partial_{\Box}$ and $\partial^{\pm}_{\Box}$ requires a constant time, as we manipulate finite sets of literals;
2. For each literal $\Box l \in HB$, we implement a hash table with pointers to rules where the literal occurs; thus, retrieving the set of rules containing a given literal requires constant time.
3. The superiority relation is also implemented by means of hash tables; once again, the information required to modify a given tuple is accessed in constant time.

Theorem 31. *Given a modal Defeasible Theory D with size S, Algorithm 4 CheckFacts terminates and its computational complexity is $O(S)$.*

Proof. Termination of Algorithm 4 CheckFacts is given by definition of modal Defeasible Theory, since the internal sub-routines (i.e., Algorithm 2 ModifyObl and 3 ModifyPerm), as well as the algorithm itself, manipulate finite sets of rules and facts.

For a correct analysis of the complexity of Algorithm 4 CheckFacts, it is of the utmost importance to correctly comprehend Definition 30. Here we underline that the size S of a theory represents the total number of occurrences of each literal in every rule of such a theory. Let us examine a theory with Y literals and whose size is Z. If we consider a cycle whose purpose is to call, for each literal, a procedure that selectively deletes it from all the rules of the theory (no matter to what end), a rough computational complexity would be $O(Z^2)$. In fact, the complexity of the procedure by itself is bound to the number of rules in the theory, which is in $O(Z)$, and this procedure is iterated $Y \in O(Z)$ times.

However, a more fined-grained analysis shows that the complexity of this loop is lower. The mistake of the previous analysis is that it considers the complexity of the procedure separately from the complexity of the external loop, whilst they are strictly dependent. Indeed, the overall number of operations made by the sum of all loop iterations cannot outrun the number of occurrences of the literals, $O(Z)$, because the operations in the inner procedure directly decrease, iteration after iteration, the number of the remaining repetitions of the outmost loop, and the other way around. Therefore, the overall complexity is not bound to $O(Z) \cdot O(Z) = O(Z^2)$, but to $O(Z) + O(Z) = O(Z)$.

We can contextualise the above reasoning to Algorithm 4 CheckFacts. The main cycle at lines 2–21 is iterated over the set of facts, whose cardinality is in $O(S)$; the operations at lines 9 and 15 (invoking Algorithms 2 ModifyObl and 3 ModifyPerm) represent an additive factor $O(S)$ in the overall complexity of the algorithm. Finally, all operations on the set of rules and the superiority relation performed require constant time, given the implementation of data structures proposed above. Therefore, we can state that the complexity of the algorithm is $O(S)$. □

Theorem 32. *Given a Defeasible Theory D with size S, Algorithm 5 ComputeDefeasible terminates and its computational complexity is $O(S)$.*

Proof. When referring to the termination of Algorithm 5 ComputeDefeasible, the most important part we have to analyse is the **repeat** cycle at lines 4–24. Once an instance of the cycle has been performed, one of the following (mutually exclusive) situations occurs:

1. no modification of the extension has occurred. In this case, line 24 ensures the termination of the algorithm;
2. the theory is modified with respect to a literal in Modal Herbrand Base HB. The algorithm removes the literal from HB once it has performed the suitable operations. As HB is finite, it will eventually emptied; at the next iteration of the cycle, we have no means to modify the extension. In this case, the algorithm terminates as well.

The analysis of the complexity of Algorithm 5 ComputeDefeasible straightly follows from the reasoning proposed to demonstrate the computational complexity of Algorithm 4 CheckFacts. Thus, the **repeat** cycle at lines 4–24 is in $O(S)$. Invocation of Algorithm 1 Discard adds a constant time, as well as all operations on the set of rules or superiority relation. Furthermore, since all invocations of Algorithm 2, 3, and 4 represent an additive factor in the order of $O(S)$, we conclude that the computational complexity of Algorithm 5 ComputeDefeasible is bound to $O(S)$. □

Theorem 33. *Algorithm 5 ComputeDefeasible is sound and complete.*

Proof. As already argued at the beginning of the section, the aim of Algorithm 5 ComputeDefeasible is to compute the defeasible extension of a given theory D through successive transformations on the set of facts and rules, and on the superiority relation. These transformations act in ways which obtain at each step a new simpler theory while retaining the same extension. Since we have already proved the termination of the algorithm, it eventually comes to a fix-point theory where no more operations can be made.

To demonstrate the soundness of Algorithm 5 ComputeDefeasible, we show in the table below that all the operations performed by the algorithm are those described in Lemmas 20–29, where we have already proved the soundness of the operation involved.

Algorithm	Line(s)		Algorithm	Line(s)	
Discard	3–4	Lem. 22, 24	ModifyObl	9–11	Lem. 21
ModifyPerm	9–10	Lem. 23	ModifyObl	5–8	Prop. 15, Lem. 21
ModifyPerm	5–8	Prop. 14, Lem. 23	CheckFacts	3–4	Lem. 20, 22–24
ComputeDefeasible	7	Lem. 28	CheckFacts	6–7	Lem. 20
ComputeDefeasible	11, 18	Lem. 29	CheckFacts	8–9	Alg. 2
ComputeDefeasible	12	Lem. 21, 26	CheckFacts	10–13	Lem. 22
ComputeDefeasible	19	Lem. 23, 27	CheckFacts	14–15	Alg. 3
			CheckFacts	16–19	Lem. 24

These results state that if in the initial theory a literal is either defeasibly proved or not, so it will be in the final theory; thus proving the soundness of the algorithm.

Moreover, the completeness of Algorithm 5 ComputeDefeasible follows from the fact that: (i.) all lemmas show the equivalence of the two theories, and (ii.) the equivalence relation is a bijection. □

6. Discussion: Different types of permission

Resuming the discussion of Section 2, we delve into the technical aspects of the three mentioned concepts of permissions within our framework.

The idea of weak (or negative) permission is easily represented as follows:

Definition 34. *Let D be a Defeasible Theory. A literal l is* weakly permitted *iff $D \vdash -\partial_{\mathrm{O}} \sim l$.*

One remark is in order: Definition 34 is useful to check whether a literal l is weakly permitted within the theory, but it cannot be directly used to explicitly derive $\mathsf{P}l$ for triggering any rule where this modal literal occurs in the antecedent. If $\mathsf{P}l$ appears in the antecedent of a rule, then the only way to activate such a rule is to explicitly derive $\mathsf{P}l$.

However, weak permissions are decisive in the applicability of a rule for conditions (1.2) of Definitions 6 and 7; when $\neg \mathsf{O} l$ occurs in the antecedent of the rule, then the theory must prove $-\partial_{\mathsf{O}} l$. This is equivalent to saying that $\sim l$ is weakly permitted in D.

This reading assumes that the distinction between weak and strong permission goes beyond the idea, defended by [2], that there is only one prescriptive type of permission. If the reader finds our proposal limiting, we can trivially revise the rule applicability conditions at point (1.3) (and adjust the algorithms accordingly) by establishing that, when $\mathsf{P} l$ occurs in the antecedent of any rule r, r is applicable if one of the following conditions holds: (i) $+\partial_{\mathsf{P}} l$, or (ii) $-\partial_{\mathsf{O}} \sim l$ (notice that $+\partial_{\mathsf{P}} l$ implies $-\partial_{\mathsf{O}} \sim l$, but not vice versa).

A simple result (from Proposition 16 Part 1) regarding weak permissions follows:

Proposition 35. *Let D be any O-consistent Defeasible Theory. For any literal l, if $D \vdash +\partial_{\mathsf{O}} l$, then l is weakly permitted.*

As expected, weak permission enjoys the deontic principle "Ought implies Can", i.e., the principle that in deontic logic is $\mathsf{O} l \rightarrow \mathsf{P} l$.

We now consider the two ways to obtain strong permissions in DL: using either explicit permissive norms or defeaters.

The first case is naturally captured in the logical framework proposed in Sections 3 and 4. In the simplest case, a literal like $\mathsf{P} l$ is derived in a theory D when there is a successful reasoning chain in which the last rule has the form $a_1, \ldots, a_n \Rightarrow_{\mathsf{P}} l$.

More complex cases are due to the fact that l may occur in an $\oslash$-expression. In this case l is obtained as strongly permitted if, for each literal c preceding l in the sequence, one of the following conditions hold:

- if c leads to derive $\mathsf{O} c$ (i.e., c occurs in an $\otimes$-subsequence of the main sequence where l occurs), then this obligation must be obtained and violated;
- if c leads to derive $\mathsf{P} c$ (i.e., c occurs in an $\odot$-subsequence of the main sequence where l occurs), then this permission is successfully attacked by an opposite obligation.

The introduction of sequences of permissions and obligations enriches the language in a significant way, since it allows us to express a preference among obligations and permissions when they are logically compatible. In the case of sequences of positive permissions, an $\odot$-sequence states that a permissive exception of an obligation is preferred with respect to another possible exception of the same obligation. However, this extension does not conceptually change the fundamental intuition that is also behind the basic case where permissive norms have the form $a_1, \ldots, a_n \Rightarrow_{\mathsf{P}} l$: the antecedent of positive permissive rules with head l provides sufficient defeasible reasons to derive $\mathsf{P} l$.

The second method considered in Section 2 to capture the notion of strong permissions acting as exceptions to obligations looks at permissions as *undercutters* in argumentation theory (this idea was discussed in [7]): a permissive norm with head l operates in such a way that, if applicable, it is not a sufficient reason for deriving neither l, nor $\sim l$, but it is a sufficient reason for blocking the derivation of $\sim l$ as obligatory. In DL, this idea is naturally implemented by using defeaters. For the sake of simplicity, we have not considered this concept of strong permission in Sections 3 and 4. However, to cover this case it is sufficient to adopt one the following definitions (compare the definition of $R^{\mathsf{P}}[q, n]$ in Section 3):

1. $R^{\mathsf{P}}[q, n] = R^{\mathsf{P}}[q, n] \cup R_{def}[q]$;
2. $R^{\mathsf{P}}[q, n] = R_{def}[q]$.

The first definition adds the defeaters to the set of rules that can be used to derive tagged literals like $+\partial_{\mathsf{P}} l$, obtaining modal literals like $\mathsf{P}l$. The latter definition restricts derivations of strong permissions only to reasoning chains where the last rule is a defeater. In both cases we do not need to change anything else in the logic (hence, in the proof conditions for $\pm\partial_{\mathsf{P}}$) or in the algorithms.

What is the difference between strong permissions obtained via rules for permission and the ones obtained via defeaters? Although rules for P and defeaters are not in general equivalent, as we have informally suggested in Section 2, they behave quite similarly when they are used to derive permissions, as well as to attack obligation rules supporting the opposite conclusion. In other words, defeaters do not clash with any permissive rules. Consequently, if this reading of defeaters is simply embedded within the proof conditions for $\pm\partial_{\mathsf{P}}$ and for $\pm\partial_{\mathsf{O}}$ by adopting either the above definition (1) or (2), then rules for P and defeaters play a very similar role in the proof theory. In fact, if we consider condition (2.3.3) in Definition 9, then two rules like $r_1 : a_1, \ldots, a_n \Rightarrow_{\mathsf{P}} l$ and $r_2 : a_1, \ldots, a_n \rightsquigarrow l$ both attack any rule s for obligation supporting $\sim l$, and s can counterattack r_1 and r_2 as well.

We remark that the significant difference between the rules for P and defeaters is that defeaters do not allow for having sequences of permissions in their head.

Finally, notice that neither type of strong permission considered enjoys the principle "Ought implies Can". This result comes directly from Proposition 16 and is based on the idea that the only manner to derive strong permissions is by means of reasoning chains where the last rule occurring in them is an explicit permissive norm.

7. Summary and related work

In this paper we examined different types of explicit permissive norms. We discussed how strong permissions can be represented with or without introducing a new consequence relation for inferring conclusions from explicit permissive norms. Finally, we combined a preference operator applicable to contrary-to-duty obligations with a new one for ordered sequences of strong permissions. Our goal was twofold: (i.) to capture some aspects of permissions within a broader view of defeasible normative reasoning; (ii.) to study the defeasibility of permissions in a computationally efficient logical framework.

A few works in the recent literature have offered significant contributions to the logical understanding of permission [24,6,7,9,27,26]. Our work shares with [24,6,27] some conceptual assumptions. Slightly rephrasing [27]'s analysis, the following guidelines inspired our concept of permission:

1. "no logic of norms without attention to the system of which they form part" [22]: our investigation of the concept of permission looks at how permissive norms and other types of norms interact within systems;
2. the distinction between positive and negative permission is meaningful;
3. one fundamental role of positive permissions is that of stating exceptions to obligations; accordingly, positive permissions are supposed to override, or at least block, some deontic conclusions coming from other norms;
4. the logical space of weak permission is the one left unregulated by mandatory norms.

However, [24,6,27] are based on a different logical formalism, Input/Output Logic (IOL) [23], thus it is difficult (and out of the scope) to compare in detail those contributions with the present one. However, there are general similarities in both approaches.

- First, DL, like IOL, models explicit and implicit permissions by distinguishing in an analogous manner a consequence relation for obligation and one for permission.
- Second, the definition of negative or weak permission in both formalisms is the same.
- Third, although we have not discussed here the notion of static permission [6], it seems relatively simple to capture it: we may derive that some p is permitted by making an essential use in the derivation of a rule for explicit permission. The only different feature with respect to DL is that in IOL static permission admits the principle "Ought implies Can", which does not hold for strong permission in our approach.

A novelty of our paper is the introduction of the new operator $\odot$ to express preferences between explicit permissions. A somehow similar idea has been suggested by [6], where a preference relation among pairs (for obligations and permissions) was introduced. Technically, it is not clear if that approach can be reframed in our setting. In fact, the superiority relation in DL plays a role in the proof theory only in case of rule conflicts. Clear advantages of the current proposal are: (i.) modal operators can occur in the applicability conditions; (ii.) we have two ordering types between permissions: the one expressed by $\odot$ and the one induced by the superiority relation applied to defeaters.

To the best of our knowledge, there is no logical system having linear complexity with analogous normative reasoning capabilities.

Acknowledgements

This work is an extended and revised version of the paper presented at Jurix 2011 [10]. We thank the anonymous referees for their valuable comments.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy, the Australian Research Council through the ICT Centre of Excellence program and the Queensland Government.

References

[1] C. E. Alchourrón. Philosophical foundations of deontic logic and the logic of defeasible conditionals. In J. J. Meyer and R. J. Wieringa, editors, *Deontic Logic in Computer Science: Normative System Specification*. Wiley, 1993.

[2] C. E. Alchourrón and E. Bulygin. Permission and permissive norms. In W. Krawietz et al., editor, *Theorie der Normen*. Duncker & Humblot, 1984.

[3] G. Antoniou, D. Billington, G. Governatori, Michael J. Maher, and Andrew Rock. A family of defeasible reasoning logics and its implementation. In *ECAI 2000*, pages 459–463, 2000.

[4] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Trans. Comput. Logic*, 2:255–287, April 2001.

[5] N. Bobbio. *Teoria della norma giuridica*. Giappichelli, 1958.

[6] G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *ICAIL '03*, pages 109–118. ACM, 2003.

[7] G. Boella and L. van der Torre. Permissions and undercutters. In *NRAC'03*, pages 51–57, Acapulco, 2003.

[8] G. Boella and L. van der Torre. Permission and authorization in normative multiagent systems. In *ICAIL '05*, pages 236–237. ACM, 2005.

[9] M.A. Brown. Conditional obligation and positive permission for agents in time. *Nordic Journal of Philosophical Logic*, 5(2):83–111, 2000.

[10] G. Governatori, F. Olivieri, A. Rotolo, and S. Scannapieco. Three concepts of defeasible permission. In K. M. Atkinson, editor, *JURIX 2011*, pages 63–72. IOS Press, 2011.

[11] G. Governatori, F. Olivieri, A. Rotolo, and S. Scannapieco. Computing strong and weak permissions in defeasible logic. *CoRR*, abs/1212.0079, 2012.

[12] G. Governatori, V. Padmanabhan, A. Rotolo, and A. Sattar. A defeasible logic for modelling policy-based intentions and motivational attitudes. *Logic Journal of the IGPL*, 17(3):227–265, 2009.
[13] G. Governatori and A. Rotolo. Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.
[14] G. Governatori and A. Rotolo. Justice delayed is justice denied: Logics for a temporal account of reparations and legal compliance. In *CLIMA XII*, volume 6814 of *Lecture Notes in Computer Science*, pages 364–382. Springer, 2011.
[15] G. Governatori, A. Rotolo, and E. Calardo. Possible world semantics for defeasible deontic logic. In T. Ågotnes, J. Broersen, and D. Elgesem, editors, *DEON 2012*, volume 7393 of *LNCS*, pages 46–60. Springer, 2012.
[16] G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *ICAIL 05*, pages 25–34. ACM Press, 2005.
[17] G. Governatori and S. Sadiq. The journey to business process compliance. *Handbook of Research on BPM*, pages 426–454, 2008.
[18] G. Governatori and S. Shek. Rule based business process compliance. In *Proceedings of the RuleML2012@ECAI Challenge*, number 874 in CEUR, page 5, 2012.
[19] W.N. Hohfeld, W.W. Cook, and A.L. Corbin. *Fundamental Legal Conceptions: As Applied in Judicial Reasoning*. Lawbook Exchange, Limited, 2010.
[20] H. Lam and G. Governatori. What are the Necessity Rules in Defeasible Reasoning? In James Delgrande and Wolfgang Faber, editors, *LPNMR-11*, pages 187–192. Springer, 2011.
[21] M. J. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, 2001.
[22] D. Makinson. On a fundamental problem of deontic logic. In Paul McNamara and Henry Prakken, editors, *Norms, Logics and Information Systems. New Studies in Deontic Logic and Computer Science*, pages 29–54. IOS Press, Amsterdam, 1999.
[23] D. Makinson and L. van der Torre. Input-output logics. *Journal of Philosophical Logic*, 29(4):383–408, 2000.
[24] D. Makinson and L. van der Torre. Permission from an input/output perspective. *Journal of Philosophical Logic*, 32(4):391–416, 2003.
[25] G. Sartor. *Legal Reasoning: A Cognitive Approach to the Law*. Springer, 2005.
[26] A. Stolpe. Relevance, derogation and permission. In *DEON*, pages 98–115. Springer, 2010.
[27] A. Stolpe. A theory of permission based on the notion of derogation. *J. Applied Logic*, 8(1):97–113, 2010.
[28] G.H. von Wright. *Norm and action: A logical inquiry*. Routledge and Kegan Paul, 1963.