Implementing Temporal Defeasible Logic for Modeling Legal Reasoning

Guido Governatori[2], Antonino Rotolo[1], Rossella Rubino[1]

[1] CIRSFID, University of Bologna, Italy
[2] NICTA, Queensland Research Laboratory, Australia

Abstract. In this paper we briefly present an efficient implementation of temporal defeasible logic, and we argue that it can be used to efficiently capture the the legal concepts of persistence, retroactivity and periodicity. In particular, we illustrate how the system works with a real life example of a regulation.

1 Introduction

Defeasible Logic (DL) is based on a logic programming-like language and, over the years, proved to be a flexible formalism able to capture different facets of non-monotonic reasoning (see [4]). Standard DL has a linear complexity [22] and has also several efficient implementations (e.g., [8,3,6,21]).

DL has been recently extended to capture the temporal aspects of several phenomena, such as legal positions [18], norm modifications (e.g., [16]), and deadlines [13]. The resulting logic, called Temporal Defeasible logic (TDL), has been developed to model the concept of temporal persistence within a non-monotonic setting and, remarkably, it preserves the nice computational properties of standard DL [15]. In addition, this logic distinguishes between permanent and transient (non-permanent) conclusions, which makes the language suitable for applications. In the legal domain, typically we have two types of effects. The first type is where normative effects may persist over time unless some other and subsequent events terminate them (example: "If one causes damage, one has to provide compensation"). For the second type we have that normative effects hold at a specific time on the condition that the antecedent conditions of the rules hold and with a specific temporal relationship between the antecedent of the rule and the effect (example: "If one is in a public building, one is forbidden to smoke", that is, if one is a public building at time *t*, then at time *t* one has the prohibition to smoke).

This paper illustrates how [23,24] Java implementation of TDL can be fruitfully applied in the legal domain by discussing the concepts of normative persistence, retroactivity, and periodicity. While the idea of persistence is of paramount importance for modelling the type of normative effects mentioned above, it is still an open question whether we really need to explicitly distinguish between persistent and non-persistent rules, as done in TDL: indeed, it may be the case that persistent effects are simulated by suitable sets of non-persistent rules. Our answer, however, is negative, since the introduction of persistent rules allows for a more efficient computation. As we will see, also the concepts retroactivity and periodicity can be feasibly handled within TDL.

3rd Juris-Informatics Workshop (Jurisin 2009), 2010.

The layout of the paper is as follows. Section 2 briefly presents TDL. Section 3 outlines [23,24]'s Java implementation of TDL. Section 4 tests and validates this implementation against some interesting cases of normative persistence, retroactivity and periodicity: Section 4.1 discusses some aspects of the concept of persistence and its relation with that of retroactivity and shows that our system allows for a very efficient computation of conclusions; Section 4.2 illustrates how to handle the regulation on road traffic restrictions of the Italian town of Piacenza, which specifies normative deadlines and periodical provisions. Some conclusions end the paper.

2 Temporal Defeasible Logic

The language of propositional TDL is based on the concept of *temporal literal*, which is an expression such as l^t (or its negation, $\neg l^t$), where l is a literal and t is an element of a discrete totally ordered set $\mathscr{T}$ of instants of time $\{t_1, t_2, \dots\}$: l^t intuitively means that l holds at time t. Given a temporal literal l the complement $\sim l$ is $\neg p^t$ if $l = p^t$, and p^t if $l = \neg p^t$.

A *rule* is an expression $lbl : A \hookrightarrow^x m$, where lbl is a unique label of the rule, A is a (possibly empty) set of temporal literals, $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$, m is a temporal literal and x is either π or τ signaling whether we have a *persistent* or *transient* rule. *Strict rules*, marked by the arrow $\rightarrow$, support indisputable conclusions whenever their antecedents, too, are indisputable. *Defeasible rules*, marked by $\Rightarrow$, can be defeated by contrary evidence. *Defeaters*, marked by $\leadsto$, cannot lead to any conclusion but are used to defeat some defeasible rules by producing evidence to the contrary. A *persistent* rule is a rule whose conclusion holds at all instants of time after the conclusion has been derived, unless interrupting events occur; *transient* rules establish the conclusion only for a specific instant of time. Thus $ex_1 : p^5 \Rightarrow^\pi q^6$ means that if p holds at 5, then q defeasibly holds at time 6 and continues to hold after 6 until some event overrides it. The rule $ex_2 : p^5 \Rightarrow^\tau q^6$ means that, if p holds at 5, then q defeasibly holds at time 6 but we do not know whether it will persist after 6. Note that we assume that defeaters are only transient: if a persistent defeasible conclusion is blocked at time t by a transient defeater, such a conclusion no longer holds after t unless another applicable rule reinstates it. Furthermore, as we will see, according to the proof conditions for TDL defeaters cannot be used to prove positive conclusions directly, thus, in this respect the distinction between transient and persistent defeaters is irrelevant. In addition, due to the skeptical nature of defeasible logic, negative conclusions can be considered persistent by default, in the sense that if no reason to prove a conclusion is given for an instant t, the conclusion is deemed as not provable at that instant. Thus, also in this case the distinction between persistent and transient defeaters is irrelevant. Finally, given the intended interpretation of rules and their applicability conditions, the effects of defeaters are, essentially, those of transient rules.

We use some abbreviations. Given a rule r and a set R of rules, $A(r)$ denotes the antecedent of r while $C(r)$ denotes its consequent; R^π denotes the set of persistent rules in R, and $R[\psi]$ the set of rules with consequent ψ. R_s, R_{sd} and R_{dft} are respectively the sets of strict rules, defeasible rules, and defeaters in R.

There are in TDL three kinds of features: facts, rules, and a superiority relation among rules. Facts are indisputable statements, represented by temporal literals. The superiority relation describes the relative strength of rules, i.e., about which rules can overrule which other rules. A *TDL theory* is a structure $(F, R, \prec)$, where F is a finite set of facts, R is a finite set of rules and $\prec$ is an acyclic binary superiority relation over R.

TDL is based on a constructive inference mechanism based on tagged conclusions. Proof tags indicate the strength and the type of conclusions. The strength depends on whether conclusions are indisputable (the tag is Δ), namely obtained by using facts and strict rules, or they are defeasible (the tag is ∂). The type depends on whether conclusions are obtained by applying a persistent or a transient rule: hence, conclusions are also tagged with π (persistent) or τ (transient).

Provability is based on the concept of a derivation (or proof) in a TDL theory D. Given a TDL theory D, a *proof* P from D is a finite sequence of tagged temporal literals such that: (1) each tag is either $+\Delta^\pi$, $-\Delta^\pi$, $+\partial^\pi$, $-\partial^\pi$, $+\Delta^\tau$, $-\Delta^\tau$, $+\partial^\tau$, or $-\partial^\tau$; (2) the proof conditions *Definite Provability* and *Defeasible Provability* given below are satisfied by the sequence P[3].

The meaning of the proof tags is a follows:

- $+\Delta^\pi p^{t_p}$ (resp. $+\Delta^\tau p^{t_p}$): we have a definite derivation of p holding from time t_p onwards (resp. p holds at t_p);
- $-\Delta^\pi p^{t_p}$ (resp. $-\Delta^\tau p^{t_p}$): we can show that it is not possible to have a definite derivation of p holding from time t_p onwards (resp. p holds at t_p);
- $+\partial^\pi p^{t_p}$ (resp. $+\partial^\tau p^{t_p}$): we have a defeasible derivation of p holding from time t_p onwards (resp. p holds at t_p);
- $-\partial^\pi p^{t_p}$ (resp. $-\partial^\tau p^{t_p}$): we can show that it is not possible to have a defeasible derivation of p holding from time t_p onwards (resp. p holds at t_p).

The inference conditions for $-\Delta$ and $-\partial$ are derived from those for $+\Delta$ and $+\partial$ by applying the Principle of Strong Negation [5]. For space reasons, in what follows we show only the conditions for $+\Delta$ and $+\partial$.

Definite Provability
If $P(n+1) = +\Delta^x p^{t_p}$, then
1) $p^{t_p} \in F$ if $x = \tau$; or
2) $\exists r \in R_s^x[p^{t'_p}]$ such that
    $\forall a^{t_a} \in A(r) : +\Delta^y a^{t_a} \in P[1..n]$

Defeasible Provability
If $P(n+1) = +\partial^x p^{t_p}$, then
1) $+\Delta^x p^{t_p} \in P[1..n]$ or
2) $-\Delta^x {\sim} p^{t_p} \in P[1..n]$ and
2.1) $\exists r \in R_{sd}^x[p^{t'_p}]$ such that
    $\forall a^{t_a} \in A(r) : +\partial^y a^{t_a} \in P[1..n]$, and
2.2) $\forall s \in R^y[{\sim} p^{t_{\sim p}}]$ either
    2.2.1) $\exists b^{t_b} \in A(s), -\partial^y b^{t_b} \in P[1..n]$ or
    2.2.2) $\exists w \in R^y[p^{t_{\sim p}}]$ such that
        $\forall c^{t_c} \in A(w) : +\partial^y c^{t_c} \in P[1..n]$ and
        $s \prec w$

where (for both proof conditions) (a) $y \in \{\pi, \tau\}$; (b) if $x = \pi$, then $t'_p \leq t_{\sim p} \leq t_p$; (c) if $x = \tau$, then $t'_p = t_{\sim p} = t_p$.

Consider the conditions for definite provability. If the conclusion is transient (if $x = \tau$), the above conditions are the standard ones for definite proofs in DL, which

[3] Given a proof P we use $P(n)$ to denote the n-th element of the sequence, and $P[1..n]$ denotes the first n elements of P.

are just monotonic derivations using forward chaining. If the conclusion is persistent ($x = \pi$), p can be obtained at t_p or, by persistence, at any time t'_p before t_p. Finally, notice that facts lead to strict conclusions, but are taken not to be persistent.

Defeasible derivations run in three phases. In the first phase we put forward a supported reason (rule) for the conclusion we want to prove. Then in the second phase we consider all (actual and potential) reasons against the desired conclusion. Finally in the last phase, we have to rebut all the counterarguments. This can be done in two ways: we can show that some of the premises of a counterargument do not obtain, or we can show that the counterargument is weaker than an argument in favour of the conclusion. If $x = \tau$, the above conditions are essentially those for defeasible derivations in DL. If $x = \pi$, a proof for p can be obtained by using a persistent rule which leads to p holding at t_p or at any time t'_p before t_p. In addition, for every instant of time between the t'_p and t_p, p should not be terminated. This requires that all possible attacks were not triggered (clause 2.2.1) or are weaker than some reasons in favour of the persistence of p (clause 2.2.2). Consider the following theory.

$$\begin{aligned}&(F = \{a^{t_1},\, b^{t_3},\, c^{t_3},\, d^{t_4}\},\\ &\ R = \{r_1 :\ a^{t_1} \Rightarrow^{\pi} e^{t_1}, r_2 :\ b^{t_3} \Rightarrow^{\pi} \neg e^{t_3}, r_3 :\ c^{t_3} \leadsto^{\tau} e^{t_3}, r_4 :\ d^{t_4} \Rightarrow^{\tau} \neg e^{t_4}\},\\ &\ \succ = \{r_3 \succ r_2,\, r_1 \succ r_4\})\end{aligned}$$

At time t_1, r_1 is the only applicable rule; accordingly we derive $+\partial^{\pi} e^{t_1}$. At time t_2 no rule is applicable, and the only derivation permitted is the derivation of $+\partial^{\pi} e^{t_2}$ by persistence. At time t_3 both r_2 and r_3 are applicable, but r_4 is not. If r_2 prevailed, then it would terminate e. However, it is rebutted by r_3, so we derive $+\partial^{\pi} e^{t_3}$. Finally at time t_4, rule r_4 is applicable, thus we derive $+\partial^{\tau} \neg e^{t_4}$ and $-\partial^{\pi} e^{t_4}$, which means that r_4 terminates e. Notice that, even if r_4 is weaker than r_1, the latter is not applicable at t_4, thus it does not offer any support to maintain e.

3 The Implementation

The system implementing TDL consists of three elements: (a) a parser, which loads sets of rules, stored either in plain text format or in RuleML format, and generates a corresponding TDL theory; (b) a Graphical User Interface for selecting defeasible theories, and for visualizing conclusions and the execution time of the algorithm; (c) an inference engine which implements the algorithm of [15] to compute conclusions of the generated TDL theory [23,24].

The parser translates sets of rules in plain text or RuleML formats to generate a corresponding theory to be processed by the inference engine. The plain text format is mostly useful to handle rulesets in a simple presentation syntax easily understandable by human users. RuleML[4] is an open, general and vendor neutral XML/RDF dialect for the representation of rules. The ability to handle sets of rules written in an interchange format is valuable insofar as it allows one to exchange rules across different rule engines and different languages (e.g., W3C RIF [20], R2ML[5] and LKIF [10]) just using XSLT

[4] `http://www.ruleml.org`
[5] `http://rewerse.net/I1`

transformations to translate from one language to another language. Furthermore, the use of XML based languages makes possible interactions of our implementation and Semantic Web technology. For example, it is possible to use the RDF loaders of [3,8,14] to load RDF stores as sets of facts. At the same time, it permits the integration of a non-monotonic reasoner and OWL reasoners [2], thus enabling the use of (legal) ontologies, paving thus the way for the use of TDL in expressive and powerful Semantic Web applications, as well as in SOA based applications.

The Graphical User Interface allows the user to select a set of rules in RuleML or plain text format and to decide the time interval within which to compute their conclusions. Rules are then elaborated by assigning a unique label to each rule and the signs $+/-$ to the literals according to whether they are positive or negative: the rules are visualized accordingly.

The inference engine of the system implements in Java the algorithm for TDL developed by Governatori and Rotolo in [15][6]. The algorithm computes the extension of any TDL theory D, where the concept of extension is defined as follows: if HB_D is the Herbrand Base for D, the extension of D is the 4-tuple $(\Delta^+, \Delta^-, \partial^+, \partial^-)$, where $\#^\pm = \{p^t | p \in HB_D, D \vdash \pm\#^x p^t, t \in \mathscr{T}\}$, $\# \in \{\Delta, \partial\}$, and $x \in \{\pi, \tau\}$. Δ^+ and Δ^- are the positive and negative definite extensions of D, while ∂^+ and ∂^- are the positive and negative defeasible extensions. The extension of a theory D contains the set of all possible conclusions (and their type) that can be derived from the theory. For example, given a theory D, ∂^+, the positive defeasible extension, is the set of the (temporalised) literals that can be proved defeasibly from the theory D, similarly for the other elements of the extension.

The computation of the extension of a TDL theory runs in three steps [15][7]:

(i) in the first step the superiority relation is removed by creating an equivalent theory where $\prec = \emptyset$; any fact a^t, too, is removed by replacing it with a rule $\to^\tau a^t$;
(ii) in the second step the theory obtained from the first phase is used to compute the definite extension;
(iii) in the third step the theory from the first step and the definite extension are used to generate the theory to be used to compute the defeasible extension.

The Java class implementing the algorithm is `TDLEngine`. This has, as its main attributes, the theory (a set of rules and atoms), the theory conclusions, the time interval within which to compute these conclusions, the execution time of the algorithm, and a log manager.

The methods of the class `TDLEngine` are of two types: (i) those that are proper of the algorithm; (ii) those that are functional to the algorithm execution. Here, we will only describe the former ones.

It is worth noting that the computation makes use of time intervals to give a compact representation for sets of contiguous instants. The algorithm works both with proper

[6] See `http://www.defeasible.org/implementations/TDLJava/index.html` for the full code and the Javadoc documentation. See [23] for more details.

[7] Governatori and Rotolo [15] proved that, given a TDL theory D, the extension of D can be computed in linear time, i.e., $O(|R| * |H_D| * |\mathscr{T}_D|)$, where R are the rules of D and $\mathscr{T}_D$ is the set of distinct instants occurring in D. It is also shown that the proposed algorithm is correct.

intervals such as $[t,t']$, i.e., intervals with start time t and end time t', and punctual intervals such as $[t]$, i.e., intervals corresponding to singletons.

Following the idea of [22], the computation of the definite and defeasible extensions is based on a series of (theory) transformations that allow us (1) to assert whether a literal is provable or not (and the strength of its derivation) (2) to progressively reduce and simplify a theory. The key ideas depend on a procedure according to which, once we have established that a literal is positively provable we can remove it from the body of rules where it occurs without affecting the set of conclusions we can derive from the theory. Similarly, we can safely remove rules from a theory when one of the elements in the body of the rules is negatively provable. The methods of `TDLEngine` for this purpose are `computeDefiniteConclusions` and `computeDefeasibleConclusions`.

The method `computeDefiniteConclusions` works as follows. At each cycle, it scans the set of literals of the theory in search of temporal literals for which there are no rules supporting them (namely, supporting their derivation). This happens in two cases: (i) there are no rules for a temporal literal l^t or (ii) all the persistent rules having the literal in their head are parametrized by a greater time than t. For each of such temporal literals we add them to the negative definite extension of the theory, and remove all rules where at least one of these literals occurs. Then, the set of rules is scanned in search of rules with an empty body. In case of a positive match we add the conclusion of the rule to the positive definite extension (with an open ended interval for a persistent rule and with a punctual interval otherwise). Finally we remove such temporal literals matching the newly added conclusions from the body of rules. The cycle is repeated until (1) there are no more literals to be examined, or (2) the set of strict rules is empty, or (3) no addition to the extension happened in the cycle.

The method `computeDefeasibleConclusions` is more complex. As regards the scanning of the set of literals of the theory–in search of temporal literals for which there are no rules supporting them–the procedure is basically the same of `computeDefi-niteConclusions` (with the difference that when we eliminate a rule we update the state of the extension instead of waiting to the end as in the case of the definite extensions). Then we search for rules with empty body. Suppose we have one of such rules, say a rule for l^t, and we know that the complement of l, i.e., $\sim l$, cannot be proved at t. So we add $(\sim l, [t])$ to ∂^-. At this stage we still have to determine whether we can insert l in ∂^+ and the instant/interval associated to it. We have a few cases. The rule for l is a defeater: defeaters cannot be used to prove conclusions, so in this case, we are done. If the rule is transient, then it can prove the conclusion only at t, and we have to see if there are transient rules for $\sim l^t$ or persistent rules for $\sim l^{t'}$ such that $t' \leq t$. If there are we have to wait to see if we can discard such rules. Otherwise, we can add $(l, [t])$ to ∂^+. Finally, in the last case the rule is persistent. What we have to do in this case is to search for the minimum time t' greater or equal to t in the rules for $\sim l$, and we can include $(l, [t,t'])$ in ∂^+.

The method `computeDefeasibleConclusions` basically calls three subroutines: `proved`, `discard`, and `persistence`.

The subroutine corresponding to `persistence` updates the state of literals in the extension of a theory after we have removed the rules in which we know at least one literal in the antecedent is provable with $-\partial^x$. Consider, for example, a theory where the rules for p and $\neg p$ are: $r: \Rightarrow^{\pi} p^1$, $s: q^5 \Rightarrow^{\tau} \neg p^{10}$, $v: \Rightarrow^{\pi} \neg p^{15}$. In this theory we can prove $+\partial^{\pi} p^t$ for $1 \leq t < 10$, no matter whether q is provable or not at 5. Suppose that we discover that $-\partial^x q^5$. Then we have to remove rule s. In the resulting theory from this transformation can prove $+\partial^{\pi} p^t$ for $1 \leq t < 15$. Thus we can update the entry for l from $(l,[1,10])$ to $(l,[1,15])$.

Secondly, `discard` adds a literal to the negative defeasible extension and then removes the rules for which we have already proved that some literal in the antecedent of these rules is not provable. The literal is parametrised by an interval. Then it further calls `persistence` that updates the state of the extension of a theory.

Third, `proved` allows to establish if a literal is proved with respect to a given time interval I. As a first step, it inserts a provable literal in the positive defeasible extension of the theory. Then it calls `discard` with the complementary literal. The next step is to remove all the instances of the literal temporalised with an instant in the interval I from the body of any rule. Finally, the rule is removed from the set of rules.

4 Validation and Testing in the Legal Domain

The main contribution of this paper is the validation in the legal domain of [23,24]'s Java implementation of TDL.

We are still at a preliminary stage for a general testing the system. In particular, we have not yet done a systematic performance evaluation using tools generating scalable test defeasible logic theories: this study is a matter of further research.[8] In this section we test and validate the implementation (and the logic) with regard to three complex temporal phenomena occurring in the legal domain, i.e., persistence, retroactivity, deadlines and periodicity. To test persistence and retroactivity we have generated some synthetic theories modelling various features of the logic and we have examined the difference of computing persistence directly or via computation for all instants in an interval. The test for deadlines and periodicity are based on a theory encoding a real life scenario.

4.1 Persistence and Backward Persistence in Legal Reasoning

We have generated some theory types (exemplified below), and for each of them we have instantiated the set of rules and computed the conclusions in the interval $[0,100]$.

- **Persistence**
  - Rules: $\Rightarrow^{\pi} a^0$
  - Output: $(a,[0,100])$
- **Backward Persistence Persistence**
  - Rules: $a^x \Rightarrow^{\pi} a^{x-1}$, $\Rightarrow^{\tau} a^{100}$
  - Output: $(a,[0,100])$
- **Backward Persistence Lazy**
  - Rules: $a^{100} \Rightarrow^{\pi} a^0$, $\Rightarrow^{\tau} a^{100}$
  - Output: $(a,[0,100])$
- **Persistence via Transient**
  - Rules: $a^x \Rightarrow^{\tau} a^{x+1}$, $\Rightarrow^{\tau} a^0$
  - Output: $(a,[0,100])$
- **Backward Persistence Transient**
  - Rules: $a^x \Rightarrow^{\tau} a^{x-1}$, $\Rightarrow^{\tau} a^{100}$
  - Output: $(a,[0,100])$
- **Backward Opposite Persistence**
  - Rules: $a^{100} \Rightarrow^{\pi} \neg a^0$, $\Rightarrow^{\tau} a^{100}$
  - Output: $(a,[100])$, $(\neg a,[0,99])$

[8] Some first non-systematic tests on large theories are encouraging: random-generated theories with about 100,000 rules, 1,000,000 of atoms, and 100 instants of time show computation times comparable to the result of [21] for theories of similar size. See [23] for a discussion.

Our preliminary experiments, reported in Table 1, were performed on an Intel Core Duo (1,80 GHz) with 3 GB main memory. The results about the execution time are fully alligned with the theoretical result about the linear computational complexity of TDL (see footnote 7), with minor variations mostly due start-up time.

Theory	**Rules**	**Atoms**	**Time**	**Execution time**
Backward Persistence Transient	101	1	[0,100]	1110 ms
Backward Persistence Persistence	101	1	[0,100]	984 ms
Backward Opposite Persistence	2	1	[0,100]	15 ms
Backward Persistence Lazy	2	1	[0,100]	32 ms
Persistence via Transient	101	1	[0,100]	1250 ms
Persistence	1	1	[0,100]	16 ms

Table 1. Performances on Persistence

Arguably, one the most distinctive features of legal reasoning is that some normative effects persist over time unless some other and subsequent events terminate them, while other effects hold on the condition and only while the antecedent conditions of the rules hold. This is reflected in TDL in the distinction between persistent and transient rules. However, we may have different ways the logic can handle this notion of persistence. One could argue that persistence is not necessary since the logic is meant to give a response for a query (i.e., whether a conclusion holds) given a start time and an instant in which to evaluate a formula and the time line is taken as a discrete total order, thus persistence could be simulated using transient rules. In particular for each (persistent) literal a, one could introduce a set of rules $a^t \Rightarrow a^{t+1}$, for t in the fixed time interval.

Unfortunately, the approach using transient rules suffers from two main limitations: (1) the approach is not efficient: compare, in Table 1, the execution times for 'Persistence' and 'Persistence via Transient'. Indeed, persistence allows us to adopt a concise and computationally efficient encoding for the representation of the phenomenon. (2) fixing the time interval based on the start time and the evaluation time of a conclusion is not enough. For example, it is not enough to consider the interval $[0,100]$ if one wants to know if a holds at time 100. Consider for example the theory $a^1 \Rightarrow^\pi b^{10}$, $b^{99} \Rightarrow^\tau c^{1000}$, $c^{1000} \Rightarrow^\tau a^{100}$. In this theory, limiting to the generation of the transient rules in $[0,100]$ does not lead to the right results. Using the 'persistence via transient' method we have to generate all rules in the interval $[0,1000]$, with a consequent, useless increase of the execution time.

The second problem reported above is due to the possibility of having rules where the time labelling the conclusion precedes some of the times in the antecedent (i.e., retroactivity). In legal reasoning it is not unusual to obtain conclusions about the past.[16] This means that a norm is introduced at a particular time, but its normative effects must be considered at times preceding the validity. In fact, this is typical, e.g., of taxation law. A common example is the introduction of norms whose validity is retroactive: for instance, it is possible to claim a tax benefit from a date in the past[9]. Even though trivial cases of this phenomenon are captured by single rules whose conclusions hold at times preceding some of the times of the antecedents, we should be able to detect retroactivity also in other scenarios, where normative effects are in fact applied retroactively to some conditions as a result of complex arguments that involve more rules, such as $a^{10} \Rightarrow^{\pi} b^{10}$ and $b^{10} \Rightarrow^{\pi} c^{0}$. This problem is of great importance not only because the designer of a normative system may have the goal to state retroactive effects in more articulated scenarios of taxation law, but also because she should be able to check whether such effects are not obtained when certain regulations regard matters for which retroactivity is not in general permitted. This is the case of criminal law, where the principle *Nullum crimen, nulla poena sine praevia lege poenali* is valid.

Modelling retroactivity is challenging if it is combined with the notion of persistence. In our synthetic experiments we focused on some types of backward persistence, where conclusions persist from times which precede the ones when rules leading to such conclusions apply. As expected, all cases of backward persistence where conclusions are re-used to derive persistent literals (‘Backward Persistence Transient’ and ‘Backward Persistence Persistence’) are more computationally demanding, while the other cases, where literals persist by default, are comparable to standard persistence (the last row in Table 1). In the theories ‘Backward Persistence Transient/Persistence’ we perform a regression from one instant to the previous instant, and the we compute the conclusions, thus the difference between persistence and transient is not relevant. On the contrary, in ‘Backward Persistence Lazy’ and ‘Backward Opposite Persistence’ we first go back in the past and then we set persistence: for this reason, the performance of the system is here in line with that of the case ‘Persistence’. However, while it is not investigated in this paper, to address the problem of (persistent) regression, one could define a logic, where conclusion are persistent in the past, and this can be achieved with the same mechanism we use to deal with persistence in the future (all one need is a discrete linearly order set of instants).

4.2 A Real-life Scenario: Road Traffic Restrictions of Piacenza

In this paper we report our test with some real-life scenarios [23]. One of them is particularly significant. It formalizes the regulation on road traffic restrictions of the Italian town of Piacenza. This case illustrates how the logic and its implementation behave in handling the concepts of deadline and periodical provision. As regards the former concept, Piacenza regulation contains several instances of so-called maintenance obligations, which state that a certain condition must obtain during all instants before the

[9] Consider for example Section 165–55 of the Australian Goods and Services Tax Act 1999 prescribing: “For the purpose of making a declaration under this Subdivision, the Tax Commissioner may: (a) treat a particular event that actually happened as not having happened; and (b) treat a particular event that did not actually happen as having happened and, if appropriate, treat the event as: (i) having happened at a particular time; and (ii) having involved particular action by a particular entity; and (c) treat a particular event that actually happened as: (i) having happened at a time different from the time it actually happened; or (ii) having involved particular action by a particular entity (whether or not the event actually involved any action by that entity.”

deadline [13]. On the other hand, the regulation includes an example of periodical provision, stating that a certain legal effect periodically occurs because of intermittent character of the pre-conditions of this effect.

The road traffic restrictions regulation of the town of Piacenza consists of the following rules:

N1. From 1 October 2008 to 1 March 2009 all vehicles Euro 0, diesel Euro 1 are prohibited from circulating in the city centre.
N2. The prohibition of clause N1 is suspended in occasion of public holidays.
N3. From 7 January 2009 to 31 March 2009 vehicles diesel Euro 2 without particulate filters are prohibited from circulating in the city centre.
N4. From 8 January 2009 to 31 March 2009, on all Thursdays all vehicles are prohibited from circulating in the city centre.
N5. All vehicles Euro 4 and Euro 5, and the vehicles diesel Euro 3 with particulate filter are permitted to circulate in the city centre.
N6. WARNING: on the occasion of the snowfall of 7 January 2009, traffic restrictions scheduled for Thursday 8 January 2009 do not apply.

For the representation of the above regulation in TDL we assume that 1/10/2008 corresponds to instant 1, while the other dates are associated to integers according to the time granularity Day.

The regulation above corresponds to the following TDL rules:

$$\begin{array}{ll}
r_1 : \ \Rightarrow^{\pi} \mathit{trafficBlock}^{1} & r_9 : e2 \Rightarrow^{\pi} \neg \mathit{restricted}^{180} \\
r_2 : \ \Rightarrow^{\pi} \neg \mathit{trafficBlock}^{151} & r_{10} : \ \Rightarrow^{\pi} \mathit{circulate}^{x} \\
r_3 : \ \Rightarrow^{\pi} \neg \mathit{restricted}^{1} & r_{11} : \mathit{restricted}^{x}, \mathit{trafficBlock}^{x} \Rightarrow^{\pi} \neg \mathit{circulate}^{x} \\
r_4 : e0 \Rightarrow^{\pi} \mathit{restricted}^{1} & r_{12} : \mathit{thursday}^{x} \Rightarrow^{\tau} \neg \mathit{circulate}^{x} \\
r_5 : e1 \Rightarrow^{\pi} \mathit{restricted}^{1} & r_{13} : \mathit{thursday}^{x} \Rightarrow^{\pi} \mathit{circulate}^{x+1} \\
r_6 : e3 \Rightarrow^{\pi} \mathit{restricted}^{1} & r_{14} : \mathit{festivity}^{x} \Rightarrow^{\tau} \neg \mathit{trafficBlock}^{x} \\
r_7 : e3, \mathit{filter} \Rightarrow^{\pi} \neg \mathit{restricted}^{1} & r_{15} : \mathit{festivity}^{x} \Rightarrow^{\pi} \mathit{trafficBlock}^{x+1} \\
r_8 : e2, \neg \mathit{filter} \Rightarrow^{\pi} \mathit{restricted}^{99} & r_{16} : \mathit{snowfall}^{x} \Rightarrow^{\tau} \neg \mathit{circulate}^{x+1}
\end{array}$$

The meaning of the propositions in the above set of rules is given in Table 2.

Temporal literal	**Meaning**
$\mathit{trafficBlock}^t$	the traffic block restrictions are in force at time t
$e0$, $e1$, $e2$, $e3$, $e4$, $e5$	type of vehicles according to Euro classification
filter	whether a vehicle is equipped with a particulate filter or not
$\mathit{restricted}^t$	whether a vehicle is subject to traffic restrictions a time t
$\mathit{circulate}^t$	whether a vehicle is permitted to circulate in the city centre at time t
$\mathit{thursday}^t$	it evaluates to true if t is a Thursday
$\mathit{festivity}^t$	it evaluates to true is t is a gazetted holiday
$\mathit{snowfall}^t$	whether a snowfall happened at time t

Table 2. Meaning of Predicates in the Piacenza Traffic Regulation

In the above set of rules, rules 1–9 are single rules (i.e., instances of rules), while rules 10–16 are schemata, where the temporal variables have to be instantiated. For the superiority relation we have[10]

$$r_4, r_5, r_6 \succ r_3, \quad r_7 \succ r_6, \quad r_2 \succ r_{15}, \quad r_{11} \succ r_{10}, \quad r_{11} \succ r_{13}, \quad r_{14} \succ r_{15}, \quad r_{16} \succ r_{11}, r_{12}$$

Rules r_{12} and r_{13} could have been simply replaced by rules with empty antecedents and whose conclusion is the literals temporalised with the day number, similarly for rules r_{14} and r_{15}.

The above set of rule exhibits several typical features of reasoning with time and norms.[11] Norms (and their effects) have a time of efficacy. This can be expressed by deadlines. Thus for example, we can consider the first norm, N1, whose efficacy is from 1 October 2008 to 1 march 2009. This is represented by rule r_1, saying that from time 1 the traffic block restrictions are in force, and they will stay in force until they are interrupted or terminated. Where the termination is represented by rule r_2: the traffic restrictions are no longer effective from day 151. However, traffic block conditions can be suspended in the circumstances defined by rule r_{12} (norm N4) and rule r_{16} (Warning, N6). Notice that these two norms suspend the general traffic block restrictions, but do not terminate the efficacy of the norm. In temporal defeasible logic, we have that we can use rule r_2 to derive that *trafficBlock* holds from time 1, so it persists till a rule for ¬*trafficBlock* becomes applicable. At that time, we conclude ¬*trafficBlock*; a 'fresh' conclusion takes precedence over a conclusion persisting from the past. However, the new conclusion is transient. After, we have to reinstate the *trafficBlock* conclusion. This is done, by a rule, i.e., r_{15}, with the same antecedent of the suspending rule, i.e., r_{14}.

The traffic regulation shows a very common feature of normative systems, norms can be enacted at different time and can have different validity time. In the regulation at hand norms N1–N4 are all enacted at the same time. However, norms N2 and N3 are in force (i.e., they can produce normative effects), only after day 99 and day 100.

Rule r_{11} could be instantiated for all days, and then we could reinstate the prohibition to enter in the city centre many times. However, it is not necessary to instantiate it for all days, all we have to do is to instantiate it for days just after turning points for the temporal literals $\mathit{restricted}^t$, $\mathit{trafficBlock}^t$, and $\mathit{circulate}^t$.[12] Thus, for our example, given that we know the gazetted public holidays we can instantiate

$$\begin{aligned}
&r_{10}^{87} : \Rightarrow^{\pi} \mathit{circulate}^{87} \\
&r_{10}^{88} : \Rightarrow^{\pi} \mathit{circulate}^{88} \\
&r_{14}^{87} : \mathit{festivity}^{87} \Rightarrow^{\tau} \neg\mathit{trafficBlock}^{87} \\
&r_{14}^{87} : \mathit{festivity}^{87} \Rightarrow^{\tau} \mathit{trafficBlock}^{88} \\
&r_{11}^{87} : \mathit{restricted}^{87}, \mathit{trafficBlock}^{87} \Rightarrow^{\pi} \neg\mathit{circulate}^{87} \\
&r_{11}^{88} : \mathit{restricted}^{88}, \mathit{trafficBlock}^{88} \Rightarrow^{\pi} \neg\mathit{circulate}^{88}
\end{aligned}$$

[10] For rule schemata, we use the superiority to mean that each instance of a schema is superior to any instance of a second schema.

[11] For a comprehensive discussion of requirements for conceptual representation of norms using rules see [11].

[12] A turning point is an instant in time –day, in the granularity of the example–, where a change of provability of a literal could happen.

The intuition of rule r_{17}, which corresponds to the generalisation of norm N6, is similar to the case of the suspension of the traffic block conditions. However, there is a caveat with this rule: We do not know in advance whether there will be a snowfall on a particular day. Thus it is not possible to generate in advance the relevant instances. However, the norm will produce its effect only when we have a fact $snowfall^t$. Thus we can generate dynamically such instances, when the relevant facts are given. We can think of this case as the introduction of the norm at the time of the snowfall. This further illustrate another important aspect to consider when reasoning with time and norms: typically one has to consider at least three temporal dimensions: The time of validity of the norms (when a norm is enacted), the time of force of the norm (when a norm can produce an effect), and the time of efficacy (when a norm produces an effect). A proper model to handle this should consider the view-point at which we look at a system of norm. Thus if I ask whether I can drive my e0 car in the city center of Piacenza on January 8, then, if I ask on January 1, the answer is no, since the norm N5 is not valid, but if I ask during the snowfall of January 7, then the answer is yes, since the norm N5, rule r_{16} become valid, in force and effective.

The model of TDL presented in this paper, and thus the implementation cannot directly handle this, it needs to instantiate the rule dynamically. In [17,16] we have extended TDL to cover the three temporal dimensions. However, currently is it not know, if it is possible to implement the extended TDL efficiently and maintaining good computational properties.

4.3 Validation of TDL

The material presented in this section is a first step towards the empirical validation of TDL as a formalism suitable for modelling and reasoning with and about norms and regulations. The results so far are promising; in particular the synthetic experiment shows:

1. The computation model behind TDL and the infrastructure to consider time does not produce a substantial overhead over standard DL;
2. The introduction of persistence leads to a substantial speed-up in computation, and reduction of the complexity of rule-set. In other term it allows for concise encoding of the formal representation of norms and regulations;
3. The implementation is able to handle large rulesets;
4. Retroactivity does not pose particular concerns both from a conceptual point of view and computationally.

On the other hand, the experiment where we encoded the Traffic Restriction Regulation of Piacenza in TDL shows that TDL is able to handle different phenomena common in norms and regulations: deadlines, interruption of the efficacy of norms, and periodical norms, as well as exceptions, and derogations. Exceptions and derogation follows immediately from the basic properties of standard DL with immediate adaptation to the temporal case.

The analysis of the scenario suggests that TDL is appropriate as computational model for regulations involving temporal references. We notice that some normative aspects require several rules for their representation. However, these constructions exhibit clear and regular patterns for the rules needed to capture them. Thus syntactic forms can be defined for them[13]. This will hidden the apparent complexity of these construction for people without expertise in defeasible logic and in general formal methods.

5 Conclusions

There are two mainstream approaches to reasoning with and about time: a point based approach, as TDL, and an interval based approach [1]. Notice, however, that TDL is able to deal with constituents holding in an interval of time: an expression $\Rightarrow a^{[t_1,t_2]}$, meaning that a holds between t_1 and t_2, can just be seen as a shorthand of the pair of rules $\Rightarrow^{\pi} a^{t_1}$ and $\leadsto^{\tau} \neg a^{t_2}$.

Non-monotonicity and temporal persistence are covered by a number of different formalisms, some of which are quite popular and mostly based on variants of Event Calculus or Situation Calculus combined with non-monotonic logics (see, e.g., [25,26]). TDL has some advantages over many of them. In particular, while TDL is sufficiently expressive for many purposes, it is possible in TDL to compute the set of consequences of any given theory in linear time to the size of the theory. To the best of our knowledge, no logic covering a set concepts comparable to what TDL covers is so efficient (see [9] for a comprehensive list of complexity results for various forms of the Event Calculus).

Temporal and duration based defeasible reasoning has been also developed by [7,19]. [19] focuses on duration and periodicity and relationships with various forms of causality. In particular, [7] proposed a sophisticated interaction of defeasible reasoning and standard temporal reasoning (i.e., mutual relationships of intervals and constraints on the combination of intervals). In these cases no complexity results are available, but these systems cannot enjoy the same nice computational properties of TDL, since both are based on more complex temporal structures.

On account of the feasibility of TDL, in this paper we reported on [23,24]'s Java implementation of this logic. In particular, we tested and validated the implementation (and the logic) with regard to three complex temporal phenomena occurring in the legal domain, i.e., persistence, retroactivity, deadlines and periodicity. Our results are encouraging and so we plan to extend the system in order to handle time plus deontic operators (see [21]) and to add temporal parameters, such as in expressions like $(a^t \Rightarrow^{\pi} b^{t'})^{t''}$, where t'' stands for the time when the norm is in force [16].

Acknowledgements

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

References

1. J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23, 1984.

[13] For a discussion about patterns for modelling deadline and norms see [13], and for periodicity, see [19].

2. G. Antoniou. A nonmonotonic rule system using ontologies. In *Proc. RuleML 2002*, volume 60 of *CEUR Workshop Proceedings*, 2002.
3. G. Antoniou and A. Bikakis. DR-Prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering*, (2):233–245, 2007.
4. G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2:255–287, 2001.
5. G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Embedding defeasible logic into logic programming. *Theory and Practice of Logic Programming*, 6:703–735, 2006.
6. M. R. Antoniou, M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10, 2001.
7. J. Augusto and G. Simari. Temporal defeasible reasoning. *Knowledge and Information Systems*, 3:287–318, 2001.
8. N. Bassiliades, G. Antoniou, and I. Vlahavas. A defeasible logic reasoner for the Semantic Web. *International Journal on Semantic Web and Information Systems*, 2:1–41, 2006.
9. I. Cervesato, M. Franceschet, and A. Montanari. A guided tour through some extensions of the event calculus. *Computational Intelligence*, 16(2):307–347, 2000.
10. ESTRELLA Project. The reference LKIF inference engine. Deliverable 4.3, European Commission, 2008.
11. T. F. Gordon, G. Governatori, and A. Rotolo. Rules and norms: Requirements for rule interchange languages in the legal domain. In Governatori et al. [12].
12. G. Governatori, J. Hall, and A. Paschke, editors. *Rule Interchange and Applications, International Symposium, RuleML 2009: Proceedings*. Springer, 2009.
13. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proc. Australian AI 2007*, pages 486–496. Springer, 2007.
14. G. Governatori and D. Pham. A semantic web based architecture for e-contracts in defeasible logic. In *RuleML 2005*, pages 145–159. Springer, 2005.
15. G. Governatori and A. Rotolo. Temporal defeasible logic has linear complexity. In *Proceedings NMR 2010*, CEUR Workshops Proceedings, 2010.
16. G. Governatori and A. Rotolo. Changing legal systems: Legal abrogations and annulments in defeasible logic. *The Logic Journal of IGPL*, forthcoming.
17. G. Governatori, A. Rotolo, R. Riveret, M. Palmirani, and G. Sartor. Variants of temporal defeasible logic for modelling norm modifications. In *Proc. ICAIL'07*, pages 155–159, 2007.
18. G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *ICAIL'05*, pages 25–34. ACM Press, 2005.
19. G. Governatori and P. Terenziani. Temporal extensions to defeasible logic. In *Proc. Australian AI 2007*, pages 476–485. Springer, 2007.
20. S. Hawke. Bringing order to chaos: RIF as the new standard for rule interchange. In Governatori et al. [12], page 1. See also `http://www.w3.org/2009/Talks/1105-ruleml/`.
21. H.-P. Lam and G. Governatori. The making of SPINdle. In Governatori et al. [12].
22. M. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1:691–711, 2001.
23. R. Rubino. *Una implementazione della logica defeasible temporale per il ragionamento giuridico*. PhD thesis, CIRSFID, University of Bologna, 2009.
24. R. Rubino and A. Rotolo. A Java implementation of temporal defeasible logic. In Governatori et al. [12].
25. M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, Cambridge, MA, 1997.
26. H. Turner. Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming*, 31(1-3):245–298, 1997.