# On the Specification and Monitoring of Timed Normative Systems

Shaun Azzopardi[1(✉)], Gordon Pace[2], Fernando Schapachnik[3], and Gerardo Schneider[1]

[1] University of Gothenburg, Gothenburg, Sweden
shaun.azzopardi@gu.se, gersch@chalmers.se
[2] University of Malta, Msida, Malta
gordon.pace@um.edu.mt
[3] ICC and Departamento de Computación, FCEyN, Universidad de Buenos Aires, Buenos Aires, Argentina
fschapachnik@dc.uba.ar

**Abstract.** In this article we explore different issues and design choices that arise when considering how to fully embrace timed aspects in the formalisation of normative systems, e.g., by using deontic modalities, looking primarily through the lens of monitoring. We primarily focus on expressivity and computational aspects, discussing issues such as duration, superposition, conflicts, attempts, discharge, and complexity, while identifying semantic choices which arise and the challenges these pose for full monitoring of legal contracts.

**Keywords:** Deontic logic · Timed logic · Normative systems · Legal contracts · Monitoring

## 1 Introduction

If Alice is permitted to download a song from an online content provider, and gets a bonus that allows her to download another one, everybody would agree that now she can download up to two songs. Let's add time to the equation and consider Alice being permitted at 7am to download a song from 8am to 10am. At 9am she is granted another download permission, from 9am to 11am. At 9:30am she downloads a song. Can she download another at 10:30am? In other words, which of the two permissions did she exhaust? Is the permission involved a conditional one? How do we specify and monitor for these kind of timed normative specifications?

Reasoning about permissions and other normative modalities is the domain of *deontic logic*, while reasoning about time is usually the domain of *temporal logic* in the verification community. Verification and monitorability of temporal logics, including ones with real-time, has been extensively investigated (e.g., [9,32,33]).

Partially supported by UBACyT 20020130200032BA and PICT-2016 201-0112, the Swedish Research Council (*Vetenskapsrådet*) under grant Nr. 2019-04951 (*X-LEGAL: Smart Legal Contracts*), and the ERC Consolidator grant DSynMA (No. 772459).


L. Feng and D. Fisman (Eds.): RV 2021, LNCS 12974, pp. 81–99, 2021.
https://doi.org/10.1007/978-3-030-88494-9_5

We argue that normative concepts (e.g., *obligations, prohibitions, and permissions/rights*) are not appropriately modelled using existing monitoring logics. For example, specifying in LTL a prohibition to never download songs illegally is easy, however how can one specify that the specification may be violated but repaired by paying an appropriate fine? The naïve approach would simply use a disjunction between the two formulas, however this does not capture the difference in priority between the clauses and the different levels of violation.

Deontic logics have been proposed instead for normative reasoning. Different deontic languages have in fact been explored, with the ability to model and monitor for violations and their repair (e.g., [6,7]). Such languages involve certain normative modalities, parametrised by some state- or event-based formula, given some appropriate background theory. The difficulty and complexity of formalising untimed normative systems using deontic concepts have been studied in [43], while the justification on why LTL, CTL, and process algebra might not be sufficient to capture all the deontic notions has been presented in [18].

In this paper we want to focus on the monitorability of these logics under extensions with timed aspects. Although the different modalities that arise from adding time to deontic logic have been studied before, there are still many unanswered questions and no analysis of their monitorability. For instance, in [25] Governatori et al. analyse permissions with deadlines but do not discuss the issue of which one is discharged in case of temporal superposition nor what happens with timed permissions in the context of contracts. Superposition presents challenges for obligations too. Hashmi et al. [30] extend previous work by Governatori et al. [26] to deal with the temporal compliance of rules and present a categorisation of many types of obligations (*punctual*, *persistent*, *achievement*, etc.) based on the timing of their effect, enforcement and violation. However, some issues remain unexplored:[1] if $O[0,10](a)$ and $O[5,15](a)$ are two *achievement* obligations (meaning the obligation is discharged by the execution of a single action $a$ during the period) and $a$ is not executed in the $[0,5)$ time interval, does it need to be performed twice during the $[5,10]$ period?

Normative conflicts become more interesting and challenging in the presence of time. We would certainly consider $F(a)$ to be in conflict with $O(a)$ in an untimed and punctual context, but what happens with $F[0,10](a)$ and $O[5,15](a)$? Is this an unresolvable conflict despite the fact that compliance is possible? Should it be concluded that while there is a conflict in the $[5,10]$ interval the contract requiring both is still valid?

In many cases it is interesting to talk about the moment a given obligation (or right) is enacted, or whenever the action or state of affairs affected by such modalities occur, and refer to them in another clauses. For instance, consider the situation in which Alice has the right to download two songs, but the second one

[1] In this paper we will use the notation $O[b,e](a)$ to denote the obligation to perform action $a$ between time $b$ and $e$. We use the same notation but with $P$ to denote permission and $F$ to denote prohibition. Note that despite the formal syntax, we are not committed to a formal semantics, since the paper is dedicated to explore the family of such semantics one can choose to adopt.

may only be downloaded between 3 and 5 days after downloading the first. Later on, an obligation to write a review on the songs is enforced, with a deadline of 30 days after having downloaded the second song. These kinds of (relative) timing constraints for deontic norms are usually not treated in the literature, as there is a need of having richer timed logics with, for example, *freeze quantifiers* [1].

In this article we continue the exploration of the different challenges that a timed deontic logic presents. Compared to previous work, we discuss the issues of timed superposition, timed deontic conflicts, discharge of deontic modalities in case of conflicts and attempting. Integrating time as a first class citizen in a logic brings not only expressivity concerns but also complexity and computability issues. In the case of a deontic logic, many other subtle issues arise, which we identify and discuss. We also discuss monitorability of deontic logics and their timed extensions, and consider the challenges to monitor synthesis.

The next section discusses different timed logics used in computer science and the trade-offs their represent in terms of expressivity and complexity, and briefly introduces deontic logic. After that, Sect. 3 digs into the different interpretation challenges presented by the inclusion of time in deontic logic. Section 4 discusses the monitorability of deontic logics and their timed extensions, and suggestions for monitor synthesis of the timed case. References to related work are made throughout the article. Section 5 concludes the paper with some final observations.

## 2 Background

### 2.1 Temporal Logics, Timed Logics and Complexity

Computer scientists have studied different *temporal* and *timed logics*, including their expressiveness, properties and decision procedures for their satisfiability, monitorability, and validity.

Temporal aspects appear in many different ways in real life, be it in computer systems or the legal domain. The simplest are probably situations related to the frequency on the occurrence of certain events, and the order between events, be it sequences, ordering or causality. *Temporal logics* have been around for a while and have been successfully used for specifying reactive and other computational systems, and also in combination with deontic logic (e.g., [24–26,29]). In practice, as much of this work has observed before, any non-trivial normative document contains temporal aspects.

Temporal logics allow for reasoning about the ordering and causality between events, and come in different flavours, depending on whether time is discrete or continuous, whether there is a single future (linear) or it captures different possible futures (branching), whether it is possible to talk about the past or only the future, whether the logic talks about points in time or intervals, whether there is a global notion of time or only relative time, etc. Timed logics may be propositional or have quantifiers. Expressiveness and decidability are of course very much dependent on a combination of the different choices made on all the dimensions mentioned above (and others).

There is extensive work on the use of different timed logics, and we will not give details here as this is beyond the scope of this paper. That said, we will briefly describe the expressive power of three timed extensions of temporal logics namely time-bounded operators, freeze quantification, and time variables as presented by Alur and Henzinger [1], due to their relevance in what follows.

The first, and possibly least expressive, way to add timing constraints to existing temporal logics is to have *bounded temporal operators* where the classic temporal operators are enhanced with (integer) intervals. In this logic you can express properties like *"every event $e_1$ is followed by another event $e_2$ within 7 time units"*, written as $\Box(e_1 \rightarrow \Diamond_{[0,7]} e_2)$. The bounded-operator notation can only express properties relating adjacent temporal operators and cannot express non-local properties of the kind *"every event $e_1$ is followed by a response $e_2$, which is followed by another response $e_3$, such that the delay between $e_2$ and $e_3$ is no more than the delay between $e_1$ and $e_2$"*.

For that there is a need of a more expressive logic, one containing *freeze quantifiers*. This second variant of timed logic allows for quantification over time variables, which may be used to compare with other time variables. The non-local property given above would be expressed as follows in such a logic: $\Box\, x.(e_1 \rightarrow \Diamond\, y.(e_2 \wedge \Diamond z.(e_3 \wedge z - y \leq y - x)))$, where $x$ and $y$ are time variables associated with the corresponding states defined by the formula in its scope ($x.\varphi(x)$ holds at time $t$ iff $\varphi(t)$ does).

A third, and more expressive, way to write timing constraints is by using *explicit clock variables*, based on first-order temporal logic and explicit (global) time, thus allowing to existentially and universally quantifier over clocks (see [35] for examples).

Expressivity, however, comes at the price of complexity or even undecidability. This is specially pressing when some type of tool is used or envisioned. For instance, encodings using timed automata may require one clock per variable (e.g., [16]), but the verification complexity depends on the number of clocks—and is exponential in the case of Timed Computational Tree Logic (TCTL).

Extending temporal logics with time-bounded temporal operators increases their complexity, in some cases yielding undecidability. For instance, the model checking problem for Metric Temporal Logic (MTL) is undecidable. Nevertheless, some interesting fragments of MTL, such as Metric Interval Temporal Logic (MITL) are decidable but EXPSPACE-complete [10].

For more examples and more details about expressiveness of these logics and other variants we direct the reader to [1] and the references therein.

### 2.2 Formalisation of Normative Systems: Deontic Logics

Deontic logic is the study of deontic modalities—mainly obligation, prohibition, and permission, meant for the modelling of legal and moral notions [22]. Initial deontic formalisms in the philosophical field (e.g., Standard Deontic Logic [23,39]) faced certain paradoxes, e.g., the inability to express what should happen to make up for an obligation being violated without introducing contradictions.

These paradoxes have remained problematic and the subject of debate [28]. However different approaches have been recently proposed to solve these problems (e.g., [16,44,45]), that have made deontic logic more useful in practice. We introduce the main concepts based on existing deontic formalisms briefly, but to keep our discussion general—we do not commit to any particular formalism for this paper.

A distinguishing feature between deontic formalisms is whether they are event/action- or propositional/state-based. One may be obliged to perform a certain action (e.g., pay a certain fee), or to reach a certain state (e.g., the state of having no pending payments to make). These approaches are dual, reaching different states typically requires performing actions (or, in a timed context, the passing of time), while actions may cause changes in state. For simplicity and without loss of generality we continue the discussion in this section by referring solely to action-based logics.

An obligation to do some action $a$ requires $a$ to be performed. This is often represented as $O(a)$. Similarly for permissions, $P(a)$, and prohibitions $F(a)$. These three can often be defined in terms of each other, e.g., a prohibition not to do an action is the obligation to not do it ($O(a) = F(\neg a)$), while a permission to do an action is often the lack of an obligation to do not do the action ($P(a) = \neg O(\neg a)$). Here we go beyond this and focus on the strand of work that views permission as a right to perform an action, as suggested in von Wright's seminal work [22], where permission includes an implicit obligation for the other parties to allow the permitted party to perform the action.

Variants of these modalities can also refer to a party, i.e. $O_p(a)$ where $p$ is the name of the obliged party (e.g., [5]). Without these notational variants there may be underlying assumptions about which party or parties are associated with each action.

An important aspect of deontic formalisms is how they handle *contrary-to-duty* norms, or *reparations*. These are clauses that come into effect when there has been a violation, allowing some action/s to repair the violation, e.g., the paying of a penalty. The ability to handle these clauses is of utmost importance for practical applications of normative systems, e.g., for the monitoring of contracts or laws that use these kind of clauses routinely. We represent this, e.g., for obligations with $O_{O(\$10);O(b)}(a)$ where a 10 dollar penalty must be paid if $a$ is not performed, and $b$ performed after.

For more extensive and in depth material about deontic logic in general see [20] and references therein.

The combination of deontic concepts with (real-)time has been considered in an *ad hoc* manner in the literature, with different interpretations chosen without justification or contrasted with other possible ones. The effects of adding time to specific deontic operators was discussed in [25,26]. Note that those, and other similar work by Governatori et al., do not address the general more complex issue of getting a fully-fledged logic (or formal language) where many timed operators co-exist. *C-O Diagrams* [13,16,38] is a formal (visual) language (not a logic) featuring deontic concepts and timed constraints, with a timed automata

semantics. The language has interesting features but does not address many of the issues discussed in this paper. No monitoring techniques has been studied for any of the above languages and logics.

## 3 Interpreting Timed Norms

In this section we resort to small examples to discuss different issues and design choices, that need to be taken into account when thinking about a deontic logic that is able to fully embrace all aspects of time-related expressions.[2]

***State- vs. Action-Based Deontic Operators.*** The duality between contracts regulating events vs. regulating the state-of-affairs is also reflected in the deontic modalities themselves when taking into consideration time. On one hand, one may have pointwise modalities—for example, the obligation to perform an action at a particular point in time. Such pointwise modalities are frequently encountered when considering a system with discrete time events. For example, if the service-provider gives priority to a particular user at time point $t_i$, they are obliged to give priority to another user at the next time point (when an event is received) $t_{i+1}$. However, when one considers continuous real-time clocks, deontic modalities are typically over intervals of time. For example, an obligation with a deadline might oblige the service provider to ensure that a service is continuously available over the coming hour; or a user accessing a digital asset management system, may be prohibited from requesting the download of a file twice within a second of each other.

If we consider interval-based deontic modalities, there lies a duality with the event- vs. state-based view of the world. Does one identify the points in time when a modality starts holding and when it terminates, or does one identify the interval over which the modality holds? The most common approach one finds in the literature is the state-based approach (e.g., [26,27,29,30]), following the approach used in interval temporal logics such as Interval Temporal Logic (ITL) [41] and duration calculus [14]. This approach correlates closely with natural language clauses expressing concepts such as deadlines: *"The user is prohibited from transferring funds to a third party in the first 7 days of the creation of an account"* or *"The bank is obliged to refund a user within 15 days of a request to redeem an account."*

However, there is also work which takes the action-based approach (e.g., [17]), in which the key is to signal the start and end of a modality e.g., $\overleftarrow{O}_p(A)$ indicates the beginning of a time interval over which there is an obligation on party $p$ to perform action $A$, while $\overrightarrow{O}_p(A)$ would be the end of this obligation. Such an approach corresponds to when these moments are identified in a legal text in separate ways, for instance *"The student has the right to upload a new assignment*

[2] It is worth noting that different interpretations of normative statements go far beyond the assignment of a formal semantics. Such differing views frequently correspond to views different parties may have of a normative text, e.g., a contract, including possibly in court.

*from the first day of term"* and *"If a student unregisters from a unit, he or she automatically loses the right to upload assignments."*.

The latter approach lends itself to a trace-based semantics, in which each event or time progression updates the clauses in force. However, this inherent state of active clauses makes compositional reasoning over contracts more difficult, and the former approach typically yields cleaner semantics.

***A Plethora of Timed Deontic Modalities.*** What time should the logic refer to? Absolute time, i.e. a universal and always accessible clock that is referred to in every time-related expression, might be relatively inexpensive from a computational point of view, yet equally unrealistic from a legal perspective. Many legal expressions also require relative time, as in *"warranty period should be at least 3 month from the time of purchase"*. Is this just syntactic sugar to an expression like *"let $U$ be the universal clock, let $p$ be the time of purchase (according to the universal clock), and the warranty period is of $w$, then the purchased item is still in warranty as long as $U \leq p + w$"*? Complexity usually grows with the number of clocks, so it is in general desirable to reduce the number of clocks. This might be possible in some cases, for instance whenever what is needed is only the time-stamp associated with a given event but not how time evolves for such event (e.g., to compute duration). In some other cases, clock reusability is possible, although this kind of optimisation is usually handled under the hood by the tools.

Some use cases do require more intricate expressions of time. Consider for instance *"license can be renewed during 10 days after expiration if the expiration cause was A, or 15 days if the expiration cause was B"*, an expression where the deadline is relative to occurrences of events in the past, or *"if the item under warranty is taken to reparations, the warranty period is extended by the amount of time the item is being repaired, each time it is repaired"*, where the deadline needs to be computed. Such expressions seem to call for an algebra of time intervals, another threat to computability when real-time is involved [31].

To complicate things further, deadlines can sometimes be expressed in relation to an event still to happen (e.g., *"service should be provided until the user disconnects"*), or as a boolean expression involving many time references (e.g., *"service should be provided until the user disconnects, with a maximum of one month of service, not surpassing the calendar year"*).

Once deontic clauses have an explicit duration, the issue of possible multiple violations during that period arises. Should multiple violations trigger multiple reparations? Also, one should be able to distinguish between multiple violations vs. one violation that has a duration. Think of trespassing: if entering a facility is forbidden during the night, is trespassing twice for 1 h each the same as trespassing once for 2 h? As timed logics allow to measure the duration of an event, duration of violation should also be available as a parameter to the reparation clause. For example, an obligation to provide food and water to passengers during a flight might be redressed with a fine, possibly proportional to the duration of the flight.

Obligations with duration present challenges of their own: how should be $O[0,10](a)$ read? If $a$ is an event or action, should it be sustained during the $[0,10]$ period, or it is only mandatory to do it at least once during the period? Again both cases are reasonable and may be required in different contexts, with the modalities allowing for the expression of the two. Even limiting our view to the variant to oblige the performance of the action once during the period, how should $O[0,10](a) \oplus O[5,15](a)$ be interpreted? Does the performance of $a$ during the interval $[5,10]$ satisfy both obligations, or are two occurrences required? Although the latter may appear to be more reasonable, it is worth noting that this would mean that conjunction is no longer idempotent, with $O[0,10](a) \oplus O[0,10](a)$ being different from $O[0,10](a)$. Actions differ on the nature of their effects. Ensuring the door is open is idempotent, but paying or buying are not.

Part of the issue at stake in the previous example is whether modalities are *dischargeable* or *permanent*. Dischargeable modalities cease their effect once they are fulfilled, while permanent ones do not. For instance, prohibitions tend to be permanent. That is, a prohibition is still enacted (and in force) even when somebody has violated it. Furthermore, a prohibition is still in force even if a violation triggers its corresponding reparation. Note that a prohibition (and actually every modality) can be at the same time permanent and time bounded (e.g., it is forbidden to enter the pool during the night).

Dischargeable obligations are also common. Consider having ten days to fill in a report. The obligation is discharged with the execution of one instance of the action (the filling of the report), and gets violated when the deadline is met without the action happening. If the obligation were permanent (e.g., behave nicely during school time, or do what your boss asks during working hours), when the deadline is met there is no violation. In this case, violations occur within the interval when the obligation is active.

Governatori et al. [25] have characterised different types of obligations with deadlines, for instance distinguishing between *achievement* obligations (corresponding to the obligation to perform an action before the deadline), and *maintenance* obligations (corresponding to the obligation to ensure that a state holds until the deadline elapses). The variety of types of obligations with deadlines the authors present encompasses many common types of obligations even if not necessarily complete (in that not all forms of obligations over intervals are covered) already indicates the variety of choices one can adopt from when designing a real-time deontic logic with connections between our discussion above and the formalised notions in [25].

***Permission and Time.*** In the case of an action-based logic, being permitted to do something within an interval can mean several different things, from having a continuous permission to repeat the action as many times as one wants (e.g., permission to enter the facility during daytime) to a one time permission, i.e. a dischargeable one.

In that last case it also seems to make sense to have some type of algebra of dischargeable permissions, as being granted a one day permission to download a song is not the same being granted the same permission twice. While it seems

clear that meeting the deadline puts an end to every instance of the permission, what happens with the combination[3] of permissions $P[0,10](a) \oplus P[5,15](a)$ if at time 6 the action $a$ is performed? When time reaches 11, is there still one permission left over? Is it always the case that the discharged permission is the oldest one? If so, why?

Another conflicting case can arise if we consider a one-time permission to use a service for half an hour. Now suppose another similar permission is granted. Can the bearer of the two half-hour permissions use the service for a full hour? In some cases there might be no difference while in others the granter of the permission may not consider them to be equivalent because a gap in between may be required (e.g., riding a horse or using a machine which might overheat).

Now consider the same example in the context of contracts, where one party can violate the other's permission to execute an action by not providing the proper synchronisation. Suppose party $p$ has two permissions to execute synchronised action $a$: $P_{O(\$10)}[0,10](a) \oplus P_{O(\$20)}[5,15](a)$, the first having a reparation fee of \$10 and the other a fee of \$20. If at time 6 party $p$ is not able to execute the permitted action because of lack of synchronisation by the other party, what is the fee applicable as reparation? Is it \$10, \$20 or even \$30?

Permissions with intervals, both permanent and dischargeable, present challenges to clearly define time-based conflicts. As an example, think of a permission to present a form in a government office until midnight on a specific date, yet the office is available only in working hours. Common sense states that the permission is for presenting during working hours until the midnight of the given date, and that there is no conflict involved. However, finding a formulation where this can be expressed naturally in a formal language is challenging, because of the chain of logical relations that need to be established to link the action of *'handing in the form'* with the *'office being open'* predicates. Furthermore, in some cases, limitations on time windows are made with the specific goal of discouraging the performance of the action (in this case the presentation of the form). If the intersection of the deadline and the working hours only left a small time frame available, should this situation be detected? This may indicate that the notion of conflict may, in some cases, be a fuzzy rather than crisp predicate.

***Attempted Actions.*** As was mentioned before, in action-based timed deontic logics hitting a deadline without fulfilling an obligation is considered to be a violation. Think of the case of a contract, in which party $A$ agrees to sell to party $B$ at a discount price because party $B$ agrees to buy at least 200 kg of goods during a one week period. During the first few days a transaction is made and $B$ buys 100 kg at a discounted price. Then the week goes by with no other transaction being completed. The case goes to court, where $A$ is claiming that $B$ took advantage of the discount price without reaching the minimum agreed volume. Party $B$ argues that she did attempt to buy several times yet on all the occasions party $A$'s shop was either closed or out-of-stock. $B$ even mentions one

[3] From this point onwards, we will use the notation $C \oplus C'$ to denote both $C$ and $C'$ being enacted. We avoid the use of symbols typically used for conjunction e.g., $\wedge$ or & in order to avoid implicit assumptions of idempotency of the operator.

occasion where she emailed $A$ to arrange for a purchase and $A$ took so long to respond that she had to get the goods from another supplier.

How should a logic handle such a case so that the attempts became observable? If buying and selling are separate actions, then that would mean parties can execute them independently, which is not the way a buy-sell agreement should be modelled. Effectively, it makes more sense to think of a synchronised buy-sell action that both parties need to agree to execute. The problem is that in most action-based logics when one party tries to synchronise on a shared action and the other party does not handshake, there is no trace of attempt in the resulting execution. Other logics, specifically those where events are timestamped, do leave a trace and the resulting execution has two events, close enough in time, and probably a third acknowledgement message, all of which can be abstracted together as a single transaction, or a high-level synchronised event.

Deontic logics usually do not allow for such a two-level interpretation: one where individual events can be seen ($B$ trying to buy without being responded by $A$) and another where a successful sequence of a buy attempt and a proper response are abstracted as a single buy-sell transaction. This is common in network protocols where a 'connection' is a high-level event with an initiator and a completer. Being an initiator is a role: any of the parties can be the initiator just by sending first the proper connection initiation message. A deontic logic should probably allow for a more complicated scenario: it should let any of the parties attempt the transaction, but without a single initiation message, i.e. a buy-sell transaction can be started either with a buy or a sell attempt.

Going back to the discount-per-volume case, $B$'s obligation of buying should be regarded as discharged because either a high-level buy-sell action took place, or a low level buy attempt was issued by $B$ without a response from $A$ within a proper time-frame, which may not be formal defined. Actually, whether two buy and sell messages separated by $t$ time units are to be considered to correspond to an acknowledged request or not may be a controversial issue among the involved parties. What tools should the logic provide to ground this discussion?

***Timed Conflicts.*** Conflicts due to time expressions are also a topic of interest. Although $F(a) \oplus O(a)$ is clearly a conflicting sentence, how should we interpret $F[0, 10](a) \oplus O[5, 15](a)$? Is this an unsatisfiable conflict? Should it be concluded that while there is a conflict during the interval $[5, 10]$—at the beginning of which $F[0, 5](a) \oplus O[0, 10](a)$ is in force?

If action $a$ takes time to perform, does the interval specify when the action should commence, finish or all the performance time? If, for instance, we take the time of commencement, $O[5, 10](a)$ means that the action must start in the interval irrespectively of when it ends (as in *'the shipment should be sent to their the destination during the next 24 h'*), but should the time instant be the same for a prohibition as in $F[5, 10](a)$? Is there a violation if action $a$ starts at time 3 and finishes at time 6?

We have already discussed the issue of having overlapping time intervals for obligations, permissions and prohibitions. The situation is of course even more complex in the presence of CTDs (contrary-to-duties) and CTPs (contrary-to-

prohibitions) clauses. Timed CTDs and CTPs may be problematic also if their triggering is conditional to some relative notions of time, and of course in case of normal delays not necessarily due to the fault of the involved parties.

For instance, a company working regulation might state that all employees must answer company email within 24 h of receipt. If they will not be able to answer within this time-frame, they should then send a standard mail at least one hour before the 24-h deadline saying that they will not be able to answer in time and state by when an answer is to be expected. In the absence of both an answer and the canned response, the company automatically sends a message with a reprimand to the employee (after the 24 h deadline) and decreases the employee's bonus by 2 points. A concrete situation might be that Alice sends her answer exactly 23 h after having received an email but a system problem causes her answer to arrive after the 24-h deadline. The system then will produce the automatic response and will decrease her bonus balance by 2 points. A solution based on time-stamps might help here: every event should have a time-stamp and all the norms should be explicit on whether it refers to the time-stamp of attempting, sending or receiving something. This solution, however, might cause inconsistencies as certain obligations will be triggered and might need then to be recalled (similar to rollbacks in long-lived transactions). What is then the meaning of recalling such obligations? Of course we should also recall all the corresponding (eventually nested) CTDs (and similarly for CTPs).

In the above example one the main issues was caused by delays. Should we allow for reasoning only for the ideal case, or should we include a model of the delays? Which delays are acceptable and which are not (from the liability point of view)?

***Other Standing Challenges.*** Deontic formal languages can serve many purposes such as conflict analysis, runtime verification, simulation, etc. Each of these domains of application impose its own constraints. For instance, matching real occurrence time of the events is an issue in run-time monitoring, specially for distributed events. Thus, coping with rollback-able attribution of guilt for failed deadlines (like the example given in Sect. 3) or fuzzy-matching of events [11] (i.e. being able to deal with the fuzziness of timestamps of real-life events) might become a requirement that is not really necessary for other types of applications.

Although there are purely logic-based approaches successfully dealing with time (e.g., [30,36]), most of the existing tools are automata-based (e.g., [37]), thus, if one wants to warrant tool support, one might want to use some kind of underlying timed transition system with annotations on the deontic imperatives.

From the design point of view, the choice might be between starting with timed automata, one of the most popular automata-based timed formalisms, and add the deontic information, or start with the standard deontic logic Kripke semantics and add time to it. This choice might be driven by different considerations and we do not have a formal argument in favour or against any of them.

A good example of how deontic modalities and timed constraints may be combined, somehow following the first approach mentioned above (interpret-

ing and encoding the deontic modalities into timed automata), is the case of C-O Diagrams [16], for which a timed automata semantics was given (see [16] for a first translation and [13] for a new optimised translation for an extension of the original diagrams). The translation was implemented as Uppaal automata and integrated into a toolchain called *Contract Verifier* [12].

## 4 Monitoring Norms and Timed Norms

In the previous section we introduced and reviewed different interpretations of deontic modalities that arise in a real-time context. In this section, we continue the discussion with a focus on monitoring of normative systems under these different modalities and interpretations.

### 4.1 Monitorability

The appropriateness of a logic for runtime verification depends, amongst other things, on its *monitorability*, that is whether for any finite execution we can eventually make a determination whether a specification is satisfied or violated [9,46]. For example, monitoring for linear temporal logic (LTL) has certain limitations, e.g., the LTL specification $\mathbf{F}a$ can only be monitored for satisfaction (if $a$ occurs), but not for violation (without some knowledge of the underlying system). We discuss these standard notions of satisfaction- and violation-monitorability with regards to deontic logics. Although one finds literature on the monitoring of norms in specific logics (e.g., [2,21]), rather than focus on a particular logic in this section we take a more high level view.

Consider an 'obligation' to eventually do a positive action, without any time limit. Is this truly an obligation? Such an 'obligation' can essentially be postponed forever, and thus we cannot monitor for its violation. If we allow it, we can however monitor for its satisfaction (similarly to $\mathbf{F}a$ in LTL). On the other hand, the obligation not to do an action (or prohibition) without any time limit does have more meaningful normative semantics over finite traces—it is violated if the prohibited action is done.

Similar to prohibitions, permissions (here the *right* to do something) do not need to be bounded to make sense—the notion of perpetual rights is standard. However, they differ to obligations and prohibitions with regards to satisfaction and violation semantics. Permissions cannot be violated by the permitted party, but instead they can be violated by others when the permitted action requires the other party or parties to synchronise in their performance [8]. One interpretation is that the parties always, at each time step, provide the required synchronising actions, or at least at the time steps the permitted party wants to exercise the permission [3]. Essentially this is a safety property when we can monitor attempts to perform actions: just monitor for the attempt to exercise the permission and if it fails then the permission has been violated [7]. This interpretation can be relaxed to take into account that there may be real-world limitations on the performance of the action, and only enforce the obligation on the other parties

at time steps where it is possible for them to provide the synchronising actions, or within a bounded time-frame, without any effect on monitorability. It is worth noting that there are different types of rights identified in the literature [34], and the interpretation of permission as the liberty of one party from other parties interfering with that first party's performance of the permitted action is but one of them.

Another pertinent issue is that when monitoring deontic specifications we are not just interested in trace violation. In deontic logic there are multiple parties to a contract, and thus we are more often also interested in which party or parties caused the violation. This kind of blame assignment may not always be possible, for example when the specification is unsatisfiable, or is difficult when the actions of a party in the past may force another party to violate the contract in the future. Another aspect is that a party may still be in compliance with a contract if they reasonably attempt to satisfy it but are prevented to do so by the environment (the real world and the other parties). For example, one may not be able to satisfy an obligation because another has not provided the synchronising action, and thus the other is at fault. Capturing and analysing this also requires the monitoring of attempts to perform an action (e.g., [7]), otherwise this kind of compliance cannot be monitored for.

We have considered the monitorability of the different kinds of atomic norms, however norms can also be composed together in different ways. Allowing unconstrained logical combinations causes certain paradoxes and dilemmas (see [28,43]), since normative modalities are not truth statements. However, deontic logics with constrained interpretations of these combinations that avoid these paradoxes also exist (e.g., [40]). Here we consider the monitorability of combinations of normative modalities with unconstrained logical operators for completeness.

Sequence and conjunction clearly both maintain monitorability given monitorability of the sub-formulas (the resulting property remains co-/safety). A clause can also have an associated reparation clause, which can be modelled using a monitor for the first clause that upon detecting a violation of the first clause triggers a monitor for the second.

Norms can also be guarded or conditioned on something happening. The monitorability of the guards depends on the allowed expressions. If the expression is a regular expression or a past-time LTL formula then monitorability is maintained. It seems unlikely that allowing unmonitorable expressions would add anything to the logic, e.g., allowing future-time LTL to guard norms would allow us to write $[\mathbf{FG}p]O(a)$ (if $p$ is true infinitely often from this point on, then you are obliged to do $a$), which seems counter-intuitive—the party cannot reasonably be held to have violated a contract if the contract expects impossible things of them, such as clairvoyance.

We also consider the remaining usual logical operations: negation, and disjunction. Usually, a negated prohibition becomes a permission, a negated obligation becomes the permission not to do the action, and a negated permission becomes a prohibition. In deontic logics the disjunction can usually be moved

to the event/state parameter side given appropriate background theories (e.g., $O(a) \vee O(b) = O(a \vee b)$), or involves clauses with mutually exclusive guards (e.g., $[p]O(a) \vee [!p]O(a) = [p]O(a) \oplus [!p]O(a)$). The former is more difficult in the timed case, e.g., $O_{[0,5]}(a) \vee O_{[4,10]}(b)$, but can be solved in the same way by moving timing to the event side, i.e., $O(a_{[0,5]} \vee b_{[4,10]})$. In the case of disjunction of more complex clauses, e.g., with sequence $O(a); P(b) \vee O(b); P(a)$, guards can be used to remove the disjunction, i.e., $O(a \vee b); ([a]P(b) \oplus [b]P(a))$ (assuming only one action can happen in each time step).

Other deontic logics use the notion of *defeasability*, where certain clauses may be in conflict with each other but have a priority function that resolves the conflict (e.g., if the first rule does not hold then try the second) [27]. This has a disjunctive nature that does not affect monitorability.

Then full monitorability here requires the ability to observe failed actions, and knowledge about the synchronising actions made available by the parties. Without these we are unable to talk about whether parties have fulfilled their obligations with respect to a deontic contract.

### 4.2 Monitor Synthesis

One approach to monitoring deontic logics could involve their translation into established runtime verification logics, however there are some features of deontic logics that do not translate well. For example, the notion of *reparations*, where a party may be obligated to perform a certain action, and failing that they are in violation of the contract, but may perform certain actions as reparations for this violation and return into compliance. The best attempt at writing $O_{O(b)}(a)$ in LTL would involve disjunction, i.e., $a \vee (\neg a \wedge \mathbf{X}(b))$. However in LTL this loses the priority implicit in the deontic logic representation. Reparations are not simply other options, but imply recognition by the performing party that they have violated a contract, an action which can have legal effect. Moreover, violation of certain clauses does not mean other obligations are not still in effect—there are different levels of violation that are not captured accurately by existing approaches to monitoring. They could perhaps be added through certain meta-level considerations, but not at the level of existing monitoring languages. Working at the level of deontic logic instead allows us to directly take into account all these considerations that are required for legal contract monitoring.

In previous work we have given operational semantics to different untimed deontic languages (e.g., [7,19]) which can easily be used for monitor synthesis. An automata construction could also be constructed, through a Kripke structure where states are associated with the sets of norm clauses that must hold when at that state. Contract automata [6,8] may be able to be re-purposed for this.

The timed case has different needs, as discussed previously. Effective and efficient monitors for relative timing constraints are especially important in the monitoring of normative systems. These often specify norms that activate at the point another norm is satisfied, or penalties that start holding at the time another norm is violated.

Looking to LTL with (real-)time as inspiration we find different approaches for monitor synthesis. Focusing on *metric temporal logic* (MTL), i.e. temporal logic with until and since modalities holding over a certain interval, we find translations to deterministic timed automata that can be re-purposed for monitoring (e.g., [42]). Another interesting approach involves reducing the problem to monitoring LTL with atoms corresponding to bounded (i.e., with bounded intervals) MTL formulas [32]. Essentially the proposed algorithm uses dynamic programming techniques to determine the value of the bounded MTL formulas by collecting events appropriately depending on the associated time they occur and the interval associated with the formula. The authors extend this work for MTL with predicates that can refer to time points, allowing for monitoring of specifications with relative timing [33]. This suggests that separately combining monitoring of timing aspects and higher-level normative aspects may also be effective.

One issue not considered in detail in previous work is that of the underlying theory. One event/state parameter may correspond to the evaluation of a more complex predicate. For example, in a state-based deontic logic we may want to specify that at the end of each month there is an obligation that the average number of transactions is below a certain number. Operationalising such specifications can involve having a layer of monitors that compute these predicates' values, which can be queried by the norm monitors. For a rich monitorable language, forms of symbolic monitor automata (e.g., DATES [15], or [4]) can be used to compute the values of these predicates.

The proposed solution for complex event predicates above may be combined with an approach inspired by that of [32] for MTL—the bounds associated with an obligation may be moved to the events: $O[x, y](a) = O(a_{[x,y]})$. Thus we may be able to re-use monitor synthesis for the untimed case by simply adding a layer that transforms timed events into appropriate timed action atoms (e.g., if $x \leq z \leq y$ then the transformation $(a, z \rightarrow a_{[x,y]}$ can be applied, where the event $(a, z)$ denotes action $a$ occurring at time $z$), which are then processed by appropriate monitors for the deontic-level specifications.

## 5 Conclusions

In this article we explored the different issues and design choices that arise when considering how to fully embraced timed aspects into a deontic logic, mainly from a computational point of view. To do so we ask questions beyond those addressed in prior work by others (e.g., [11,16,17,25,26,30]). We resorted to small examples to discuss issues such as duration, superposition, conflicts, attempts, discharge, complexity and tool support among others, many of which were not covered in the literature.

In summary, we considered the state- and action-based approaches for interval-based deontic modalities, which respectively require the identification of the interval on which the modality holds, and the actions that identify the start and end of the interval. We discussed choices with regards to underlying

clocks (universal or relative), and different constraints required out of modalities (that something must hold until a deadline, or within a certain interval). The latter allows for different interpretations—given two intersecting obligations, doing one event may be able to satisfy both, or not. The issue of dischargeability and permanence of norms was also discussed (a norm may no longer hold after being first satisfied, or it may continue holding). Moreover, taking into account that attempts to fulfil a norm may fail, through no fault of the actor, in a timed context requires reasoning about synchronising actions not necessarily occurring at the same time step. Finally, we discussed conflicts due to overlapping time intervals, where a contract may have satisfying traces but other traces that exercise the conflict. This can be a problem especially in the context of reparations with some relative deadline.

One thing that many of the used examples have in common is that the different interpretations proposed seem to correspond with interpretations that different stakeholders might sustain in case of conflict, even in court. One research direction is to investigate a logic using nondeterminism to correspond to possible interpretations. What such a logic would provide is not settling over an interpretation but rather coherence: one branch might flag violations for actions that commence during the prohibited interval, irrespectively of where they end, while another one might only flag violations for prohibited action that happen entirely in the interval, but no branch would mix both interpretations. Thus, legal arguments become arguments about choosing (or pruning) branches in the logic.

Finally, we have discussed and analysed what are the main issues and challenges in the monitoring of (un)timed deontic logics. To the best of our knowledge no work exists on the monitoring of such logics. This is an open research direction for researchers in the RV community to consider. Though a successful approach might first need to address all the issues discussed in this paper concerning the extension of deontic modalities with time, we believe that existing approaches for MTL can inspire monitor synthesis techniques for timed deontic logics.

## References

1. Alur, R., Henzinger, T.A.: Logics and models of real time: a survey. In: de Bakker, J.W., Huizing, C., de Roever, W.P., Rozenberg, G. (eds.) REX 1991. LNCS, vol. 600, pp. 74–106. Springer, Heidelberg (1992). https://doi.org/10.1007/BFb0031988
2. Alvarez-Napagao, S., Aldewereld, H., Vázquez-Salceda, J., Dignum, F.: Normative monitoring: semantics and implementation. In: De Vos, M., Fornara, N., Pitt, J.V., Vouros, G. (eds.) COIN -2010. LNCS (LNAI), vol. 6541, pp. 321–336. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21268-0_18
3. Azzopardi, S.: Extending contract automata with reparation, hypothetical and conditional clauses. Technical report, University of Malta, May 2014
4. Azzopardi, S., Colombo, C., Ebejer, J.-P., Mallia, E., Pace, G.J.: Runtime verification using VALOUR. In: RV-CuBES 2017. Kalpa Publications in Computing, vol. 3, pp. 10–18 (2017)
5. Azzopardi, S., Gatt, A., Pace, G.J.: Reasoning about partial contracts. In: JURIX 2016, pp. 23–32 (2016)

6. Azzopardi, S., Pace, G.J., Schapachnik, F.: Contract automata with reparations. In: JURIX 2014, pp. 49–54 (2014)
7. Azzopardi, S., Pace, G.J., Schapachnik, F.: On observing contracts: deontic contracts meet smart contracts. In: JURIX 2018, pp. 21–30 (2018)
8. Azzopardi, S., Pace, G.J., Schapachnik, F., Schneider, G.: Contract automata - an operational view of contracts between interactive parties. Artif. Intell. Law **24**(3), 203–243 (2016)
9. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**(4) (2011)
10. Bouyer, P., Laroussinie, F.: Model checking timed automata. In: Modeling and Verification of Real-Time Systems: Formalisms and Software Tools, pp. 111–140 (2010)
11. Cambronero, M., Llana, L., Pace, G.J.: Timed contract compliance under event timing uncertainty (2019, submitted for publication)
12. Camilleri, J.J., Haghshenas, M.R., Schneider, G.: A web-based tool for analysing normative documents in English. In: SAC-SVT 2018, pp. 1865–1872. ACM (2018)
13. Camilleri, J.J., Schneider, G.: Modelling and analysis of normative documents. Logical Algebraic Methods Program. **91**, 33–59 (2017)
14. Chaochen, Z., Hoare, C.A.R., Ravn, A.P.: A calculus of durations. Inf. Process. Lett. **40**(5), 269–276 (1991)
15. Colombo, C., Pace, G.J., Schneider, G.: Dynamic event-based runtime monitoring of real-time and contextual properties. In: Cofer, D., Fantechi, A. (eds.) FMICS 2008. LNCS, vol. 5596, pp. 135–149. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03240-0_13
16. Díaz, G., Cambronero, M.-E., Martínez, E., Schneider, G.: Specification and verification of normative texts using C-O Diagrams. Trans. Softw. Eng. **40**(8), 795–817 (2014)
17. Farrell, A.D.H., Sergot, M.J., Sallé, M., Bartolini, C.: Using the event calculus for tracking the normative state of contracts. Int. J. Cooperative Inf. Syst. **14**(2–3), 99–129 (2005)
18. Fenech, S., Okika, J., Pace, G.J., Ravn, A.P., Schneider, G.: On the specification of full contracts. In: FESCA 2009. ENTCS, vol. 253(1), pp. 39–55 (2009)
19. Fenech, S., Pace, G.J., Schneider, G.: Automatic conflict detection on contracts. In: Leucker, M., Morgan, C. (eds.) ICTAC 2009. LNCS, vol. 5684, pp. 200–214. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03466-4_13
20. Gabbay, D., van der Meyden, R., Horty, J., Parent, X., van der Torre, L.: The Handbook of Deontic Logic. College Publications (2013)
21. Aranda García, A., Cambronero, M.-E., Colombo, C., Llana, L., Pace, G.J.: Runtime verification of contracts with Themulus. In: de Boer, F., Cerone, A. (eds.) SEFM 2020. LNCS, vol. 12310, pp. 231–246. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58768-0_13
22. Wright, G.H.V.: Deontic logic. Mind **60**(237), 1–15 (1951)
23. Wright, G.H.V.: Deontic logic: a personal view. Ratio Juris **12**, 26–38 (1999)
24. Gorín, D., Mera, S., Schapachnik, F.: A software tool for legal drafting. In: FLACOS 2011, pp. 1–15. Elsevier (2011)
25. Governatori, G., Hulstijn, J., Riveret, R., Rotolo, A.: Characterising deadlines in temporal modal defeasible logic. In: Orgun, M.A., Thornton, J. (eds.) AI 2007. LNCS (LNAI), vol. 4830, pp. 486–496. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76928-6_50

26. Governatori, G., Rotolo, A.: Justice delayed is justice denied: logics for a temporal account of reparations and legal compliance. In: Leite, J., Torroni, P., Ågotnes, T., Boella, G., van der Torre, L. (eds.) CLIMA 2011. LNCS (LNAI), vol. 6814, pp. 364–382. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22359-4_25
27. Governatori, G., Rotolo, A., Sartor, G.: Temporalised normative positions in defeasible logic. In: ICAIL 2005, pp. 25–34 (2005)
28. Hansen, J.: The paradoxes of deontic logic: alive and kicking. Theoria **72**(3), 221–232 (2006)
29. Hashmi, M., Governatori, G., Wynn, M.T.: Modeling obligations with event-calculus. In: Bikakis, A., Fodor, P., Roman, D. (eds.) RuleML 2014. LNCS, vol. 8620, pp. 296–310. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09870-8_22
30. Hashmi, M., Governatori, G., Wynn, M.T.: Normative requirements for regulatory compliance: an abstract formal framework. Inf. Syst. Front. **18**(3), 429–455 (2015). https://doi.org/10.1007/s10796-015-9558-1
31. Henzinger, T.A.: It's about time: real-time logics reviewed. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 439–454. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0055640
32. Ho, H.-M., Ouaknine, J., Worrell, J.: Online monitoring of metric temporal logic. In: Bonakdarpour, B., Smolka, S.A. (eds.) RV 2014. LNCS, vol. 8734, pp. 178–192. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11164-3_15
33. Ho, H.-M., Ouaknine, J., Worrell, J.: On the expressiveness and monitoring of metric temporal logic. CoRR, abs/1803.02653 (2018)
34. Kanger, S., Kanger, H.: Rights and parliamentarism. Theoria **32**(2), 85–115 (1966)
35. Konur, S.: Real-time and probabilistic temporal logics: an overview. CoRR, abs/1005.3200 (2010)
36. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Longman Publishing Co., Inc. (2002)
37. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. Softw. Tools Technol. Transfer **1**(1), 134–152 (1997)
38. Martínez, E., Díaz, G., Cambronero, M.-E., Schneider, G.: A model for visual specification of E-contracts. In: IEEE SCC 2010, pp. 1–8. IEEE Computer Society (2010)
39. McNamara, P.: Deontic logic. In: Gabbay, D.M., Woods, J., (eds.) Handbook of the History of Logic, vol. 7, pp. 197–289. North-Holland Publishing (2006)
40. Meyer, J.-J., Dignum, F., Johannes, R.: The Paradoxes of Deontic Logic Revisited: A Computer Science Perspective. Technical report UU-CS-1994-38, EWI-IS: Department of Computer Science, University of Utrecht, Utrecht, September 1994
41. Moszkowski, B., Manna, Z.: Reasoning in interval temporal logic. In: Clarke, E., Kozen, D. (eds.) Logic of Programs 1983. LNCS, vol. 164, pp. 371–382. Springer, Heidelberg (1984). https://doi.org/10.1007/3-540-12896-4_374
42. Ničković, D., Piterman, N.: From MTL to deterministic timed automata. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 152–167. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15297-9_13
43. Pace, G.J., Schneider, G.: Challenges in the specification of full contracts. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 292–306. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00255-7_20
44. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: Bonsangue, M.M., Johnsen, E.B. (eds.) FMOODS 2007. LNCS, vol. 4468, pp. 174–189. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72952-5_11

45. Prisacariu, C., Schneider, G.: A dynamic deontic logic for complex contracts. J. Logic Algebraic Program. **81**(4), 458–490 (2012)
46. Stucki, S., Sánchez, C., Schneider, G., Bonakdarpour, B.: Gray-box monitoring of hyperproperties. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 406–424. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_25