Received: 8 May 2026 | Revised: 6 August 2026 | Accepted: 28 September 2026 | Published online: 30 October 2026

RESEARCH ARTICLE

Journal of Computational Law and Legal Technology
2026, Vol. 00(00) 1–10
DOI: 10.47852/bonviewJCLLT62029051

BON VIEW PUBLISHING

Weak Permission Is Not Well-Founded, Grounded, and Stable

Guido Governatori[1,*]

[1] *School of Engineering and Technology, Central Queensland University, Australia*

Abstract: We investigate the notion of weak permission as the failure to derive the obligation for the contrary. In particular, we focus on the issue of whether a weak permission should be derivable in the presence of a deontic conflict. Then, we investigate weak permission from the point of view of non-monotonic reasoning. Specifically, we look at two of the most popular paradigms for non-monotonic reasoning: logic programming and structured argumentation; and the commonly adopted semantics for each of them: well-founded and (skeptical) stable semantics for logic programming and grounded and stable semantics for structured argumentation. We prove that it is not possible to capture weak permission in the presence of deontic conflicts under these semantics. The results of our analysis have implications for the design of Rules as Code initiatives and computational law systems, as they highlight a fundamental tension between the logical treatment of deontic conflicts and the normative requirements of legitimate legal systems.

Keywords: weak permission, well-founded semantics, grounded semantics, stable semantics

1. Introduction

The recent *Rules as Code* movement aims to represent legislation, regulations, and administrative policies in forms that can be executed, queried, or checked by software systems [1]. Its practical motivation is clear: governments and institutions increasingly seek digital services capable of delivering timely, consistent, and explainable advice about rights, obligations, eligibility conditions, and compliance requirements [2]. Yet the translation of legal or normative materials into code is not merely an engineering exercise. It raises foundational questions about how rules should be represented, how exceptions and priorities should be handled, and how conclusions should be justified when the relevant norms or factual assumptions are in tension. It is at this point that computational law becomes directly relevant.

Computational law can be understood as the use of formal methods, specifically computational logic, to model legal reasoning and legal knowledge. One of the aims of computational law is to develop formal models and computational tools for representing and reasoning about legal norms, legal concepts, and legal reasoning processes. Computational law provides the conceptual and formal machinery required to move from textual norms to executable representations. At a minimum, it investigates how legal directives can be modeled in logical, rule-based, or computational form and how those representations can support automated reasoning, explanation, and verification. More specifically, it aims to determine what normative requirements (e.g., obligations, prohibitions, and permissions) follow from a given set of norms and facts and to identify if they have been violated or complied with. In addition, it should explain and justify those results in a way that is faithful to the underlying legal materials.

More importantly, computational law studies features of legal reasoning that are often suppressed in simplified digital implementations: defeasibility, exceptions, temporal qualifications, institutional competence, burdens of proof, and conflicts among norms. These are not peripheral complications. They are pervasive features of legal and normative domains, and any serious Rules as Code initiative must ultimately confront them if it wishes to remain faithful to the source materials it encodes.

Accordingly, computational law contributes to Rules as Code in at least three ways. First, it offers *representation languages* capable of expressing obligations, permissions, prohibitions, exceptions, priorities, and conditional applicability. Second, it supplies *reasoning frameworks*—including defeasible logics, argumentation systems, and deontic formalisms—for determining what follows from a body of potentially inconsistent or incomplete normative material. Third, it supports *explanatory and validation mechanisms* that make it possible to compare coded rules against legal sources and to justify the conclusions produced by an automated system. Without these contributions, Rules as Code risks collapsing into a superficial transcription exercise in which legal texts are converted into decision trees or algorithms that are operationally convenient but jurisprudentially fragile.

The relevance of computational law is therefore not confined to implementation. It also bears on the normative legitimacy of digital legal systems. Once legal rules are encoded, the design choices embedded in that encoding determine how ambiguity is resolved, how conflicts are prioritized, and what counts as a sufficient justification for a legal conclusion. These are substantive jurisprudential questions. For that reason, debates about (weak)

***Corresponding author:** Guido Governatori, School of Engineering and Technology, Central Queensland University, Australia. Email: g.governatori@cqu.edu.au

© The Author(s) 2026. Published by BON VIEW PUBLISHING. This is an open access article under the CC BY License (https://creativecommons.org/licenses/by/4.0/).

permission, conflict resolution, and deontic reasoning are not abstract side issues; they go to the heart of whether Rules as Code can deliver systems that are not only functional, but also legally faithful, transparent, and contestable. A mature account of Rules as Code must therefore draw on computational law not simply as a toolkit for automation but as a framework for understanding the structure and limits of machine-processable legality.

In the rest of the paper, we focus on the issue of weak permission. First (Section 2), we introduce its definition, and we discuss its relevance to legal reasoning. Then, we look at the issue of weak permission from two popular paradigms for non-monotonic reasoning and some of the most adopted semantics. More specifically, in Section 3, we examine logic programming under the well-founded [3] and stable-model semantics [4]. Then, in Section 4, we study how to integrate deontic reasoning in (structured) argumentation [5], and we look at it through the lens of grounded and stable semantics for argumentation [6]. Given the close relationships between logic programming and argumentation and their semantics, we get the same result: weak permission is not supported by well-founded, grounded, and (skeptical) stable semantics when there are deontic conflicts. Finally, in Section 5, we discuss some related work.

2. Weak Permission in Legal Reasoning

Deontic Logic is the branch of logic that investigates the logical behavior of obligations, permissions, prohibitions, and related notions [7]. In general, systems of deontic logic take obligation as primitive and leave the others as derived from obligations [7]. On the other hand, formal approaches to normative reasoning/legal theory (see, for instance [8–11]) often distinguish between two notions of permission: *Strong Permission* and *Weak Permission*. While the definitions of the types of permission vary, and other notions of permission have been proposed (for a discussion on the topic, see [12]. Often a strong permission models a derogation from a prohibition or the obligation to the contrary [8, 9, 13, 14], and we have weak permission when we fail to obtain the obligation of the contrary.

Another way to look at the issue is whether there are norms that explicitly permit something. If there are and the norms are effective, then we obtain an explicit (strong) permission. Notice that the two ways to understand strong permission are strongly correlated. First of all, to derogate a norm, we need to have a second norm that explicitly derogates the first one. And if the first norm is for an obligation, the derogating norm is either for a permission or the obligation of the contrary. However, the second derogating norm must be effective to derogate the first one. Accordingly, to have a strong permission, we need an explicit and effective permissive norm. In addition, the existence of a permissive norm does not necessarily imply that the norm is effective, and we can still have a weak permission when the permissive norm is not effective.

Thus, we have a weak permission if no norms make the opposite obligatory. But what if there are norms for the opposite that are not effective? Here we have two cases.

1) The first case is when there is a norm for the opposite, but the norm is not applicable (namely, the conditions to apply the norm do not hold). Effectively, we can treat this situation as one without norms.
2) The second case is when there are norms for the obligation of the opposite, but these norms conflict with some other norms for the conclusion.

So, the basic situation here is that we have a norm for $\mathsf{obl}(a)$ (a is obligatory) and a second norm for $\mathsf{obl}(\neg a)$ (a is forbidden, or $\neg a$ is obligatory), and there is no *prima facie* mechanism to resolve the conflict. There are a few options.

1) The first is to recognize that there is a conflict and prevent any conclusion from following from the conflict. In this case, we cannot assert that we have a weak permission.
2) The second option is to endorse the conflict as the failure to obtain the obligations and to accept the corresponding weak permissions.

Let us illustrate the situation with an example inspired by a real-life case.

Example 1. *In 2019, the Sea Watch 3, an NGO vessel, was on a mission to rescue migrants in the Mediterranean Sea. However, the Italian Government issued a decree that banned the Sea Watch 3 from rescue operations (for migrants) in the Italian Contiguous Zone waters.*

$$r_1\colon\ distress, proximity \Rightarrow \mathsf{obl}(assistance)$$

$$r_2\colon\ SeaWatch, migrants, ItalianContiguousZone \Rightarrow \mathsf{obl}(\neg assistance)$$

Maritime Law stipulates that a vessel in proximity to a distressed vessel must assist the vessel in distress (rule r_1). On the other hand, r_2 encodes the prohibition on rescuing migrants in Italian Contiguous Zone waters. Moreover, Maritime Law requires vessels permitted to refrain from assisting vessels in distress to alert the closest relevant authorities and keep clear of the rescue area.

$$r_3\colon\ \mathsf{perm}(\neg assistance) \Rightarrow \mathsf{obl}(alertAuthorities) \,\&\, \mathsf{obl}(keepClear)$$

Suppose a migrant vessel is in distress in the Italian Contiguous Zone, and Sea Watch 3 is nearby. Does Sea Watch 3 have the obligation to alert the authorities and keep clear of the rescue area? Here, two opposite norms conflict, and it is unclear which one takes precedence over the other.[1] *Also, suppose that Sea Watch 3 alerted the competent authorities but remained in the operation area (and eventually assisted with the rescue operation). Do the Sea Watch 3 actions contravene the norms? As we said, there are two options.*

1) *The permission does not follow from the conflict. In this case, the Sea Watch is not required to leave the area, and it does not contravene the norms.*
2) *In the second approach, the (weak) permission follows from the conflict, and consequently, leaving the area is a legal requirement, and Sea Watch 3 does not comply with the norms.*

What about if weak permission does not follow from a conflict? Do we have to consider the obligation to alert the authorities and keep clear of the rescue area? Strictly speaking, following the encoding in r_3 the answer is negative. However, we can consider an alternative formulation of the last provision. Assume that it stipulates that vessels for which the obligation to assist does not hold

[1] In response to the Sea Watch 3 incident, an Italian Tribunal established that International Maritime Law prevailed over the Italian Government Decree for the specific case. However, International Maritime Law scholars debated over the proper course of action.

must alert the closest competent authorities and keep clear of the rescue area.

r_3': not obl(*assistance*) ⇒ obl(*alertAuthorities*)& obl(*keepClear*)

Given the conflict, we can argue that the obligation to offer assistance does not hold, and the Sea Watch 3 has to follow the obligations given by r_3'.

The example above illustrates a feature of legal and normative reasoning: obligations, permissions, and prohibitions (and the lack of them) can trigger other obligations, permissions, and prohibitions. For example, Governatori and Rotolo [15] argue that for some legal applications, we "stop" the conflict but still use it (in this case, the failure to derive an obligation) to obtain further conclusions. Logics and formalisms for deontic reasoning must be able to handle such cases. Furthermore, it has been argued (e.g., by [16]) that, in general, deontic reasoning is an essential component of normative reasoning. At the same time, normative reasoning is non-monotonic by nature (specifically, defeasibility is a key aspect of normative and legal reasoning). Indeed, many works propose to integrate and ground deontic and normative reasoning in non-monotonic reasoning. Too many works to mention; we limit ourselves to the seminal proposal by Sergot et al. [17] to use logic programming to model norms (albeit with the severe limitation that it lacks deontic reasoning), the approach by Horty [18] to study deontic logic through the lens of non-monotonic reasoning, and the survey on formal argumentation approaches to legal reasoning by Prakken and Sartor [19]. Two common themes have emerged: approaches either do not incorporate deontic features or fail to address the suitability of the underlying formalisms, logics, and semantics for deontic reasoning.

As we have seen in the example above, deontic conflicts can arise in legal reasoning. The question, then, is how to interpret situations where there is a deontic conflict. First of all, legal reasoning has developed various techniques for dealing with conflicts, for example, by establishing priorities among norms according to various criteria (e.g., *lex superior, lex specialis, lex posterior*, etc.). However, there are situations where no clear priority can be established. How should we proceed in these cases?

To address this issue, we consider the *Principles of Legality* introduced by Fuller [20]. Fuller's Principles of Legality provide fundamental criteria for evaluating the legitimacy of legal systems and norms. While the principles were originally developed in the context of legal philosophy to support the moral, political, and societal legitimacy of legal systems, they have recently been proposed as a framework to evaluate the legitimacy of technical systems [21–23]. We believe that these principles are particularly relevant to deontic reasoning in the presence of conflicts. Fuller identifies eight principles:

1) *Generality*: Laws should be general rules, not arbitrary commands directed at specific individuals.
2) *Promulgation*: Laws must be publicly announced and made known.
3) *Non-retroactivity*: Laws should not be applied retroactively to conduct prior to their enactment (application).
4) *Clarity*: Laws must be clearly stated so that subjects can understand what is required.
5) *Non-contradiction*: The legal system should not contain contradictory norms.
6) *Possibility of compliance*: Laws must not demand the impossible.
7) *Stability*: Laws should remain stable over time, not constantly changing.
8) *Congruence*: There should be alignment between the official rules and their actual administration.

Suppose a set of norms contains two conflicting norms. One for the obligation of A and another for the obligation of $\neg A$. If there is no mechanism to resolve the conflict. Here, we have a few options:

1) Conclude both obligations
2) Derive one of the obligations
3) Prevent the derivation of both obligations

If we adopt the first option, we violate Fuller's principle of non-contradiction (principle 5) and the principle of possibility of compliance (principle 6), given that it is impossible for a subject to comply with both obligations simultaneously, rendering the system incoherent from a legality perspective.

If we go for the second option, then either we assume that there was a conflict-resolution mechanism (introducing a priority among norms), and in this case, we violate the principle of promulgation (principle 2), given that a resolving mechanism was not made explicit, or we violate the principle of clarity (principle 4), because the subjects cannot know which norm prevails.

If no (general) mechanism to resolve the conflict is available, and one of the obligations is derived, then the system fails to comply with the principle of generality (principle 1), as the resolution of the conflict would depend on specific circumstances rather than general rules; the principle of non-retroactivity (principle 3), as subjects cannot anticipate which obligation will prevail in future conflicts; and the principle of stability (principle 7), as the resolution of conflicts may vary over time without a consistent framework. Moreover, it potentially violates the principle of congruence (principle 8) given that the official rules may not align with their administration in practice.

Finally, the only option that allows us to comply with Fuller's principles of legality is to prevent the derivation of both obligations in the presence of unresolved conflicts. This observation motivates our investigation: if weak permission is understood as the failure to derive an obligation, then, in the presence of unresolved deontic conflicts, neither the obligation nor its negation can be established definitively.

Our formal analysis demonstrates that standard non-monotonic reasoning semantics cannot properly accommodate weak permission under these conditions, revealing a fundamental tension between the logical treatment of deontic conflicts and the normative requirements of legitimate legal systems.

3. Logic Programming and Weak Permission

The language of a deontic program is built from a set of literals (Lit), where a literal is either an atom (l) or its negation ($\neg l$). In addition, we extend the language with *deontic literals*. The set of deontic literals is defined as

$$\{\mathsf{dop}(l), \mathsf{not}\,\mathsf{dop}(l) | l \in \mathrm{Lit}\}$$

where dop is a deontic operator, more precisely

$$\mathsf{dop} \in \{\mathsf{obl}, \mathsf{perm}, \mathsf{perm}_\mathsf{w}\}.$$

The language also admits negation as failure (not). Literals and deontic literals can appear in the scope of not, but negation as failure cannot appear in the scope of a deontic literal (this

is a standard condition in logic programming and Answer Set Programming, since a deontic operator is a predicate in such formalism, and negation as failure cannot occur inside predicates). Given a literal l, we use $\sim l$ to denote the complement of l; more precisely, if l is an atomic proposition, then $\sim l = \neg l$. If l is a negated atomic proposition $l = \neg m$, then $\sim l = m$.

Definition 1 (Program). *A program is a set of clauses or rules, where a rule r is an expression*

$$c \leftarrow a_1, \ldots, a_n, \mathsf{not}\, b_1, \ldots, \mathsf{not}\, b_m.$$

where $a_1, \ldots, a_n, b_1, \ldots, b_m, c$ *are either literals or deontic literals.*

Given a rule r, c is called the head of the rule, noted $h(r)$, and

$$\{a_1, \ldots, a_n, b_1, \ldots, b_m\}$$

is the body of r (noted $b(r)$). We split the body into the positive part

$$b^+(r) = \{a_1, \ldots, a_n\}$$

and the negative part

$$b^-(r) = \{\mathsf{not}\, b_1, \ldots, \mathsf{not}\, b_m\}.$$

The head and the body can be empty. We will refer to a rule with an empty head as an *integrity constraint*.

The *Herbrand Base* of a program P (noted as HB_P) is the set of all literals and deontic literals appearing in P.

Definition 2 (Basic Deontic Program). *A basic deontic program is a program with the restriction that no head of a rule is a weak permission; that is, no rule has the form* $c \leftarrow \mathsf{not}\, b_1, \ldots, \mathsf{not}\, b_m$ *where* $c = \mathsf{perm_w}(l)$ *for some literal l.*

As we discussed above, the distinction between strong permission and weak permission is that a strong permission is an explicit permission, while we have a weak permission when there are no explicit effective rules for it or when we fail to derive the obligation of the contrary. Thus, the restriction that no rule has a weak permission as its head is a way to ensure that there are no explicit rules for the weak permission, because if there were, we would have a rule for a strong permission.

The idea is to encode a set of (legal) norms in a basic deontic program and then extend the program with the set of rules and integrity constraints above to create a corresponding deontic program. Clauses (1)–(4) are meant to capture and implement the properties of deontic reasoning (as we just discussed), while the rules in the basic deontic program encode the conditions of the legal norms.

To capture deontic reasoning, we introduce deontic programs, namely, programs extended with a set of additional clauses encoding key properties of deontic reasoning.

Definition 3 (Deontic Program). *A deontic program is a program including the following clauses and integrity constraints:*

$$\mathsf{perm}(X) \leftarrow \mathsf{obl}(X). \tag{1}$$

$$\mathsf{perm_w}(X) \leftarrow \mathsf{not}\, \mathsf{obl}(\sim X). \tag{2}$$

$$\leftarrow \mathsf{obl}(X), \mathsf{obl}(\neg X). \tag{3}$$

$$\leftarrow \mathsf{obl}(X), \mathsf{perm}(\sim X). \tag{4}$$

The rule in (1) corresponds to the D axioms of Standard Deontic Logic, where an obligation implies the corresponding permission. Clause (2) establishes that we have a weak permission if we fail to derive the obligation to the contrary. The two integrity constraints ensure the (deontic) consistency of a deontic program. So, according to the first integrity constraint, no proposition can be at the same time obligatory and forbidden. Similarly, according to (4), no literal can be obligatory (forbidden) when its opposite is permitted.

From the first integrity constraint, we have that no program with the following two rules

$$\mathsf{obl}(p) \leftarrow .$$
$$\mathsf{obl}(\neg p) \leftarrow .$$

is satisfiable. Accordingly, to have a satisfiable deontic conflict, we have to encode the rules for obligation as follows:

$$\mathsf{obl}(p) \leftarrow \mathsf{not}\ \mathsf{obl}(\neg p).$$
$$\mathsf{obl}(\neg p) \leftarrow \mathsf{not}\ \mathsf{obl}(p).$$

Notice that the construction above is the standard technique in logic programming (in particular, Answer Set Programming) to prevent an inconsistency. Effectively, it means that we can assert $\mathsf{obl}(p)$ provided that the conditions for all the clauses supporting $\mathsf{obl}(\neg p)$ fail to succeed (the argument for $\mathsf{obl}(\neg p)$ is symmetrical).

Example 2. *Based on the discussion so far, let us see how to encode the scenario depicted in Example 1. The basic deontic program consists of the following rules:*

$$\mathsf{obl}(\mathit{assistance}) \leftarrow \mathsf{not}\ \mathsf{obl}(\neg \mathit{assistance}), \mathit{distress}, \mathit{proximity}.$$
$$\mathsf{obl}(\neg \mathit{assistance}) \leftarrow \mathsf{not}\ \mathsf{obl}(\mathit{assistance}), \mathit{SeaWatch}, \mathit{migrants}, \mathit{ItalianContiguousZone}.$$

The corresponding program is obtained by adding clauses (1)–(4) to the basic deontic program. We include only the instances where the variable X is instantiated with a literal appearing in a deontic literal in the base program.

$$\mathsf{perm}(\mathit{assistance}) \leftarrow \mathsf{obl}(\mathit{assistance}).$$
$$\mathsf{perm}(\neg \mathit{assistance}) \leftarrow \mathsf{obl}(\neg \mathit{assistance}).$$
$$\mathsf{perm_w}(\mathit{assistance}) \leftarrow \mathsf{not}\, \mathsf{obl}(\neg \mathit{assistance}).$$
$$\mathsf{perm_w}(\neg \mathit{assistance}) \leftarrow \mathsf{not}\, \mathsf{obl}(\mathit{assistance}).$$
$$\leftarrow \mathsf{obl}(\mathit{assistance}), \mathsf{obl}(\neg \mathit{assistance}).$$
$$\leftarrow \mathsf{obl}(\mathit{assistance}), \mathsf{perm}(\neg \mathit{assistance}).$$
$$\leftarrow \mathsf{obl}(\neg \mathit{assistance}), \mathsf{perm}(\mathit{assistance}).$$

The way a set of conclusions is derived from a deontic program depends on the semantics adopted. In the following, we will look at two of the most popular semantics for logic programming: well-founded [3] and stable semantics [24].

The intuition behind well-founded semantics is that it tries to draw all conclusions that are definitely justified, reject all conclusions that are definitely false, and leave everything else undetermined. The semantics assigns atoms to true, false, or undefined by accepting only conclusions supported by a non-circular foundation, rejecting atoms whose support is purely circular, and leaving unresolved negative cycles as undefined. This makes it a conservative, skeptical semantics that extracts the maximum amount of information that is unequivocally justified by the program.

On the other hand, the intuition behind stable semantics is that it tries to find self-consistent sets of conclusions. A stable model is a set of atoms that is justified by the program when

assuming that all atoms not in the set are false. This semantics can yield multiple stable models, each representing a different consistent way to interpret the program. Alternatively, it makes hypotheses about what is true and keeps it only if the program itself justifies exactly that hypothesis. This makes it a more credulous semantics that can capture multiple, potentially conflicting, perspectives on the program's conclusions. However, we will focus on the skeptical interpretation of stable semantics, which only accepts conclusions that are true in all stable models.

We are now ready to introduce the semantic framework for deontic programs. We follow the presentation by Caminada et al. [25]. While the approach is not standard, its advantage is that it provides a unified view of well-founded and stable semantics, where the well-founded semantics can be seen as a 3-valued variant of the 2-valued stable semantics as proposed by Przymusinski [24].

Definition 4. *A 3-valued Herbrand Interpretation I of a logic program P is a pair $\langle T, F\rangle$ with $T, F \subseteq HB_P$ and $T \cap F = \emptyset$. The atoms in T are said to be true, the atoms in F are said to be false, and the atoms in $HB_P \setminus (T \cup F)$ are said to be undefined.*

Definition 5. *Let I be a 3-valued Herbrand Interpretation of the logic program P. The reduct of P with respect to I (written as P/I) is the logic program constructed using the following steps.*

1. Starting from P, remove each rule r from P that has not $b_i \in b^-(r)$ for some $b_i \in T$;

2. From the result of step 1, for each rule, for every $b_i \in F$, remove not b_i from the body of the rule;

3. From the result of step 2, replace any remaining occurrences of not b_i by **u**.

where **u** *is an atom not in HB_P which is undefined in all interpretations of P (a constant). $\Psi_P(I) = \langle T_\Psi, F_\Psi\rangle$ with minimal T_Ψ and maximal F_Ψ (w.r.t. set inclusion) is the unique least 3-valued model of P such that, for every $a \in HB_P$:*

- *$a \in T_\Psi$ if there is a rule $r' \in P/I$ with $h(r') = a$ and $b^+(r') \subseteq T_\Psi$;*
- *$a \in F_\Psi$ if every rule $r' \in P/I$ with $h(r') = a$ has $b^+(r') \cap F_\Psi \neq \emptyset$.*

Example 3. *Consider the deontic program P consisting of the rules:*

$$\begin{aligned} a &\leftarrow \mathsf{not}\, b.\\ b &\leftarrow \mathsf{not}\, a.\\ c &\leftarrow .\\ d &\leftarrow c.\\ e &\leftarrow \mathsf{not}\, c, b.\\ f &\leftarrow c, e. \end{aligned}$$

Let $I = \langle\{c\}, \emptyset\rangle$ be a 3-valued interpretation. Applying the reduct construction, we obtain:

$$\begin{aligned} a &\leftarrow \mathbf{u}.\\ b &\leftarrow \mathbf{u}.\\ c &\leftarrow .\\ d &\leftarrow c.\\ f &\leftarrow c, e. \end{aligned}$$

Notice that the rule for e has been removed since $c \in T$ and notc appears in the body of the rule. Computing $\Psi_P(I)$, we find $T_\Psi = \{c, d\}$ (rules for c and d have all positive premises satisfied) and $F_\Psi = \{e, f\}$ (there are no rules for e, and the rule for f has an unsatisfiable body: $e \in F_\Psi$). Thus $\Psi_P(I) = \langle\{c, d\}, \{e, f\}\rangle$.

The proof of Theorem 1 provides an example of a reduct and computation of $\Psi_P(I)$ for the deontic program

$$\begin{aligned} \mathsf{obl}(l) &\leftarrow \mathsf{not}\, \mathsf{obl}(\neg l).\\ \mathsf{obl}(\neg l) &\leftarrow \mathsf{not}\, \mathsf{obl}(l). \end{aligned}$$

Definition 6 (Logic Programming Semantics). *Let $I = \langle T, F\rangle$ be a 3-valued Herbrand Interpretation of logic program P.*

- *I is a partial stable (or P-stable) model of P iff $\Psi_P(I) = I$.*
- *T is a well-founded model of P iff I is a P-stable model of P where T is minimal (w.r.t. set inclusion) among all P-stable models of P.*
- *T is a (2-valued) stable model of P iff I is a P-stable model of P where $T \cup F = HB_P$.*
- *Let $\mathscr{I} = \{I_i = \langle T_i, F_i\rangle : \text{ such that } T_i \text{ is a stable model of } P\}$. $T = \bigcap_{I_i \in \mathscr{I}} T_i$ is the skeptical stable model of P.*

Example 4. *Let us consider the deontic program P of Example 3. First of all, it is easy to verify that $I = \langle\{c, d\}, \{e, f\}\rangle$ is a P-stable model of P. Indeed*

$$\Psi_P(\langle\{c, d\}, \{e, f\}\rangle) = \langle\{c, d\}, \{e, f\}\rangle\,,$$

and $T = \{c, d\}$ is the minimal model.
Given that the Herbrand Base of P is

$$HB_P = \{a, b, c, d, e, f\},$$

for the stable semantics, we have two stable models:

$$I_1 = \langle\{a, c, d\}, \{b, e, f\}\rangle\; I_2 = \langle\{b, c, d\}, \{a, e, f\}\rangle\,.$$

The reduct under I_1 is

$$\begin{aligned} a &\leftarrow .\\ c &\leftarrow .\\ d &\leftarrow c.\\ f &\leftarrow c, e. \end{aligned}$$

while the reduct under I_2 is

$$\begin{aligned} b &\leftarrow .\\ c &\leftarrow .\\ d &\leftarrow c.\\ f &\leftarrow c, e. \end{aligned}$$

Definition 7 (Conflicted Deontic Program). *A deontic program P is conflicted if, for a literal l, it contains the rules*

$$\mathsf{obl}\,(l) \leftarrow \mathsf{not}\,\mathsf{obl}\,(\neg l)\,.\quad \mathsf{obl}(\neg l) \leftarrow \mathsf{not}\,\mathsf{obl}(l).$$

We say that l is a conflicted literal.

Theorem 1. *Let P be a conflicted deontic program, and let l be a conflicted literal. Then $\mathsf{perm}_\mathsf{w}(l)$ and $\mathsf{perm}_\mathsf{w}(\neg l)$ are not conclusions of the program under well-founded and stable semantics.*

Proof. *To prove the property, we can assume, without any loss of generality, that the language is restricted to the conflicted literal l, the Herbrand Base consists of the deontic literals that can be built from l, that is,*

$$HB_P = \{\mathsf{obl}(l), \mathsf{obl}(\neg l), \mathsf{perm}(l), \mathsf{perm}(\neg l), \mathsf{perm}_\mathsf{w}(l), \mathsf{perm}_\mathsf{w}(\neg l)\},$$

and the program consists of the two rules such that l is conflicted plus the instantiation of clauses (1)–(4).

First, we show that $I = \langle \emptyset, \emptyset \rangle$ *is a partial stable model. By Definition 5, P/I contains the following rules (for the sake of clarity and conciseness, we removed the instances of the integrity constraints):*

$$\mathsf{perm}(l) \leftarrow \mathsf{obl}(l). \quad \mathsf{perm_w}(l) \leftarrow \mathbf{u}. \quad \mathsf{obl}(l) \leftarrow \mathbf{u}.$$
$$\mathsf{perm}(\neg l) \leftarrow \mathsf{obl}(\neg l). \ \mathsf{perm_w}(\neg l) \leftarrow \mathbf{u}. \ \mathsf{obl}(\neg l) \leftarrow \mathbf{u}.$$

It is easy to verify that $T_\Psi = \emptyset$ *and* $F_\Psi = \emptyset$*. Clearly,* $T_\Psi = \emptyset$ *is minimal. Let us show that* $F_\Psi = \emptyset$ *is maximal. Given the integrity constraints (3) and (4), it is not possible that* $\mathsf{obl}(l)$ *and* $\mathsf{obl}(\neg l)$ *are true in the same interpretation; similarly, for* $\mathsf{obl}(l)$ *and* $\mathsf{perm}(\sim l)$*, and by rule (2),* $\mathsf{obl}(l)$ *and* $\mathsf{perm_w}(\neg l)$ *cannot be both true.*

Suppose that there is an interpretation $I' = \langle \emptyset, F' \neq \emptyset \rangle$*. Moreover, assume* $\mathsf{obl}(l) \in F$*. Then* P/I' *contains the rule* $\mathsf{obl}(\neg l) \leftarrow$*. Thus,* $\mathsf{obl}(\neg l) \in T_\Psi$*, and so,* $I' \neq \Psi_P(I')$*. Suppose* $\mathsf{perm}(l) \in F'$*, given the rule* $\mathsf{perm}(l) \leftarrow \mathsf{obl}(l)$*.* $\mathsf{obl}(l)$ *must be in* F' *as well, and we can repeat the argument above. Finally, if* $\mathsf{perm_w}(l)$ *were in* F'*, then* P/I' *should either not contain any rule for* $\mathsf{perm_w}(l)$*, but this means that* $\mathsf{obl}(\neg l) \in T$ *(which is not the case); or* $\mathsf{perm_w}(l) \leftarrow \mathbf{u} \in P/I'$*, and then* $\mathbf{u} \in F'$*; however, as stipulated,* $\mathbf{u}$ *does not belong to* HB_P*, and so, it is not in* F'*. The proof for deontic literals based on* $\neg l$ *is the same. Accordingly, we have shown that* $F_\Psi = \emptyset$ *is maximal, and so,* T_Ψ *is the well-founded model of P, and* $\mathsf{perm_w}(l), \mathsf{perm_w}(\neg l) \notin T_\Psi$*.*

For the stable semantics, we are looking for interpretations where the elements of the (deontic) Herbrand Base are distributed over the set of true and false literals. As noted above, in one interpretation, $\mathsf{obl}(l)$ *and* $\mathsf{obl}(\neg l)$ *cannot be true in the same interpretation. So, one of them must be false; let us say* $\mathsf{obl}(l) \in F$*. Then P/I contains the rule* $\mathsf{obl}(\neg l) \leftarrow$*. Thus, in that interpretation,* $\mathsf{obl}(\neg l) \in T$*. Accordingly, we have two interpretations:* $I_1 = \langle T_1, F_1 \rangle$*, and* $I_2 = \langle T_2, F_2 \rangle$*, where* $\mathsf{obl}(l) \in T_1$ *and* $\mathsf{obl}(\neg l) \in T_2$*. Hence, the two reducts are*

$$\begin{aligned} P/I_1 = \{&\mathsf{perm}(l) \leftarrow \mathsf{obl}(l). \\ &\mathsf{perm}(\neg l) \leftarrow \mathsf{obl}(\neg l). \\ &\mathsf{perm_w}(l) \leftarrow . \\ &\mathsf{obl}(l) \leftarrow .\} \\ P/I_2 = \{&\mathsf{perm}(l) \leftarrow \mathsf{obl}(l). \\ &\mathsf{perm}(\neg l) \leftarrow \mathsf{obl}(\neg l). \\ &\mathsf{perm_w}(\neg l) \leftarrow . \\ &\mathsf{obl}(\neg l) \leftarrow .\} \end{aligned}$$

and the complete interpretations are

$$T_1 = F_2 = \{\mathsf{obl}(l), \mathsf{perm}(l), \mathsf{perm_w}(l)\}$$
$$F_1 = T_2 = \{\mathsf{obl}(\neg l), \mathsf{perm}(\neg l), \mathsf{perm_w}(\neg l)\}$$

Thus, we have two stable models, one containing $\mathsf{perm_w}(l)$ *and the other containing* $\mathsf{perm_w}(\neg l)$*. Therefore,* $\mathsf{perm_w}(l)$ *and* $\mathsf{perm_w}(\neg l)$ *are not skeptical conclusions under the stable semantics.*

For the sake of simplicity, we restricted the definition of conflicted deontic programs to the case where the program contains two rules $\mathsf{obl}(l) \leftarrow \mathsf{not}\,\mathsf{obl}(\neg l)$ and $\mathsf{obl}(\neg l) \leftarrow \mathsf{not}\,\mathsf{obl}(l)$. However, the result of Theorem 1 can be easily extended to more general cases where the two obligations are supported by more complex rules. More specifically, the results can be extended as follows.

Definition 8. *A deontic program P is conflicted under an interpretation I (*$I = \langle T, F \rangle$*) if, for a literal l, the reduct of the program under I contains the rules*

$$\mathsf{obl}(l) \leftarrow \mathbf{u}, a_1, \ldots, a_n.$$
$$\mathsf{obl}(\neg l) \leftarrow \mathbf{u}, b_1, \ldots, b_m.$$

for the well-founded semantics, and

$$\mathsf{obl}(l) \leftarrow a_1, \ldots, a_n, \mathsf{not}\,\mathsf{obl}(\neg l), \mathsf{not}\, c_1, \ldots \mathsf{not}\, c_k.$$
$$\mathsf{obl}(\neg l) \leftarrow b_1, \ldots, b_m, \mathsf{not}\,\mathsf{obl}(l)\, \mathsf{not}\, d_1, \ldots \mathsf{not}\, d_j.$$

such that $\{a_1, \ldots, a_n, b_1, \ldots, b_m\} \subseteq T$ *and* $\{c_1, \ldots c_k, d_1 \ldots, d_j\} \subseteq F$ *for the stable semantics.*

According to this definition, the deontic program in Example 1 is conflicted under an interpretation I, where the T contains the literals *distress*, *proximity*, *SeaWatch*, *migrants*, and *ItalianContiguousZone* and F is empty.

Theorem 2. *Let P be a conflicted deontic program under an interpretation I, and let l be a conflicted literal. Then* $\mathsf{perm_w}(l)$ *and* $\mathsf{perm_w}(\neg l)$ *are not conclusions of the program under well-founded and stable semantics based on the interpretation I.*

Proof. Straightforward from the proof of Theorem 1, given that in the reduct under I, we have the same structure as in the proof of Theorem 1.

4. Argumentation and Weak Permission

For the language of our deontic argumentation framework, we use the same language $\mathscr{L}$ as the previous part, but we do not admit negation as failure. Arguments are built from rules, where a rule has the following format:

$$a_1, \ldots a_n \Rightarrow c$$

where $\{a_1, \ldots, a_n\}$ is a (possibly empty) set of literals and deontic literals and c is either a literal or a deontic literal but not a weak permission (i.e., $c \neq \mathsf{perm_w}(l)$; otherwise, it would be an explicit permission).

Definition 9. *A Deontic Argumentation Theory is a structure*

$$(F, R)$$

where F is a (finite and possibly empty) set of literals (the facts or assumptions of the theory) and R is a (finite) set of rules.

The key concept of an argumentation theory is the notion of an argument. Definition 10 below defines what an argument is. Each argument A has associated with it its conclusion $C(A)$ and its set of sub-arguments $Sub(A)$.

Definition 10. *Given a Deontic Argumentation Theory* (F, R)*, A is an argument if A has one of the following forms:*

1) $A = \mathsf{perm_w}(l)$ *for any literal* $l \in Lit$*, the conclusion of the argument* $C(A) = \mathsf{perm_w}(l)$*, and* $Sub(A) = \{A\}$*.*
2) $A = a$ *for* $a \in F$*;* $C(A) = a$ *and* $Sub(A) = \{A\}$*.*
3) $A = A_1, \ldots, A_n \Rightarrow c$*, if there is a rule* $a_1, \ldots, a_n \Rightarrow c$ *in R such that for all* $a_i \in \{a_1, \ldots, a_n\}$*, there is an argument* A_i *such that* $C(A_i) = a_i$*;* $C(A) = c$ *and* $Sub(A) = \{A\} \cup Sub(A_1) \cup \cdots \cup Sub(A_n)$*.*

4) $A = B \Rightarrow \mathsf{perm}(l)$, *if* B *is an argument such that* $C(B) = \mathsf{obl}(l)$; $C(A) = \mathsf{perm}(l)$, *and* $Sub(A) = \{A\} \cup Sub(B)$.

Condition 1) encodes the idea that weak permission is the failure to derive an obligation to the contrary (more on this when we discuss the notion of attack between arguments). Thus, by default, every literal is potentially weakly permitted, and we form an argument for this type of conclusion. Condition 2) gives the simplest form of an argument. We have an argument for a if a is one of the assumptions/facts of a case/theory. Condition 3) allows us to form arguments by forward chaining rules. Thus, we can form an argument from a rule if we have arguments for all the elements of the body of the rule. The way the condition is written allows us to create arguments from rules with an empty body. Finally, condition 4) corresponds to the D axiom of Standard Deontic Logic.

Definition 11 (Attack). *Let A and B be arguments. A attacks B (*$A > B$*) iff*

1) $B = \mathsf{perm}_{\mathsf{w}}(l)$ *and* $C(A) = \mathsf{obl}(\sim l)$;
2) $\exists B' \in Sub(B)$, $C(B') = l$ *and* $C(A) = \sim l$;
3) $\exists B' \in Sub(B)$, $C(B') \in \{\mathsf{obl}(l), \mathsf{perm}(l)\}$ *and* $C(A) = \mathsf{obl}(\sim l)$;
4) $\exists B' \in Sub(B)$, $C(B') = \mathsf{obl}(l)$, *and* $C(A) = \mathsf{perm}(\sim l)$.

We use $\mathscr{A}$ *to denote the set of all arguments of the theory.*

As we alluded to above and as we have seen in Section 3, the idea of weak permission is the negation as failure of the obligation to the contrary. Thus, in Definition 10, we create an argument for the weak permission for any literal l; however, this argument is attacked by any argument for $\mathsf{obl}(\sim l)$ (condition 1 above, and notice this is the only case where the attack is not symmetrical). The rest of the conditions define an attack when the two arguments have opposite conclusions: condition 2 covers the case of plain literals, while conditions 3 and 4 are reserved for deontic literals. Specifically, we have opposite deontic conclusions when one of the two is an obligation for a literal, and the other is either an obligation or a permission for the opposite literal. Furthermore, an argument attacks another argument when the conflict is on the conclusion of the second argument (this corresponds to the notion of rebuttal) or when there is a conflict with one of the sub-arguments of the attacked argument (known as an undercutting attack).

Similarly to logic programming, the way a set of arguments is evaluated depends on the semantics adopted. In the following, we will look at Dung's semantics [6]. Again, we focus on some of the most popular semantics: grounded and stable semantics; moreover, as we did for logic programming, we concentrate on skeptical conclusions.

The key idea of the grounded semantics is to accept only those arguments that can be justified without making any controversial assumptions. Grounded semantics is the most skeptical argumentation semantics. It builds acceptance from the ground up, admitting only arguments whose acceptability can be established in an uncontroversial way. In other terms, it accepts arguments whose justification is ultimately grounded and leaves all unresolved conflicts undecided.

The intuition for the stable semantics in argumentation is that it tries to find self-consistent sets of arguments. A stable extension is a set of arguments that is justified when assuming that all arguments not in the set are defeated.

Definition 12 (Dung Semantics). *Let* (F, R) *be a Deontic Argumentation Theory, and let S be a set of arguments. Then:*

- S *is conflict-free iff* $\forall X, Y \in S : X \not> Y$.
- $X \in \mathscr{A}$ *is acceptable with respect to S iff* $\forall Y \in \mathscr{A}$ *such that* $Y > X : \exists Z \in S$ *such that* $Z > Y$.
- S *is an admissible set iff S is conflict free and* $X \in S$ *implies X is acceptable w.r.t. S.*
- S *is a complete extension iff S is admissible, and if* $X \in \mathscr{A}$ *is acceptable w.r.t. S, then* $X \in S$.
- S *is the grounded extension iff S is the set-inclusion-minimal complete extension.*
- S *is a stable extension iff S is conflict free and* $\forall Y \notin S$, $\exists X \in S$ *such that* $X > Y$.

Definition 13 (Justified Argument). *Let* $D = (F, R)$ *be a Deontic Argumentation Theory; an argument A is skeptically justified under a semantic T iff 1) an extension for D exists under T and 2)* $A \in S$ *for all sets of arguments S that are an extension under T.*

Definition 14 (Justified Conclusion). *A literal or a deontic literal* $l \in \mathscr{L}$ *is a justified conclusion under a semantics T iff 1) an extension for D exists under T and 2) for every extension S under T, there is an argument A such that* $A \in S$ *and* $C(A) = l$.

Let us consider a Deontic Argumentation Theory where $F = \emptyset$ and R contains the two rules

$$r_1 :\Rightarrow \mathsf{obl}(a) \quad r_2 :\Rightarrow \mathsf{obl}(\neg a)$$

This theory has the following arguments:

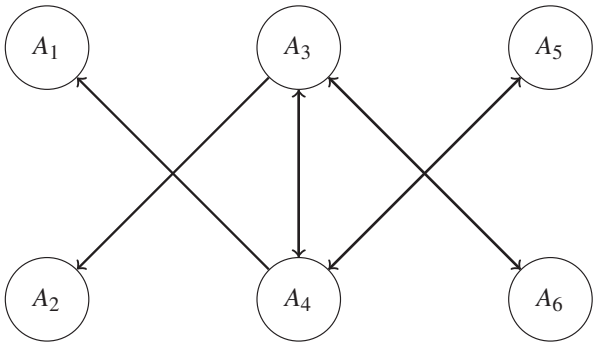
$$A_1 \colon \mathsf{perm}_{\mathsf{w}}(a) \quad A_3 :\Rightarrow \mathsf{obl}(a) \quad A_5 \colon A_3 \Rightarrow \mathsf{perm}(a)$$
$$A_2 \colon \mathsf{perm}_{\mathsf{w}}(\neg a) \quad A_4 :\Rightarrow \mathsf{obl}(\neg a) \quad A_6 \colon A_4 \Rightarrow \mathsf{perm}(\neg a)$$

For the attack relation, we have the following instances:

$$A_3 > A_2 \quad A_3 > A_4 \quad A_3 > A_6 \quad A_5 > A_4$$
$$A_4 > A_1 \quad A_4 > A_3 \quad A_4 > A_5 \quad A_6 > A_3$$

It is easy to verify that {} is a complete extension (and trivially, it is the minimal complete extension w.r.t. set inclusion). Thus, it is the grounded extension of the theory. Accordingly, there is no argument in the grounded extension such that its conclusion is either $\mathsf{perm}_{\mathsf{w}}(a)$ or $\mathsf{perm}_{\mathsf{w}}(\neg a)$. Hence, $\mathsf{perm}_{\mathsf{w}}(a)$ and $\mathsf{perm}_{\mathsf{w}}(\neg a)$ are not justified conclusions under the grounded semantics.

The following diagram depicts the argumentation graph of the theory (where the arrows represent the attack relation).



When we consider the stable semantics, we have the following two extensions:

$$\{A_1, A_3, A_5\} \quad \{A_2, A_4, A_6\}$$

Clearly, $\text{perm}_w(a)$ is a conclusion of the first extension but not of the second one; conversely, $\text{perm}_w(\neg a)$ is a conclusion of the second extension but not of the first one. Consequently, $\text{perm}_w(a)$ and $\text{perm}_w(\neg a)$ are not justified conclusions under the stable semantics.

The above example should suffice to show that weak permission is not supported by grounded and stable semantics when a deontic conflict exists: we have a scenario where we fail to conclude that an obligation, but at the same time, we cannot conclude the weak permission of the opposite. However, we can generalize the result; the result holds for any theory with a conflict between two applicable obligation rules. This is formalized by the following definition.

Definition 15. *A Deontic Argumentation Theory is conflictual when it contains a pair of rules* $b_1, \ldots, b_n \Rightarrow \text{obl}(c)$ *and* $d_1, \ldots d_m \Rightarrow \text{obl}(\neg c)$*, such that there are arguments* B_i *with conclusion* b_i*,* $1 \leq i \leq n$*, and* D_j *with conclusion* d_j*,* $1 \leq j \leq m$*. We will call c the conflicted literal.*

Theorem 3. *Let* $D = (F, R)$ *be a conflictual theory. For any conflicted literal l,* $\text{perm}_w(l)$*,* $\text{perm}_w(\neg l)$ *are not justified conclusions under grounded and stable semantics.*

Proof. Let l be a conflicted literal. By the definition of argument, the theory D and the fact that the theory is conflicted, we have the following arguments:

$$A_1 : \text{perm}_w(l) \quad A_3 : B_1, \ldots, B_n \Rightarrow \text{obl}(l)$$
$$A_2 : \text{perm}_w(\neg l) \quad A_4 : D_1, \ldots, D_m \Rightarrow \text{obl}(\neg l)$$

such that $A_3 > A_2$ and $A_4 > A_1$. Given that there are arguments attacking A_1 and A_2, these two arguments are not in the minimal complete extension. Accordingly, $\text{perm}_w(l)$ and $\text{perm}_w(\neg l)$ are not justified conclusions under the grounded semantics.

Given the attack relationship among A_1, A_2, A_3, and A_4, we can conclude that there are at least two extensions E_1 and E_2 such that $A_1, A_3 \in E_1$ and $A_2, A_4 \notin E_1$, and $A_2, A_4 \in E_2$ and $A_1, A_3 \notin E_2$. Hence, there is an extension where no argument has $\text{perm}_w(l)$ as its conclusion, and there is an extension where no argument has $\text{perm}_w(\neg l)$ as its conclusion. Therefore, $\text{perm}_w(l)$ and $\text{perm}_w(\neg l)$ are not justified conclusions under the stable semantics.

While other semantics have been proposed in the literature, they are not widely adopted for legal reasoning, or, for the argument of the paper, they can be seen as variants of the semantics we analyzed. For instance, for the preferred semantics [6], a preferred extension is a maximally complete extension, and the intersection of all preferred extensions corresponds to the grounded extension [6]. Thus, the impossibility result for the grounded semantics also applies to the preferred semantics.

5. Related Work

Recent work in Rules as Code and computational law has produced several formalisms for encoding norms in executable form. For a concise survey of some of the Rules as Code systems, see [26] and [27] for a survey of some argumentation-based approaches to legal reasoning. Two main issues nevertheless emerge from this literature. Either an approach does not support deontic concepts at all (see, e.g., [28–30]), or, when deontic concepts are supported, it is not clear that the approach is grounded in an adequate model of deontic reasoning (see, e.g., [31–34]). A notable exception is the work on Defeasible Deontic Logic [13].

The aim of Flood and Goodenough [28] is to show how to represent financial contracts using discrete finite automata (DFA). The paper also argues that deontic concepts are not needed to represent contracts and, accordingly, does not provide explicit support for deontic notions. While some contracts can be represented without deontic concepts, the contract discussed in that paper contains several deontic features. In addition, DFA-based representations can grow rapidly with the number of states, so it is not clear how well the approach scales to more complex contracts.

Catala and Merigoux's work [29] is a domain-specific language for building legal expert systems, particularly in taxation and social security. The language is based on a functional programming paradigm and provides strong support for numerical computation. However, it does not support deontic concepts directly and is therefore not well suited to representing explicitly normative situations involving obligations, permissions, and prohibitions.

Morris [30] aims to provide a visual programming language for encoding norms. However, the language does not support deontic concepts explicitly, and it is based on Answer Set Programming, which, as we have shown in this paper, does not provide a satisfactory model of weak permission in the presence of deontic conflict. A similar issue arises for s(LAW) [32], which, like Blawx, builds on the s(CASP) approach to logic programming [35]. Likewise, deolingo [34] extends Answer Set Programming with several deontic concepts, but it adopts the standard stable semantics. Similarly, deon-$\mathscr{B}$ [33] extends logic programming under the well-founded semantics with deontic concepts for deontic planning scenarios. Finally, Logical English [31] is a controlled natural language that has been proposed "as an alternative language to conventional legal English and to conventional computer languages for automating legal documents." Currently, Logical English can be translated to Prolog and Answer Set Programming. The common feature of these approaches is that the underlying semantics are affected by the impossibility results we have shown in this paper.

The impossibility results established in the previous sections apply to well-founded, grounded, and stable semantics, but they do not extend to all forms of non-monotonic reasoning. For example, the approach to deontic reasoning based on defeasible logic proposed by Governatori et al. [13] does not suffer from the weak-permission problem. In addition, these results do not apply to all argumentation semantics. Recently, Governatori and Rotolo [36] introduced a novel argumentation semantics that likewise avoids the weak-permission problem.

Answer Set Programming is based on stable-model semantics, and its direct use for deontic reasoning therefore inherits the difficulty with weak permission. However, Governatori's work [37] shows how Answer Set Programming can be used as a meta-programming framework to encode the reasoning mechanism of Defeasible Deontic Logic [13], thereby avoiding the weak-permission problem discussed in this paper.

6. Summary and Conclusions

We investigated the issue of weak permission, defined as the lack of the obligation to the contrary, in the context of some forms of non-monotonic reasoning (specifically, logic programming and structured argumentation) and some of the most adopted semantics (well-founded, grounded and stable). We proved that when the failure to obtain an obligation depends on an unsolved deontic conflict, the corresponding weak permission is not a conclusion under the well-founded, grounded, and (skeptical) stable

semantics. Accordingly, approaches to deontic reasoning adopting such semantics cannot offer a proper model of full deontic reasoning. This result suggests that Rules as Code systems and computational law approaches should be cautious in relying on these semantics for legal reasoning and should instead consider alternative semantics, such as the novel argumentation semantics proposed by Governatori and Rotolo [36].

Acknowledgments

The author thanks the anonymous reviewers for their valuable comments, criticisms, and suggestions on previous versions of this paper.

Generative AI tools were used to generate the TikZ diagram in Section 4, to create some BibTeX entries for the bibliography, and to check the text for grammar and typos. The author carefully reviewed and edited the content produced by the AI tools and takes full responsibility for the content of the publication.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by the author.

Conflicts of Interest

Guido Governatori is the Editor-in-Chief for *Journal of Computational Law and Legal Technology* and was not involved in the editorial review or the decision to publish this article, who has no other relevant affiliations or financial involvement with any organization or entity with a financial interest in or financial conflict with the subject matter or materials discussed in the manuscript apart from those disclosed. The author declares that he has no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Guido Governatori: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization.

References

[1] Waddington, M. (2021). Rules as code. *Law in Context*, *37*(1), 179–186. https://doi.org/10.26826/law-in-context.v37i1.134

[2] Mohun, J., & Roberts, A. (2020). Cracking the code: Rule making for humans and machines. *OECD Working Papers on Public Governance*. https://doi.org/10.1787/3afe6ba5-en

[3] van Gelder, A., Ross, K. A., & Schlipf, J. S. (1991). The well-founded semantics for general logic programs. *Journal of the ACM*, *38*(3), 619–649. https://doi.org/10.1145/116825.116838

[4] Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the International Logic Programming Conference and Symposium*, 1070–1080.

[5] Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation*, *1*(2), 93–124. https://doi.org/10.1080/19462160903564592

[6] Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, *77*(2), 321–357. https://doi.org/10.1016/0004-3702(94)00041-X

[7] McNamara, P., & Van De Putte, F. (2025). Deontic logic. In E. N. Zalta & U. Nodelman (Eds.), *The Stanford encyclopedia of philosophy*. Metaphysics Research Lab, Stanford University. Winter 2025.

[8] Alchourrón, C. E., & Bulygin, E. (1984). Permission and permissive norms. In W. Krawietz, H. Schelsky, G. Winkler, A. Schramm (Eds.), *Theorie der Normen*. Duncker & Humblot.

[9] Arriagada Cáceres, M. B. (2021). The closure of the systems of legal norms of competence. Episode I: The normative powers of public authorities. *Revus. Journal for Constitutional Theory and Philosophy of Law*, *43*. https://doi.org/10.4000/revus.7374

[10] Kletzer, C. (2021). Philosophy, law, and permission. *The American Journal of Jurisprudence*, *66*(2), 373–393. https://doi.org/10.1093/ajj/auab023

[11] Opałek, K., & Woleński, J. (1973). On weak and strong permissions. *Rechtstheorie*, *4*, 169–182.

[12] Hansson, S. O. (2013). The varieties of permission. In D. Gabbay, J. Horty, X. Parent, R. van der Meyden, & L. van der Torre (Eds.), *Handbook of deontic logic and normative reasoning* (pp. 195–240). College Publications.

[13] Governatori, G., Rotolo, A., & Sartor, G. (2021). Logic and the law: Philosophical foundations, deontics, and defeasible reasoning. In D. M. Gabbay, J. Horty, X. Parent, R. van der Meyden, & L. van der Torre (Eds.), *Handbook of deontic logic and normative reasoning* (pp. 655–760). College Publications.

[14] Olszewski, M., Parent, X., & Van der Torre, L. (2024). Permissive and regulative norms in deontic logic. *Journal of Logic and Computation*, *34*(4), 728–763. https://doi.org/10.1093/logcom/exad024

[15] Governatori, G., & Rotolo, A. (2023). Deontic ambiguities in legal reasoning. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law (ICAIL '23)*, 91–100. ACM Press. https://doi.org/10.1145/3594536.3595175

[16] Jones, A. J. I., & Sergot, M. (1992). Deontic logic in the representation of law: Towards a methodology. *Artificial Intelligence and Law*, *1*(1), 45–64. https://doi.org/10.1007/BF00118478

[17] Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P., & Cory, H. T. (1986). The British Nationality Act as a logic program. *Communications of the ACM*, *29*(5), 370–386. https://doi.org/10.1145/5689.5920

[18] Horty, J. F. (1993). Deontic logic as founded on nonmonotonic logic. *Annals of Mathematics and Artificial Intelligence*, *9*(1), 69–91. https://doi.org/10.1007/BF01531262

[19] Prakken, H., & Sartor, G. (2015). Law and logic: A review from an argumentation perspective. *Artificial Intelligence*, *227*, 214–245. https://doi.org/10.1016/j.artint.2015.06.005

[20] Fuller, L. L. (1964). *The morality of law*. Yale University Press. https://doi.org/10.2307/2217903

[21] Brownsword, R. (2016). Technological management and the rule of law. *Law, Innovation and Technology*, *8*(1), 100–140. https://doi.org/10.1080/17579961.2016.1161891

[22] Diver, L. (2021). Digisprudence: The design of legitimate code. *Law, Innovation and Technology*, *13*(2), 325–354. https://doi.org/10.1080/17579961.2021.1977217

[23] Yermukhametova, S. R., Mukaldyeva, G. B., & Balashov, T. T. (2026). From natural language to digital logic: Conceptualizing 'rules as code' within the framework of statutory

interpretation. *Statute Law Review*, *47*(2), hmag016. https://doi.org/10.1093/slr/hmag016

[24] Przymusinski, T. (1990). Well-founded semantics coincides with three-valued stable semantics. *Fundamenta Informaticae*, *13*(4), 445–463. https://doi.org/10.3233/FI-1990-13404

[25] Caminada, M., Sá, S., Alcântara, J., & Dvořák, W. (2015). On the equivalence between logic programming semantics and argumentation semantics. *International Journal of Approximate Reasoning*, *58*, 87–111. https://doi.org/10.1016/j.ijar.2014.12.004

[26] Chun, A., Wong, M. W., & Lauritsen, M. (2024). Domain-specific languages and legal applications. *RAIL: The Journal of Robotics, Artificial Intelligence & Law*, *7*(1), 19–42.

[27] Billi, M., Calegari, R., Contissa, G., Lagioia, F., Pisano, G., Sartor, G., & Sartor, G. (2021). Argumentation and defeasible reasoning in the law. *J*, *4*(4), 897–914. https://doi.org/10.3390/j4040061

[28] Flood, M. D., & Goodenough, O. R. (2022). Contract as automaton: Representing a simple financial agreement in computational form. *Artificial Intelligence and Law*, *30*(3), 391–416. https://doi.org/10.1007/s10506-021-09300-9

[29] Huttner, L., & Merigoux, D. (2022). CATALA: Moving towards the future of legal expert systems. *Artificial Intelligence and Law*, 1–24. https://doi.org/10.1007/s10506-022-09328-5

[30] Morris, J. (2023). Building Blawx. In *Proceedings of the International Conference on Logic Programming 2023 workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023)*. CEUR-WS.org

[31] Kowalski, R. A., Dávila, J. A., Sartor, G., & Calejo, M. (2023). Logical English for law and education. In D. S. Warren, V. Dahl, T. Eiter, V. M. Hermenegildo, R. A. Kowalski, & F. Rossi (Eds.), *Prolog: The next 50 years Lecture Notes in Computer Science* (pp. 287–299). Springer. https://doi.org/10.1007/978-3-031-35254-6_24

[32] Arias, J., Moreno-Rebato, M., Rodriguez-García, J. A., & Ossowski, S. (2024). Automated legal reasoning with discretion to act using s(LAW). *Artificial Intelligence and Law*, *32*(4), 1141–1164. https://doi.org/10.1007/s10506-023-09376-5

[33] Soldà, D., & Eiter, T. (2026). deon-B: A language for well-founded deontic planning. In G. Casini, B. Dundua, & T. Kutsia (Eds.), *Logics in artificial intelligence* (pp. 187–204). Springer Nature Switzerland. https://doi.org/10.1007/978-3-032-04587-4_12

[34] Cabalar, P., & Manteiga, O. (2025). deolingo: Extending answer set programming with deontic reasoning. In *Deontic logic and normative systems: 17th International Conference, DEON 2025*, 61–77.

[35] Arias, J., Carro, M., Salazar, E., Marple, K., & Gupta, G. (2018). Constraint answer set programming without grounding. *Theory and Practice of Logic Programming*, *18*(3–4), 337–354. https://doi.org/10.1017/S1471068418000285

[36] Governatori, G., & Rotolo, A. (2026). *Deontic argumentation*. arXiv. https://arxiv.org/abs/2509.25781

[37] Governatori, G. (2024). An ASP implementation of defeasible deontic logic. *Künstliche Intelligenz*, *38*, 79–88. https://doi.org/10.1007/s13218-024-00854-9

How to Cite: Governatori, G. (2026). Weak Permission Is Not Well-Founded, Grounded, and Stable. *Journal of Computational Law and Legal Technology*. https://doi.org/10.47852/bonviewJCLLT62029051