Computing Temporal Defeasible Logic

Guido Governatori[1] and Antonino Rotolo[2]

[1] NICTA, Australia
[2] CIRSFID and DSG, University of Bologna, Italy

Abstract. We investigate the complexity of temporal defeasible logic, and propose an efficient algorithm to compute the extension of any theory. The logic and algorithm are discussed in regard to modeling deadlines and normative retroactivity.

1 Introduction

Defeasible Logic (DL) [22, 3] is historically the first of a family of approaches based on the idea of logic programming without negation as failure. DL is a simple, efficient but flexible non-monotonic formalism capable of dealing with many different intuitions of non-monotonic reasoning. The logic was designed to be easily implementable right from the beginning, unlike most other approaches, and has a linear complexity [20]. Recent implementations include DR-Prolog [2], DELORES [19], DR-DEVICE [7].

DL proved to be modular and flexible. In particular, propositional DL has been recently extended in two different directions.

In a first case, different types of modal operators (capturing notions such as directed and undirected deontic statements, actions, beliefs, and intentions) have been embedded within propositional DL [**?**,11]. The result was a number of logics having still linear complexity and being able to model the deliberation of cognitive agents and their interplay with normative systems. Some implementations have been recently developed for such logics [**?**,**?**,17].

DL has been also extended to capture temporal aspects of normative reasoning [15, 12]several specific phenomena, such as legal positions [15] and modifications [10, 12], deadlines [8]. Although Temporal Defeasible Logic (TDL) proved to be sufficiently expressive for those purposes, and many variants of it have been proposed accordingly, no systematic investigation on the proof-theoretic and computational properties of TDL has been so far carried out. This paper is a first step in this direction. In particular, we will present an expressive variant of TDL, which is computationally feasible. This variant is able to represent different types of deadlines and capture backward causation and normative retroactivity. We will prove that it is possible in all cases to compute the complete set of consequences of any given TDL theory in linear time, thus preserving the nice computational features of standard DL.

The layout of the paper is as follows. Section 2 describes a variant of TDL. Section 3 considers a first case-study: modeling the concept of deadline. Section 4 investigates the complexity of the logic, and proposes an algorithm to compute the extension of any theory. Section 5 discusses the approach and considers a second case-study: modeling retroactivity. A section on related work ends the paper.

2 Temporal Defeasible Logic (TDL)

Consider a set $\mathscr{P}$ of atomic propositional literals. The language of TDL is based on the concept of *temporalised 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 temporalised 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 (finite, possibly empty) set of temporalised literals, $\hookrightarrow \in \{\rightarrow, \Rightarrow, \leadsto\}$, m is a temporalised 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 or fact 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.

Example 1. Let us consider two examples from the normative domain. Suppose Bob buys today a car. If so, he will have the obligation to pay for that. Clearly, this obligation holds today and will persist afterwards (until Bob pays the car). Hence:

$$r_1{:}\, \mathit{Bob\_Car}^{\mathit{today}} \Rightarrow^\pi \mathrm{OBL}\mathit{Pay}^{\mathit{today}}$$

Suppose now that Bob enters a Catholic church. As long as he is in the church, he is obliged not speak loudly. In this case, the obligation does not persist, but only holds insofar as Bob is in the church:

$$r_2{:}\, \mathit{Bob\_Church}^x \Rightarrow^\tau \mathrm{OBL}\neg \mathit{Speak\_Loudly}^x$$

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, R^τ the set of rules with consequent ψ. R_s, R_{sd} and R_{dft} are respectively the set of strict rules, the set of strict and defeasible rules, and the set of defeaters in R.

Note that we assume that persistent literals, once they are blocked at a certain time t, no longer hold after t unless other applicable rules reinstate them. Indeed, this is the usual rendering of the common sense law of inertia [25]. Hence, we assume accordingly that defeaters are only transient: if a persistent defeasible conclusion is blocked at t by a defeater, there is no need that such a defeater is still applicable after t (thus, the defeater does not need to be persistent).

Example 2. Consider again the rule:

$$r_1{:}\, \mathit{Bob\_Car}^{\mathit{today}} \Rightarrow^\pi \mathrm{OBL}\mathit{Pay}^{\mathit{today}}$$

If Bob pays, then he is no longer obliged to pay:

$$r_3 \colon Pay^x, \mathrm{OBL}Pay^x \leadsto^{\tau} \neg\mathrm{OBL}Pay^{x+1}$$

There is no reason here to say that this obligation to pay will sooner or later come back: if this happens, it implies that something new has happened in the meantime that has triggered again the obligation. Hence, there are good reasons to state that defeaters have not to be persistent, at least in the domain of normative reasoning.

Remark 1. Impeding reasons work as depicted in the example above in many contexts other than the normative domain: indeed, this is the usual rendering of the common sense law of inertia [25]. However, there are examples where a reason r to block the persistence of a literal l (for example, state of affairs) is effective with respect to this literal only as long as r's effect persists too: when such an r no longer applies, then l comes back by inertia without any further positive reason. Consider the example of Bob, a person who sleeps because he under the effect of of a drug. Bob is asleep as long as the drug is active: when it is no longer active, Bob wakes up. Intuitively, a simple way to model this scenario in TDL is as follows:

$$r_4 \colon \neg Drug\_Active^x \Rightarrow^{\pi} Awake^x$$
$$r_5 \colon Drug\_Active^y \leadsto^{\mathbf{z}} \neg Awake^y$$

Should r_5 be persistent or transient? If persistent, it would block the inertia of *Awake* indefinitely, but this is not required: in this context, we can simply say that a precondition for Bob to be awake is that no drug is active on him. So we can capture the scenario by stating that $z = \tau$.

There are three kinds of features in TDL: facts, rules, and a superiority relation among rules. Facts are indisputable statements, represented by temporalised literals. The superiority relation ($\succ$) provides information about the relative strength of rules, i.e., about which rules can overrule which other rules. A knowledge base that consists of these items is called a TDL theory.

Definition 1. *A* TDL theory *is a structure $(F, R, \succ)$, where F is a finite set of facts, R is a finite set of rules and $\succ$ is an acyclic binary relation over R.*

Given a rule r, $r_{inf} = \{s \colon r \succ s\}$ and $r_{sup} = \{s \colon s \succ r\}$.

TDL is based on a constructive inference mechanism based on tagged conclusions. Proof tags indicate the strength 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 ∂).

Provability is defined below and is based on the concept of a derivation (or proof) in a TDL theory D.

Definition 2. *Given a TDL theory D, a* proof *P from D is a finite sequence of tagged temporalised literals such that:*

1. *Each tag is one of the following: $+\Delta$, $-\Delta$, $+\partial$, $-\partial$;*
2. *The proof conditions* Definite Provability *and* Defeasible Provability *given below are satisfied by the sequence P.*

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. The meaning of the proof tags is as follows:

- $+\Delta p^{t_p}$: we have a definite derivation of p holding at time t_p;
- $-\Delta p^{t_p}$: we can show that it is not possible to have a definite derivation of p holding at time t_p;
- $+\partial p^{t_p}$: we have a defeasible derivation of p holding at time t_p; in other terms p is provable at time t_p.
- $-\partial p^{t_p}$: we can show that it is not possible to have a defeasible derivation of p holding at time t_p; thus p is refuted at time t_p.

Note that the inference conditions for negative proof tags ($-\Delta$ and $-\partial$) are derived from the inference conditions for the corresponding positive proof tags by applying the Principle of Strong Negation introduced by [4]: the strong negation of a formula is closely related to the function that simplifies it by moving all negations to an innermost position in the resulting formula and replace the positive tags with the respective negative tags and viceversa.

Definite Provability

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

If $P(n+1) = -\Delta p^{t_p}$, then
1) $p^{t_p} \notin F$; and
2) $\forall r \in R_s^x[p^{t'_p}]$,
$\exists a^{t_a} \in A(r)$: $-\Delta a^{t_a} \in P[1..n]$.

where:

(a) if $x = \pi$, then $t'_p \leq t_p$;
(b) if $x = \tau$, then $t'_p = t_p$.

If the rule used to derive p is transient (if $x = \tau$), the above conditions are the standard ones for definite proofs in DL, which are just monotonic derivations using forward chaining. If the rule 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. This condition can be relaxed or we could introduce persistent and well as transient facts.

Defeasible Provability

If $P(n+1) = +\partial p^{t_p}$, then
1) $+\Delta p^{t_p} \in P[1..n]$ or
2) $\exists r \in R_d^x[p^{t'_p}]$ such that
  1) $-\Delta{\sim}p^{t_{\sim p}} \in P[1..n]$ and
  2) $\forall a^{t_a} \in A(r)$: $+\partial a^{t_a} \in P[1..n]$, and
  3) $\forall s \in R[{\sim}p^{t_{\sim p}}]$ either
    1) $\exists b^{t_b} \in A(s), -\partial b^{t_b} \in P[1..n]$ or
    2) $\exists w \in R[p^{t_{\sim p}}]$ such that $\forall c^{t_c} \in A(w)$:
    $+\partial c^{t_c} \in P[1..n]$ and $w \succ s$.

If $P(n+1) = -\partial p^{t_p}$, then
1) $-\Delta p^{t_p} \in P[1..n]$ and
2) $\forall r \in R_d^x[p^{t'_p}]$ either
  1) $+\Delta{\sim}p^{t_{\sim p}} \in P[1..n]$ or
  2) $\forall r \in R_d^x[p^{t'_p}]$ or
  3) $\exists s \in R[{\sim}p^{t_{\sim p}}]$ such that
    1) $\forall b^{t_b} \in A(s), +\partial b^{t_b} \in P[1..n]$ and
    2) $\forall w \in R[p^{t_{\sim p}}]$ $\exists c^{t_c} \in A(w)$:
    $-\partial c^{t_c} \in P[1..n]$ or $w \not\succ s$.

where

(1) if $x = \pi$, then $t'_p \leq t_{\sim p} \leq t_p$;
(2) if $x = \tau$, then $t'_p = t_{\sim p} = t_p$.

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 possible (actual and not) 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 argument 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 2.3.1) or are weaker than some reasons in favour of the persistence of p (clause 2.3.2).

Given a rule r and a derivation P, we will say that r is applicable in P, or simply applicable when the derivation is clear from the context, iff $\forall a^t \in A(r), +\partial a^t \in P[1..n]$ for some $n \in \mathbb{N}$.

Example 3. Consider the following theory, where $t_1 < t_2 < t_3 < t_4$:

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

At time t_1, r_1 is the only applicable rule; hence, we derive $+\partial e^{t_1}$. At time t_2 no rule is applicable, and the only derivation permitted is the one of $+\partial e^{t_2}$ 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 $+\partial e^{t_3}$. At time t_4, rule r_4 is applicable, thus we derive $+\partial\neg e^{t_4}$ and $-\partial e^{t_4}$, which means that r_4 terminates e. 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.

Proposition 1. *Let D be a TDL theory.*

1. *It is not possible that both $D \vdash +\#p^t$ and $D \vdash -\#p^t$ (for $\# \in \{\Delta, \partial\}$);*
2. *if $D \vdash +\partial p^t$ and $D \vdash +\partial{\sim}p^t$, then $p^t, {\sim}p^t \in F$.*

The proof of Proposition 1 is a simple extension of the ones for Theorems 1 and 2 in [11] and is here omitted for space reasons. Proposition 1 shows the soundness of TDL: it is not possible to derive a tagged conclusion and its opposite, and that we cannot defeasibly prove both p and its complementary unless the definite part of the theory proves them; this means that inconsistency can be derived only if the theory we started with is inconsistent, and even in this case the logic does not collapse to the trivial extensions (i.e., everything is provable).

Definition 3. *Let HB_D be the Herbrand Base for a TDL theory D. The extension of D (denoted by E^D) is the 4-tuple $(\Delta^+, \Delta^-, \partial^+, \partial^-)$, where*

$$\partial^+ = \{p^t | p \in HB_D, D \vdash +\partial p^t, t \in \mathscr{T}\},\quad \partial^- = \{p^t | p \in HB_D, D \vdash -\partial p^t, t \in \mathscr{T}\},$$
$$\Delta^+ = \{p^t | p \in HB_D, D \vdash +\Delta p^t, t \in \mathscr{T}\},\quad \Delta^- = \{p^t | p \in HB_D, D \vdash -\Delta p^t, t \in \mathscr{T}\}.$$

We will refer to Δ^+ and Δ^- as the definite extension, to ∂^+ and ∂^- as the defeasible extension, to Δ^+ and ∂^+ as the positive extension, and to Δ^- and ∂^- as the negative extension.

3 Computing Consequences in TDL

In this section we present an algorithm to compute the extension of a TDL theory. We show that the time complexity of the algorithm is linear to the size of the theory. Following the idea of [20] the algorithm 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) and (2) to progressively reduce and simplify a theory.

At this point we have to make precise what we mean for two theories to be equivalent, and we formally define the notion of transformation.

Definition 4. *Two theories D and D' are* equivalent *if and only if they have the same extension, namely $D \equiv D'$ iff $E^D = E^{D'}$.*

The key ideas behind the approach depends on the following properties that allow us to transform theories into 'simpler' ones (i.e., either with rules with less elements in their antecedent or with less rules).

Proposition 2. *Let $D = (\emptyset, R, \succ)$ be a temporal defeasible theory*[3]*. Then*

1. *If $r\colon \to^{\pi} p^t \in R$ then $D \vdash +\Delta p^{t'}$, $\forall t' \geq t$.*
2. *If $r\colon \to^{\tau} p^t \in R$ then $D \vdash +\Delta p^t$.*
3. *If $D \vdash +\Delta p^t$, then $D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \to c^{t_c}\} \equiv D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n} \to c^{t_c}\}$*
4. *Let $t_{min} = \min\{t\colon R_s[p^t] \neq \emptyset\}$, then $D \vdash -\Delta p^{t'}$ $\forall t' < t_{min}$.*
5. *Let $t_\pi = \min\{t\colon R_s^{\pi}[p^t]\}$, $t_\tau^- = \min\{t\colon R_s^{\tau}[p^t] \neq \emptyset, t > t_{min}\}$, $t_\tau^+ = \max\{t\colon R_s^{\tau}[p^t] \neq \emptyset, t < t_\pi\}$. Then $D \vdash -\Delta p^{t'}$, $\forall t'$ such that either $t_{min} < t' < t_\tau^-$ or $t_\tau^+ < t' < t_\pi$.*
6. *If $D \vdash -\Delta p^t$, then $D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \to c^{t_c}\} \equiv D \cup \{r\colon a_1^{t_1}, \ldots, a_n^{t_n}, p^t \Rightarrow c^{t_c}\}$*

The properties in Proposition 2 give us an alternative characterisation of the proof theory for definite conclusions. Properties 1 and 2 provide conditions under which we are allowed to positively assert a definite conclusion. Properties 3 presents the condition for the transformation. In fact it tell us that we can remove already proved temporal literals from the body of other (strict) rules without affecting the conclusions we can derive from a theory. Thus we can use Properties 1–3 to transform a theory into a simpler but equivalent theory.

Properties 4–6, on the other hand, are related to transformations and conditions to derive negative definite conclusions. Properties 4, establishes that for all times less than the minimum time appearing associated to a literal in the conclusion of a rules, we fail to prove literal, thus we can assert the literal with $-\Delta$. The meaning of Property 5 is similar to that of Property 4, but this time, the focus is for the times in an interval; in particular it states that for all instants between two persistent rules (for a literal p) such that no other rule is that interval (provided that the smallest instant for a persistent rule for the same literal is after the last of the right extreme of the interval), we can assert that the literal is provable with $-\Delta$. Similarly for an interval between the smallest instant for a persistent rule and a transient rule. Finally, Property 6 tells us that after we have assessed that a strict rule is not applicable for the computation of definite conclusion, we can transform the rule into a defeasible rule.

[3] Notice that the restriction to theories with an empty set of facts is not a limitation. For any theory $D = (F, R, \succ)$ there is an equivalent theory $D' = (\emptyset, R \cup \{\to^{\tau} a^t \colon a^t \in F\}. \succ)$ where the set of facts is empty.

Proposition 3. *Let D be a theory in TDL. For $y \in \{\pi, \tau\}$:*
(1) If $D \vdash +\partial p^t$, then $D \cup \{r : p_1^{t_1}, \ldots, p_n^{t_n}, p^t \Rightarrow^y q\} \equiv D \cup \{r : p_1^{t_1}, \ldots, p_n^{t_n} \Rightarrow^y q\}$.
(2) if $D \vdash -\partial p^t$, then $D \cup \{r : p_1^{t_1}, \ldots, p_n^{t_n}, p^t \Rightarrow^y q\} \equiv D$.

The meaning of (1) in the above proposition is that once we have established that a temporalised literal is positively provable we can remove it from the body of rules without affecting the set of conclusions we can derive from the theory. Similarly (2) states that we can safely remove rules from a theory when one of the elements in the body of the rules is negatively provable.

To give conditions under which we can conclude that a temporal literal is defeasibly provable, we use the notion of *inferiorly defeated* set introduced in [18]. The set of inferiorly defeated rules, R_{infd}, is thus defined $R_{infd} = \{r : \exists s, s \succ r, \text{ and } A(r) = \emptyset\}$.

Proposition 4. *Let D be a TDL theory. If $r : \ \Rightarrow^x p^t \in R$ and $R[\sim p^t] \subseteq R_{infd}$, then $D \vdash +\partial p^t$ and $D \vdash -\partial \sim p^t$.*

This proposition gives us the main criterion to assess whether we can defeasibly prove a literal.

We compute the extension of a TDL theory in two phases:

1. in the first phase we compute the definite extension;
2. in the second phase we use the theory from the first phase to generate the theory to be used to compute the defeasible extension.

3.1 Computing the Definite Extension

Before giving the algorithm to compute the extension we have to introduce some auxiliary notation to refer to the data structures needed for the algorithms that compute the extension.

Definition 5. *Let D be a TDL theory and H_D be the set of literals in D. For each $a \in H_D$ we have the following sets:*

- *$ptimes(a) = \{t : \exists r \in R^\pi[a^t]\}$;*
- *$ttimes(a) = \{t : \exists r \in R^\tau[a^t]\}$;*
- *$times(a) = ptimes(a) \cup ttimes(a)$.*

We use the same abbreviations as those of Section 2. Thus, e.g., $ptimes_s(a)$ is the set of instants associated to $R_s[a]$.

In the presentation of the algorithms we use intervals to give a compact representation for sets of contiguous instants. We will use both proper intervals, i.e., intervals with both start and end time, and punctual intervals, i.e., intervals corresponding to singletons. We will use $[t, t']$ for a proper interval and $[t]$ for a punctual interval.

Definition 6. *Given an interval I we say that $t^* \in I$ iff*
(1) if $I = [t, t']$, $t < t'$ and $t \leq t^* < t'$ or
(2) if $I = [t]$ and $t^* = t$.

We adopt a compact but isomorphic representation for the extensions, namely, elements of extensions are now pairs (l,I) were l is a literal and I is an interval; thus (l,I) corresponds to the set of temporalised literals l^t such that $t \in I$.

Algorithm 1: *ComputeDefinite*

```
Input: A Temporal Defeasible Theory $D = (F, R, \succ)$
$\Delta^+ \leftarrow \{(a,[t]) \colon a^t \in F\}$
$\Delta^- \leftarrow \{(a,[0,\infty] \colon R_s[a] = \emptyset)\}$
$H_D \leftarrow H_D \setminus \{a \colon R_s[a] = \emptyset\}$
while $\Delta_+ \neq \emptyset$ do
    $\Delta_+ \leftarrow \emptyset$
    for $a \in H_D$ do
        $R_s \leftarrow R_s \setminus \{s \colon a^{t'} \in A(s), t' < \min(ptimes(a)$ and $t' \notin ttimes(a))\}$
        if $times(a) = \emptyset$ then
            $H_d \leftarrow H_d \setminus \{a\}$
    for $r \in R_s$ do
        if $A(r) = \emptyset$ and $C(r) = a^t$ then
            if $r \in R^\pi$ then
                $\Delta_+ \leftarrow \Delta_+ \cup \{(a,[t,\infty])\}$
                $R_s \leftarrow R_s \setminus \{s \colon C(s) = a^{t'}, t' \geq t\}$
                $R_s \leftarrow \{r \colon A(r) \setminus \{l^{t'}\} \rightarrow C(s) \colon r \in R_s, t' \geq t\}$
            if $r \in R^\tau$ then
                $\Delta_+ \leftarrow \Delta_+ \cup \{(a,[t])\}$
                $R_s \leftarrow R_s \setminus \{s \colon C(s) = a^t\}$
                $R_s \leftarrow \{r \colon A(r) \setminus \{l^t\} \rightarrow C(s) \colon r \in R_s\}$
    $\Delta^+ \leftarrow \Delta^+ \cup \Delta_+$
for $a \in H_D$ do
    $t_{left} \leftarrow \max(0, \{t \colon (a,[t',t]) \in \Delta^-\})$
    $t_{right} \leftarrow \min(0, \{t \colon (a,[t,\infty]) \in \Delta^+\})$
    $T_\tau \leftarrow \{t \colon (a,[t]) \in \Delta^+, t_{left} < t < t_{right}\}$
    $T_\tau \leftarrow T_\tau \cup \{t \colon C(r) = a^t, r \in R_s^\tau[a], t_{left} < t < t_{right}\}$
    $T \leftarrow T_\tau \cup \{t \colon C(r) = a^t, r \in R_s^\pi[a], t_{left} < t < t_{right}\}$
    while $T_\tau \neq \emptyset$ do
        $t_{min} \leftarrow \min(T_\tau)$
        $T_\tau \leftarrow T_\tau \setminus \{t_{min}\}$
        if $t_{min} + 1 \notin T$ and $t_{min} + 1 < t_{right}$ then
            $\Delta^- \leftarrow \Delta^- \cup \{(a,[t_{min}])\}$
```

At each cycle the algorithm *ComputeDefinite* scans the set of literals in search of temporalised literals for which there are no rules for them. This happens in two cases: (i) there are no rules for a temporalised literal or (ii) all the persistent rules for the literal have a greater time. For each of such temporalised literals *ComputeDefinite* adds them to the negative definite extension of the theory, and removes all rules where at least one of these literals occurs.

Then *ComputeDefinite* scans the set of rules in search of rules with an empty body. In case of a positive match the algorithm adds 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 the algorithm removes such temporalised literals matching the newly added conclusions from the body of rules.

We repeat the cycle 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.

Proposition 5. *ComputeDefinite is correct.*

Proof. The correctness of *ComputeDefinite* follows immediately from Proposition 2. The transformation in *ComputeDefinite* correspond to the properties listed in Proposition 2.

3.2 Computing the Defeasible Extension

We are now ready to give the algorithm to compute the defeasible extension of a theory. We first give some subroutines corresponding to theory transformations to be used in the main algorithm. ∂_+ and ∂_- are sets of accumulators for the conclusions proved in each cycle of the main routine.

The first algorithm we consider is concerned with literals to be tagged with $-\partial$.

Algorithm 2: $discard(l,I)$

```
Input: a literal $l$ and an interval $I$
1 $\partial_- \leftarrow \partial_- \cup \{(l,I)\}$
2 $S \leftarrow \{s : l^t \in A(s), t \in I\}$
3 $R \leftarrow R \setminus S$
4 $\succ \leftarrow \succ \setminus \{(r,s),(r,s) : s \in S\}$
5 $persistence(S)$
```

The algorithm *discard* adds a literal to the negative defeasible extension and then removes rules for which we have already proved that some literal in the antecedent of the rules is not provable. The literal is parametrised by an interval. This means that the operation is performed for all instances of the literal temporalised with an instant in the interval. The transformation corresponding to it is justified by Proposition 3.(2). The algorithm further calls the subroutine *persistence* that updates the state of the extension of a theory.

Algorithm *proved* (Algorithm 3) concerns defeasible provable literals.

Algorithm 3: $proved(l,r,I)$

```
Input: a literal $l$, a rule $r$, and and interval $I$
1 $\partial_+ \leftarrow \partial_+ \cup \{(l,I)\}$
2 $discard(\sim l,I)$
3 for $s \in R$ do
4     if $l^t \in A(s)$ and $t \in I$ then
5         $A(s) \leftarrow A(s) \setminus \{l^t\}$
6     $R \leftarrow R \setminus \{r\}$
```

It first inserts a provable literal in the positive defeasible extension of the theory. Then *proved* 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 rules. Finally it removes the rule for the set of rules. The transformations implemented by this algorithm are justified by Propositions 3, (1), and 4.

Algorithm *persistence* updates the state of literals in the extension of a theory after we have removed rules we know cannot longer be fired (i.e., at least one literal in the antecedent of the rule is provable with $-\partial^x$).

Algorithm 4: *persistence*(S)

```
Input: a set of rules $S$
1 for $(l,[t,t'] \in \partial^+)$ do
2     if $s \in S$ and $C(s) = \sim l^{t'}$ then
3         if $t^* = \min\{k \in ttimes(\sim l) : k > t'\}$ then
4             $\partial_+ \leftarrow (\partial_+ \setminus \{(l,[t,t'])\}) \cup \{(l,[t,t^*])\}$
5             $proved(l, \emptyset, [t', t^*])$
```

As we have seen in Section 2 a conclusion proved using a persistent rule persists until it is terminated by another (applicable) rule for its complementary, thus an entry $(l,[t,t'])$ in ∂^+ means that l holds from t to $t'-1$. Hence, there is a rule for $\sim l^{t'-1}$. When we insert $(l,[t,t'])$ in ∂^+ we do not know if the rule for $\sim l^{t'-1}$ is applicable or not. The set S passed as parameter to the algorithm is the set of rules we have discovered to be no longer applicable. At this point we can update the entry for l in ∂^+, and we set it to t'', where t'' is the next instant for which we have a rule for $\sim l$. Consider, e.g., a theory where the rules for p and $\neg p$ are:

$$
\begin{aligned}
r&:\ \Rightarrow^{\pi} p^1,\\
s&:\ q^5 \Rightarrow^{\tau} \neg p^{10},\\
v&:\ \Rightarrow^{\pi} \neg p^{15}.
\end{aligned}
$$

Here, we can prove $+\partial p^t$ for $1 \le t < 10$, no matter whether q is provable at 5 or not. Suppose we discover that $-\partial q^5$. Then we have to remove rule s. In the resulting theory can prove $+\partial p^t$ for $1 \le p < 15$. Thus we can update the entry for l from $(l,[1,10])$ to $(l,[1,15])$.

For *ComputeDefeasible* lines 4–11 are essentially the same as the main loop in *ComputeDefinite*, lines 4–27 (with the difference that when we eliminate a rule we update the state of the extension instead of waiting to the end as we did for definite extensions).

From line 12 we search for rules with empty body. Suppose we have one of such rules, let us say a rule for l^t. If there are no stronger rules for the opposite, no matter what type of rule we have, we know that the complementary of l, i.e., $\sim l$, cannot be proved at t. So we call *discard* with parameter $(\sim l,[t])$, and remove the rule where $\sim l^t$ appears in the body. 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 is a defeater. Defeaters cannot be used to prove conclusions, so in this case, all we can do is to insert all rules weaker than the defeater in the set of inferiorly defeated rules (line 24). 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 $t' \le t$. If there are we have to wait to see if we can discard such rules. Otherwise, we can add $(l,[t])$ to ∂^+ and the related housekeeping. Finally, in the last case the rule is persistent, and, again the second part of the condition in line 15 holds, what we have to do in this case is to

Algorithm 5: *ComputeDefesible*

```
Input: A TDL theory $(F, R, \prec)$
$\partial^+ \leftarrow F$
$\partial^- \leftarrow \{(a,[t]) : \sim a^t \in F\}$
$R \leftarrow R \setminus \{r \in R : \sim C(r) \in F\}$
$\prec \leftarrow \prec \setminus \{(r,s),(r,s) : s \in R, \sim C(s) \in F\}$
repeat
    $\partial_+ \leftarrow \emptyset$
    $\partial_- \leftarrow \emptyset$
    $S = \{s \in R : \exists a^{t'} \in A(s) : t' < \min(ptimes(a)), t' \notin ttimes(a)\}$
    $R \leftarrow R \setminus S$
    $persistence(S)$
    for $a \in H_D$ do
        if $R[l] = \emptyset$ then
            $discard(a,[0,\infty])$
        if $\exists r \in R[l] : A(r) = \emptyset$ and $C(r) = a^t$ then
            if $r_{sup} = \emptyset$ then
                $discard(\sim a,[t])$
                if $r \in R_d$ and $R[\sim l] - R_{infd} \subseteq r_{inf}$ then
                    if $r \in R_d^\tau$ then
                        $proved(a,[t])$
                    else if $r \in R_d^\pi[a]$ then
                        $t^* = \min(\{k \in times(\sim a)\} \cup \{\infty\})$
                        $proved(a,[t,t*])$
        else
            $R_{infd} \leftarrow R_{infd} \cup r_{inf}$
until $\partial_+ = \emptyset$ and $\partial_- = \emptyset$
$\partial^+ \leftarrow \partial^+ \cup \partial_+$
$\partial^- \leftarrow \partial^- \cup \partial_-$
```

search for the minimum time greater or equal to t in the rules for $\sim l$, and we can include $(l,[t,t'])$ in ∂^+ and the related housekeeping.

Proposition 6. *Let $D = (\emptyset, R, \succ)$ be a TDL theory such that there is no strict rule with empty body in R; then the transformation ComputeDefeasible(D) is correct.*

3.3 Computing the Extension

To compute the full extension of a TDL theory D we use the following series of transformation:

1. Let D' be the theory obtained from *ComputeDefinite*(D)
2. The defeasible extension is obtained from *ComputeDefeasible*(D').

We call the transformation *ComputeExtension*.

Theorem 1. *ComputeExtension is correct.*

Proof. The result follows from the correctness of the two steps (Propositions 5 and 6).

Theorem 2. *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 $\mathscr{T}_D$ is the set of distinct instants in D.*

Proof. First of all we notice that for each literal a the set $R[a]$ can be implemented as an hash table with pointers to the rules, where each rule is implemented as an ordered list of pairs. The sets of *ptimes* and *ttimes* can be constructed as indexes: the information stored in them can be accessed (in the form required by *ComputeDefeasible*) in linear time.

For *ComputeDefeasible* we have a loop over the set of literals, and inside a loop over the set of rules. After a successful execution of the loop we reduce the complexity of the theory and keep trace of the literals and rules for which we have to repeat the cycle. Every time we remove a rule in the algorithm *ComputeDefeasible*, we call *persistence* to update the extension. For each cycle the number of time we call the procedure is bounded by the number of instants such that there is a particular literal with that instant in the head of a rule. The number of operations we are going to perform is bounded by the number of symbols in the theory (we either remove a rule or a literal from a rule). The total number of symbols in a theory is bounded by the product of the number of rules, the number of atoms and the distinct instant of time in the theory.

Thus, the complexity of *ComputeExtension* is $O(|R| * |H_D| * |\mathscr{T}_D|)$.

4 Discussion and Implementation

TDL is an extension of basic defeasible logic [3] for which [20] proved that the complexity is linear. The extension is twofold: on the syntactic side, literals are labelled with timestamps, on the conceptual side TDL introduces persistent and transient conclusions. The idea of persistent conclusions is that once a conclusion has been classified as persistent then it continues to hold until there are some reasons to terminate it. From a computational point of view, we can propagate persistent conclusions from one instant to the successive instant unless there are some reasons that prevent the propagation. Based on this intuition, if we restrict the language to rules with the form

$$a_1^{t_1}, \dots, a_n^{t_n} \Rightarrow b^t$$

such that

$$\max(\{t_1, \dots, t_n\}) \leq t$$

then we can devise the following procedure [16] to compute the extension of a theory.

- At time 0, consider the sub-theory restricted to the rules whose consequent is labelled by 0. Then use the algorithms given in [20] to compute the extension of the sub-theory at time 0.
- At time $n+1$, consider the extension at time n. Then for each positive conclusion (i.e., conclusion whose proof tag is $+\partial$) p_i:
  - introduce a rule $r^n_{p_i}$: $\Rightarrow^\tau p_i$
  - introduce an instance of the superiority relation $r^n_{p_i} \prec s$ for each s such that $C(s) = {\sim}p_i^{n+1}$;
  - remove p_i^n from the body of rules where it occurs;

For each negative conclusion q_j remove rules where q_j appears in the body. Compute the extension for the sub-theory restricted to the rules whose consequent is labelled with $n+1$.

Thus, we compute the extension of a theory D in the interval $[0,t]$ in $O(|D|*t)$-time (where $|D|$ is the number of instances of literals in it).

The above procedure applies an incremental swap over the time-line, and the problem with it is that it cannot look backward. Therefore the major limitation is that it cannot handle rules where the time of the conclusion precedes the time of some of its antecedents. This is a drawback, since rules of that form are able to capture interesting phenomena, such as backward causation and normative retroactivity. While it is debatable whether effects can precede the causes (a discussion is reported in [16]), normative retroactivity occurs indeed in law. 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 happens, e.g., of taxation law, where it is possible to claim a tax benefit from a date in the past. Theories containing rules such as $\{a^{10} \Rightarrow^{\tau} b^{10}, b^{10} \Rightarrow^{\pi} c^{0}, c^{3} \Rightarrow^{\pi} a^{0}\}$ raise computational problems: clearly, it is not possible to handle the computation by sequentially running the algorithms of [20] for each time-slice. The procedure proposed in this paper handles these cases by keeping the complexity under control. An efficient Java implementation of TDL based on the algorithms presented in this paper is discussed in [24, 23]. Initial experiments with the implementation confirm the linearity (and scalability) of the approach. The implementation has been tested on both synthetic theories (designed to test particular features of TDL) and concrete theories obtained from real life scenarios.

On the synthetic side, TDL and the proposed algorithms can account for compact encodings and efficient computations. For example consider the theories:

$$T_1\colon \Rightarrow^{\pi} p^{0} \qquad\qquad T_2\colon \Rightarrow^{\tau} p^{0}, p^{i} \Rightarrow^{\tau} p^{i+1} \text{ for } 0 \leq i \leq 99$$

The two theories are equivalent (i.e., they generate the same extension) in the time interval $[0,100]$, but, trivially, our algorithms compute the extension much more quickly when given the first theory as input than when the second theory is used as input. On the practical side, of particular interest is the formalisation in TDL of the Road Traffic Restriction Regulation of the Italian town of Piacenza [23, 14].

5 Related Work

Typically there are two mainstream approaches to reasoning with and about time. A point based approach, as in the present paper, and an interval based approach [1]. Notice that the current approach 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. While TDL is able to cover many different aspects (in particular retroactivity), its time complexity is linear: to the best of our knowledge, no logic with the same coverage of TDL is so efficient.

Anyway, we would like to point out that interval and duration based defeasible reasoning has been developed by [5, 16]. [16] focus on duration and periodicity and relationships with various forms of causality: no complexity result was presented there. [5] integrates temporal reasoning in argumentation theory and proposes a sophisticated interaction of defeasible reasoning and standard temporal reasoning (i.e., mutual relationships of intervals and constraints on the combination of intervals). In [5] constraint-based temporal reasoning combining intervals and instants are integrated in an argumentation framework. Predicates as *Holds*, *Occurs* and *Do* are used to associate constrained temporal information to logic formulas representing properties, events and actions. From the general point of view, [5] develops a very expressive (but not necessarily tractable) argumentation system: indeed, no complexity results are available for this work, but the system deals with very complex temporal structures without any apparent computational concern.

Other interesting approaches integrating temporal reasoning in argumentation theory include Mann and Hunter's [21] and Barringer and Gabbay's [6]. The last one proposes temporal and modal languages to represent arguments in the nodes of a network and give a Kripke semantics. Mann and Hunter in [21] encode temporal information via formulas of the form Holds(α, i) to express that α holds at interval i and propose a translation into classical propositional calculus.

6 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.

A previous version of this work was presented at NMR'2010 [13]. We are grateful to the referees for their valuable comments on earlier versions.

References

1. J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.
2. G. Antoniou and A. Bikakis. Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Trans. Knowl. Data Eng.*, 19(2):233–245, 2007.
3. G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2:255–287, 2001.
4. 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.
5. J. Augusto and G. Simari. Temporal defeasible reasoning. *Knowledge and Information Systems*, 3:287–318, 2001.
6. H. Barringer and D. Gabbay. Time for verification. chapter Modal and temporal argumentation networks, pages 1–25. Springer, 2010.
7. 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.

8. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Australian AI*, volume 4830 of *LNCS*. Springer, 2007.
9. G. Governatori, V. Padmanabhan, and A. Rotolo. Rule-based agents in temporalised defeasible logic. In *PRICAI 2006*, number 4099 in LNAI, pages 31–40, Berlin, 2006. Springer.
10. G. Governatori, M. Palmirani, R. Riveret, A. Rotolo, and G. Sartor. Norm modifications in defeasible logic. In *JURIX 2005*, pages 13–22. IOS Press, Amsterdam, 2005.
11. G. Governatori and A. Rotolo. A computational framework for institutional agency. *Artif. Intell. Law*, 16(1):25–52, 2008.
12. G. Governatori and A. Rotolo. Changing legal systems: Legal abrogations and annulments in defeasible logic. *Logic Journal of IGPL*, 18(1):157–194, 2010.
13. G. Governatori and A. Rotolo. On the complexity of temporal defeasible logic. In *NMR 2010*, 2010.
14. G. Governatori, A. Rotolo, and R. Rubino. Implementing temporal defeasible logic for modeling legal reasoning. In *JSAI-isAI 2009 Workshop*. Springer, 2010.
15. G. Governatori, A. Rotolo, and G. Sartor. Temporalised normative positions in defeasible logic. In *ICAIL'05*, pages 25–34. ACM Press, 2005.
16. G. Governatori and P. Terenziani. Temporal extensions to defeasible logic. In *Australian AI 2007*, pages 476–485. Springer, 2007.
17. H. Lam and G. Governatori. The making of SPINdle. In *RuleML 2009*. Springer, 2009.
18. H.-P. Lam and G. Governatori. What are the necessity rules in defeasible reasoning? In *LPNMR 2011*, pages 187–192. Springer, 2011.
19. M. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10(4):483–501, 2001.
20. M. J. Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1:691–711, 2001.
21. N. Mann and A. Hunter. Argumentation using temporal knowledge. In *COMMA 2008*, pages 204–215. IOS Press, 2008.
22. D. Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 353–395. OUP, 1993.
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 *RuleML 2009*, pages 298–305. Springer, 2009.
25. M. Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, 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.