A Solution to the Ramification Problem Expressed in Temporal Description Logics

Nikos Papadakis[1], Polydoros Petrakis[1], Dimitris Plexousakis[2],Haridimos Kondylakis[2]

1: Science Department, Technological Educational Institute of Crete
2: Computer Science Department University of Crete & Institute of Computer Science, FORTH,Crete
{ npapadak,ppetrak,dp,kondylak)@csd.uoc.gr

Abstract. The ramification problem in Artificial Intelligence is concerned with the indirect effects of an action. It has been shown in previous work that the ramification problem can be solved with the use of integrity constraints and actions representation. In this paper we begin with a quick review of the existing Description Logics Languages, and then we describe a Temporal Extension of Description Logics, able to represent integrity constraints, temporalized actions and non persistent effects. We describe a thorough solution to the ramification problem in Temporal Settings expressed in Temporal Description Logics.

Keywords: Ramification problem, Temporal Description Logics, Temporalized Actions

1 Introduction

The ramification problem, in Artificial Intelligence field, is concerned with the indirect effects of an action. In other words, it deals with the problem of representing the consequences of an action. It is a hard and ever existing problem, in systems exhibiting a dynamic behavior [11].

We describe a solution to the ramification problem, by using an example originally presented by [11]. However, the example and solution are this time expressed in whole, in Temporal Description Logics instead of First Order Logic. In order to accomplish that, we put together features of existing Description Logics languages and enrich them to become more specific and deterministic, so as to describe the integrity constraints and temporalized actions, as well as the effects of those actions, with uttermost clarity. More specifically, we combine features from the language $\mathcal{TL}$-$\mathcal{F}$ presented by Artale and Franconi [3], along with features of the syntax rules from the Schmiedel proposal [2] (also shown in figure 4), in order to be able to represent actions and integrity constraints, in Interval Based Temporal Description Logics. We are also able to represent the non persistent effects of those actions. The effects refer to specific and well defined time intervals.

In the following section (Section 2), we are going to present the basic syntax of Description Logics, along with existing implementations of Description Logic Languages and some Temporal Extensions of Descriptions Logics. In Section 3, we present a solution to the ramification problem with the use of an example and Temporal Description Logics. We provide algorithms for the production of static rules and the evaluation of dynamic and static rules. In the last part of Section 3, there are two theorems proving the correctness of the previously presented algorithms. In Section 4, we summarize the information provided in this paper, and describe further extensions or applications of the provided solution of the ramification problem.

2 Description Logics Basics and Previous Work

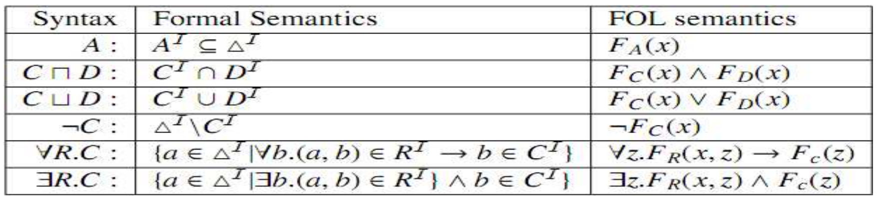
In this section, we will initially present cases where it is useful to migrate from First Order Logic to Description Logics. Then we will describe the basics of Description Logics and mention existing Description Logic Languages, as well as Temporal Extensions to Description Logics.

First Order Logics is nowadays the most common way used for knowledge representation. However using FOL (First Order Logic) to represent Knowledge Bases in Relational Databases has proved to be inefficient, as FOL has too much expressive power and therefore lacks computational speed and efficient procedures. Also, Semantic Networks and frames do not require the whole part of First Order Logic. Additionally, the direct use of FOL can have too low inference power for expressing interesting theories. Therefore "Description Logics" has been introduced as a formal language for representing knowledge and reasoning about it [7] Description Logics, a structured fragment of FOL, is nowadays considered the most important knowledge representation formalism, unifying and giving a logical basis to Frame-based systems, Semantic Networks, Object Oriented and semantic data models. Description Logics are preferred for their high expressivity and decidability, as well as their reasoning algorithms, which always return correct answers [3]. It is shown by [8], that Knowledge Base satisfiability in Description Logics (specifically with language $\mathcal{ALCQI}$) can be EXPTIME-decidable and EXPTIME-complete.

The basic types of a concept language are concepts, roles, and individuals (concept names) [5]. A concept is a description of a collection of individuals with common properties [13], for example “Employee” is a concept. Roles express relations between these individuals of concepts. For example “supervise”, represents the relationship between individuals belonging in Employee and Employer concepts. Individuals are constants of concepts e.g. “Nick” is a constant of Employee (Nick: Employee). The most basic Description Logic Language is $\mathcal{ALC}$ (Attributive Language with Complements) with syntax:

$$\mathcal{ALC} ::= \ \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists\, R.C \mid \forall\, R.C \ .$$

The following table represents the formal semantics of $\mathcal{ALC}$ and their FOL representation [13]..



Syntax	Formal Semantics	FOL semantics
A :	$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}$	$F_A(x)$
$C \sqcap D$:	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$F_C(x) \wedge F_D(x)$
$C \sqcup D$:	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$F_C(x) \vee F_D(x)$
$\neg C$:	$\triangle^{\mathcal{I}} \backslash C^{\mathcal{I}}$	$\neg F_C(x)$
$\forall R.C$:	$\{a \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	$\forall z.F_R(x,z) \rightarrow F_c(z)$
$\exists R.C$:	$\{a \in \triangle^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\} \wedge b \in C^{\mathcal{I}}\}$	$\exists z.F_R(x,z) \wedge F_c(z)$

Fig. 1. Semantics of $\mathcal{ALC}$.

Let us note that an interpretation **I** (shown in Figure 1), is a model of a knowledge base Σ, if and only if every axiom of Σ, is satisfied by **I**. Σ logically implies A ⊑ C (written Σ ⊨ A ⊑ C) if $A^I \subseteq C^I$ for every model of Σ: we say that A is subsumed by C in Σ [3].

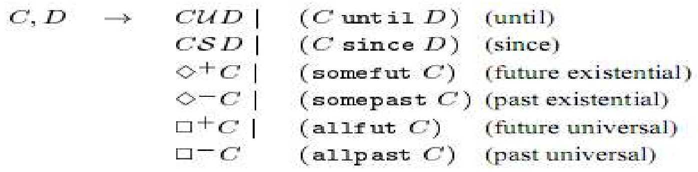
Description logics can be extended in order to have a temporal basis. We can have either point-based temporal description logics ($\mathcal{ALCT}$ [13], $\mathcal{CQIus}$ [12] , $\mathcal{ALCQIT}$ [4], or interval based temporal description logics [13]. A way of adding tense logic (point based) to Description Logics was introduced by [12], with $\mathcal{CQIus}$, and extended by [4], with $\mathcal{ALCQIT}$. In the aforementioned extensions the temporal operators $\mathcal{U}$(until) and $\mathcal{S}$ (Since), along with the temporal operators $\diamond^{+}$, $\diamond^{-}$, $\square^{+}$, $\square^{-}$ were introduced, in order to enhance Description Logics with a temporal dimension. The meaning of those operators (also shown in Figure 2) is:

$\diamond^{+}$ (Sometime in the future) $\diamond^{+} C = \top \mathcal{U} C$

$\diamond^{-}$ (Sometime in the past) $\diamond^{-} C = \top \mathcal{S} C$

$\square^{+}$ (Always in the future)

$\square^{-}$ (Always in the past)



C, D	$\rightarrow$	$C \mathcal{U} D$ |	$(C$ **until** $D)$	(until)
		$C \mathcal{S} D$ |	$(C$ **since** $D)$	(since)
		$\diamond^{+} C$ |	(**somefut** C)	(future existential)
		$\diamond^{-} C$ |	(**somepast** C)	(past existential)
		$\square^{+} C$ |	(**allfut** C)	(future universal)
		$\square^{-} C$	(**allpast** C)	(past universal)

Fig. 2. Temporal Extension of $\mathcal{ALCQI}$

A further extension of the languages $\mathcal{ALCT}$, $\mathcal{ALCQIT}$ and $\mathcal{CQIus}$ was made by [9], with the language $\mathcal{ALC}_{\mu \bullet \circ}$ that defines the new operators ●(next time), ○(previous time), and fixpoint concept expressions like $\mu A.C$. A language describing Interval Based Description Logics is $\mathcal{TDL}$ presented by Lutz [10]. Also a powerful temporal description language named $\mathcal{DLRus}$ is described by [6]. $\mathcal{DLRus}$ variants accomplish EXPTIME-complete reasoning, and EXPSPACE-complete satisfiability and logical implication [6].

Artale and Franconi have also presented a language able to describe both non-temporal feature logic and interval temporal networks is $\mathcal{TL}$-$\mathcal{F}$ [3]. The basic types of this language are concepts, individuals, temporal variables and intervals. Concepts can be specified to hold at a certain temporal variable (or interval). In this way, actions (resp. individual actions) can be expressed in a uniform way by temporally related concepts (resp. individuals) [3]. The most common notation used for Temporal Interval Relations, used in Interval Based Temporal Description Logics, is the one originally presented by Allen in [1],[5], and also shown in Figure 3.

Concepts expressions are denoted by C, D are built out of atomic concepts denoted by A. Atomic features are denoted by f, whereas atomic parametric features are denoted by ⋆g. Parametric features [3], (e.g. ⋆Employee in ⋆Employee: Misdemeanor) plays the role of formal parameter of the action mapping any individual action of type Misdemeanor independently from time. Temporal variables are denoted by X, Y. When writing C@X it means that concept C holds at time interval denoted by variable X. There is also the special temporal variable **now (or #)** which is used as the reference temporal variable of the action (concept). Now variable can be omitted when possible. For example, the expression Illegal ⊑ Suspended is the same with Illegal@now ⊑ Suspended@now.

Relation	Abbr.	Inverse
before(i, j)	b	a
meets(i, j)	m	mi
overlaps(i, j)	o	oi
starts(i, j)	s	si
during(i, j)	d	di
finishes(i, j)	f	fi

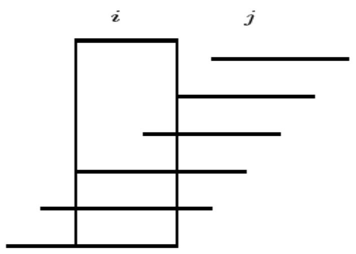


Fig. 3. Allen's interval relationships.

```
<concept> ::= <atomic-concept>
            | (and <concept>+)
            | (all <role> <concept>)
            | (atleast min <role>)
            | (atmost max <role>)
            | (at <interval> <concept>)
            | (sometime (<interval-variable>+) <time-net>.<concept>)
            | (alltime (<interval-variable>+) <time-net>.<concept>)
<atomic-concept> ::= symbol
<role> ::= <atomic-role>
            | (and <role>+)
            | (domain <concept>)
            | (range <concept>)
            | (at <interval> <role>)
            | (sometime (<interval-variable>+) <time-net>.<role>)
            | (alltime (<interval-variable>+) <time-net>.<role>)
<atomic-role> ::= symbol
<time-net> ::= <time-constraint>
            | (and <time-constraint>+)
<time-constraint> ::= (<interval-relation> <interval> <interval>)
            | (<comparison> <interval> <duration-constant>)
            | (<granularity> <interval>)
<interval-relation> ::= equal | meets | met-by | after | before
            | overlaps | overlapped-by | starts | started-by
            | finishes | finished-by | during | contains
            | (or <interval-relation>+)
<comparison> ::= < | ≤ | = | ≥ | >
<granularity> ::= sec | min | hour | . . .
<interval> ::= <interval-variable> | <interval-constant> | now
<interval-variable> ::= symbol
<interval-constant> ::= symbol
<duration-constant> ::= symbol
```

Fig. 4. Syntax rules of Schmiedel's Temporal Extension proposal

As we have mentioned before in the following section we combine features from the language $\mathcal{TL}$-$\mathcal{F}$ presented by [3], as well as features of the syntax rules from the Schmiedel proposal [2] (also shown in figure 4), in order to represent actions, integrity constraints and non persistent effects, in Interval Based Temporal Description Logics. Extensions have been made, in cases where the existing Temporal Description Logic Languages, were not specific enough to express actions, integrity constraints and time restrictions, with clarity as shown in the following example of Section 3.

3 Dealing with the ramification problem with algorithms expressed in Temporal Description Logics

A solution to the ramification problem with the use of Temporal Description Logics is presented in this section. Consider the following example from [11]): If a public employee commits a misdemeanor, then for the next five months s/he is considered illegal, except if s/he receives a pardon. When a public employee is illegal, then s/he must be suspended and cannot take promotion, for the entire time interval over which s/he is considered illegal. Also when a public employee is suspended s/he cannot take her/his salary until s/he stops being suspended. Each public employee is evaluated for her/his work. If s/he receives a bad grade, then s/he is assumed to be a bad employee and s/he cannot take promotion until s/he receives a good grade. If s/he receives a good grade, then s/he is assumed to be a good employee and s/he takes a bonus if s/he is not suspended. Also, assume that a public worker is not illegal, if there does not exist information that proves s/he is illegal, and is not suspended if there does not exist information that proves s/he is suspended and takes his/her salary, if there does not exist information that proves the opposite. This helps us define the default axioms. As we observe we have four actions *misdemeanor*, *take_pardon*, *good_grade*, *bad-grade* and seven fluents: good*_employee*, *bad_employee*, *illegal*, *take_salary*, *take_bonus*, *take_promotion*, *suspended*. The direct effects of the four actions are expressed in propositional form by the following constraints:

FOL Representation of the Integrity Constraints as presented by Papadakis and Plexousakis [11]:

occur(misdemeanor(p), t) ⊃ illegal(p,5m) **(1)**
occur(take_pardon(p), t) ⊃ ¬illegal(p, ∞) **(2)**
occur(bad_grade(p), t) ⊃ ¬good_employee(p, ∞) **(3)**
occur(good_grade(p),t) ⊃ good_employee(p, ∞) **(4)**

t is a temporal variable and occur(misdemeanor(p), t) means that the action misdemeanor(t) takes place at time t. The following integrity constraints describe the indirect effects of the four aforementioned actions.

illegal(p, t1) ⊃ suspended(p, t1) **(5)**
illegal(p,t1) ⊃ ¬ take_promotion(p, t1) **(6)**
suspended(p, t1) ⊃ ¬ take_salary(p, t1) **(7)**
¬ good_employee(p, t1) ⊃ ¬ take_promotion(p, t1) **(8)**
¬ suspended(p, t1) ∧ good_employee(p, t2) ⊃ take_bonus(p, min(t1,t2)) **(9)**
¬ good_employee(p,t1) ⊃ ¬ take_bonus(p, t1) **(10)**
¬ suspended(p, t1) ⊃ take_salary(p, t1) **(11)**

Temporal Description Logics Representation of the Integrity Constaints:

$\diamond^{+}$ (x) (**starts now** x) (month x) (= x 5)(⋆Employee: Misdemeanor ⊑ ⋆Employee: Illegal@x) **(1)**
$\diamond^{+}$ (x) (**starts now** x) (⋆Employee: Take_pardon ⊑ ⋆Employee: ¬ Illegal@x) **(2)**

$\Diamond^+$ (x) (**starts now** x) ($\star$Employee: Bad_grade $\sqsubseteq$ $\star$Employee:$\neg$Good_employee@x) **(3)**

$\Diamond^+$ (x) (**starts now** x) ($\star$Employee: Good_grade $\sqsubseteq$ $\star$Employee: Good_employee@x) **(4)**

Also we have got the following **integrity constraints** describing the **indirect effects** of the aforementioned actions.

$\star$Employee:Illegal $\sqsubseteq$ $\star$Employee:Suspended **(5)**

$\star$Employee:Illegal $\sqsubseteq$ $\star$Employee: $\neg$Take_promotion **(6)**

$\star$Employee:Suspended $\sqsubseteq$ $\star$Employee: $\neg$Take_salary **(7)**

$\star$Employee: $\neg$ Good_employee $\sqsubseteq$ $\star$Employee: $\neg$Take_promotion **(8)**

$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z min(x, y)) ($\star$Employee: $\neg$Suspended@x $\sqcap$ $\star$Employee: Good_employee@y $\sqsubseteq$ $\star$Employee: Take_bonus@z) **(9)**

$\star$Employee:$\neg$Good_employee $\sqsubseteq$ $\star$Employee:$\neg$Take_bonus **(10)**

$\star$Employee:$\neg$Suspended $\sqsubseteq$ $\star$Employee: Take_salary **(11)**

In Temporal Settings we need to describe the direct and indirect effects of an action, not only in the immediately resulting next situation, but possibly for many future situations. In the above example, the action Misdemeanor has the indirect effect that the public worker is in suspension in the next five months. In these five months the action Good_grade may occur, but even if that happens, the employee still cannot take promotion. This means that the world changes situations while the direct and indirect effects of some actions still hold. In the above example the dynamic axioms are the (1) – (4), while the static axioms are the rest (5) – (11). We have the following **default axioms**:

$\Diamond^+$ (x y) (**starts** now x)(**starts** now y) (= x 0) ($\star$Employee: Illegal@x $\sqcap$ $\star$Employee:$\neg$Illegal@x $\sqsubseteq$ $\star$Employee: $\neg$Illegal@y)

$\Diamond^+$ (x y) (**starts** now x)(**starts** now y) (= x 0) ($\star$Employee: Take_salary@x $\sqcap$ $\star$Employee: $\neg$Take_salary@x $\sqsubseteq$ $\star$Employee: Take_salary@y)

$\Diamond^+$ (x y) (**starts** now x)(**starts** now y) (= x 0) ($\star$Employee: Suspended@x $\sqcap$ $\star$Employee: $\neg$Suspended@x $\sqsubseteq$ $\star$Employee: $\neg$Suspended@y)

a. Algorithms for the Production of Static Rules

The static rules can be used to deduct the indirect effects of the execution of each action. The indirect effects exist due to the presence of integrity constraints. Therefore, it is possible to produce the static rules, from the integrity constraints. The following algorithm describes the steps needed for the aforementioned production.

1. Transform each integrity constraint in its CNF (conjunctive) form. Now each integrity constraint has the form $C_1 \sqcap C_2 \sqcap C_3 \ldots \sqcap C_n$
2. For each i from 1 to n do

```
Assume $C_i = F_1 \sqcup F_2 \dots \sqcup F_m$
For each j from 1 to m do
        For each k from 1 to m and $k \neq j$ do
                if $(F_j, F_k) \in I$ then
                $R = R \sqcup (\neg F_j$ causes $F_k$ if $\sqcap F_l)$, $l=1,\dots,m$, $l \neq j, k$
```

3. For each fluent F_k the rules have the following form:

   $\sqcap F_i$ causes F_k if Φ

   $\sqcap F_i'$ causes $\neg F_k$ if Φ'

   We change the static rules from the form:

   $G \sqsubseteq F_k$

   $K \sqsubseteq \neg F_k$

   to the form:

   $G' \sqsubseteq F_k$

   $K' \sqsubseteq \neg F_k$

   where

   $G' = G \sqcup (\sqcap F_i \sqcap \Phi)$

   $K' = K \sqcup (\sqcap F_i' \sqcap \Phi')$
4. We replace each rule $G_p \sqsubseteq F_p$ with $\Diamond^+$ (x)(**starts** now x)($G_p@x \sqsubseteq F_p@x$)

Now we apply the algorithm to the previous example. First we have to produce the set *I*. In order to do that, we make use of an algorithm presented by Papadakis and Plexousakis [11]. The algorithm is however described with Description Logics this time.

1. For each fluent $F \in G_f$, $F' \in K_f$, where $G_f \sqsubseteq K_f$ is a specified integrity constraint add the pair $(F, F') \in I$.
2. For each $F \in G_f$, $F' \in K_f$, where $G_f \equiv K_f$ is a specified constraint do
   - If F can change its truth values as the direct effect of an action, then add (F,F') in *I*. If F' can change its truth value as a direct effect of an action then add (F',F) in *I*.

All the integrity constraints (IC) have the form $A \sqsubseteq B$. We have that:
(Illegal, Suspended) $\in$ I (from IC 5)
(Illegal, $\neg$Take_promotion) $\in$ I (from IC 6)
(Suspended, $\neg$Take_salary) $\in$ I (from IC 7)
($\neg$Good_employee, $\neg$Take_promotion) $\in$ I (from IC 8)
($\neg$Suspended, Take_bonus) $\in$ I (from IC 9)
(Good_employee, Take_bonus) $\in$ I (from IC 9)
($\neg$Good_employee, $\neg$ Take_bonus) $\in$ I (from IC 10)
($\neg$Suspended, Take_salary) $\in$ I (from IC 11)

The transformation of integrity constraints in conjunctive normal form (step 1) yields:

$\star$Employee: $\neg$Illegal $\sqcup$ $\star$Employee: Suspended **(5)**

⋆Employee: ¬Illegal ⊔ ⋆Employee: ¬Take_promotion **(6)**
⋆Employee: ¬Suspended ⊔ ⋆Employee: ¬Take_salary **(7)**
⋆Employee: Good_employee ⊔ ⋆Employee: ¬ Take_promotion **(8)**
$\Diamond^{+}$ (x y z) (**starts** now x)(**starts** now y)(= z min(x,y)) ⋆Employee: Suspended@x ⊔ ⋆Employee: ¬Good_employee@y ⊔ ⋆Employee: Take_bonus@z **(9)**
⋆Employee: Good_employee ⊔ ⋆Employee: ¬Take_bonus **(10)**
⋆Employee: Suspended ⊔ ⋆Employee: Take_salary **(11)**

In step 2 we estimate all causal relationships omitting the atomic parametric feature "⋆Employee" for better readability.

R = {Illegal *causes* Suspended if ⊤, Illegal *causes* ¬Take_promotion if ⊤,
Suspended *causes* ¬Take_salary if ⊤, ¬Good_employee *causes* ¬Take_promotion if ⊤,
Good_employee *causes* Take_bonus if ¬Suspended,
¬Suspended causes Take_bonus if Good_employee,
¬Good_employee *causes* ¬Take_bonus if ⊤, ¬Suspended *causes* Take_salary if ⊤}.

In the following step (step 3), we construct the fluent formulas, which make each fluent true. In case that there are more than one causal relationships affecting the same fluent, we integrate them in this step. Take for example the second and fourth causal relationships from step 2.

R = {Illegal ⊑ Suspended,
Illegal ⊔ ¬Good_employee ⊑ ¬Take_promotion,
Suspended ⊑ ¬Take_salary,
¬Suspended ⊓ Good_employee ⊑ Take_bonus,
¬Good_employee ⊑ ¬Take_bonus,
¬Suspended ⊑ Take_salary }.

Finally in step 4 (which must be exectued, at each time point, at which the static rules are evaluated) we have the following six static rules:

R = {$\Diamond^{+}$ (x)(**starts** now x)(⋆Employee: Illegal@x ⊑ ⋆Employee: Suspended@x),
$\Diamond^{+}$ (x y z)(**starts** now x)(**starts** now y)(= z max(x,y))(⋆Employee:Illegal@x ⊔ ⋆Employee: ¬Good_employee@y ⊑ ⋆Employee: ¬Take_promotion@z),
$\Diamond^{+}$ (x)(**starts** now x)(⋆Employee: Suspended@x ⊑ ⋆Employee: ¬Take_salary@x),
$\Diamond^{+}$ (x y z)(**starts** now x)(**starts** now y)(= z min(x, y)) (⋆Employee: ¬Suspended@x ⊓ ⋆Employee: Good_employee@y ⊑ ⋆Employee: Take_bonus@z),
$\Diamond^{+}$ (x)(**starts** now x)(⋆Employee: ¬ Good_employee@x ⊑ ⋆Employee: ¬Take_bonus@x),
$\Diamond^{+}$ (x)(**starts** now x)(⋆Employee: ¬Suspended@x ⊑ ⋆Employee: Take_salary@x) }.

In step 4, for each static rule, we find the maximum time that its body is true. For example in the second rule, the body is true when Illegal@x ⊔ ¬Good_employee@y is true. This means that we take the maximum of times x, y for which ¬Take_promotion is true. On the fourth rule, the body is true when ¬Suspended@x ⊓ Good_employee@y is true. This means that we must take the minimum of times x, y.

b. Algorithms for the Evaluation of Dynamic and Static Rules

In this section we present an algorithm for the evaluation of rules:

1. After the execution of one action evaluate the dynamic rule which references at this action.
2. Each time moment do:
    a. Evaluate the default axioms
    b. Repeat until no change occurs
        i. Evaluate all static rules
        ii. If a fluent $\Diamond^+$ (a x)(**starts** a x)(F@x) becomes true after the evaluation of a static rule, then set $\Diamond^+$ (a y)(**starts** a y)(= y 0)(¬F@y) (the negation is false), for the same time interval a as in the previous rule.

Consider the above example with the public worker. We have four dynamic rules (1-4) as we have described in the previous section. Also we have produced the **six static rules**:

$\Diamond^+$ (x)(**starts now** x)(⋆Employee: Illegal@x ⊑ ⋆Employee: Suspended@x),
$\Diamond^+$ (x y)(**starts now** x)(**starts now** y)(= z max(x,y))(⋆Employee: Illegal@x ⊔ ⋆Employee: ¬Good_employee@y ⊑ ⋆Employee: ¬Take_promotion@z),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: Suspended@x ⊑ ⋆Employee: ¬Take_salary@x),
$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z min(x, y)) (⋆Employee: ¬Suspended@x ⊓ ⋆Employee: Good_employee@y ⊑ ⋆Employee: Take_bonus@z),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬ Good_employee@x ⊑ ⋆Employee: ¬Take_bonus@x),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬Suspended@x ⊑ ⋆Employee: Take_salary@x)

Assume now that we have a public worker (employee) ⋆E, and the initial situation is:

S_0 = {$\Diamond^+$ (x) (starts now x)(⋆E :¬Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: ¬Good_employee@x, ⋆E: ¬Illegal@x)}

Time starts at 0 and has the granularity of month. Assume that the following actions occur at the following time points:

⋆E: Good_grade@2
⋆E: Misdemeanor@4
⋆E: Bad_grade@6

⋆E: Good_grade@8
⋆E: Misdemeanor@10
⋆E: Take_pardon@12

At time point 2 the action Good_grade executes. According to the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule (4) we have the following situation:

S'$_1$ = {$\Diamond^+$ (x) (starts now x) (⋆E:¬Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: Good_employee@x, ⋆E: ¬Illegal@x)}

As we observe the following static rule will be evaluated:

$\Diamond^+$ (x y z)(starts now x)(starts now y)(starts now z)(= z min(x,y))(⋆E: ¬Suspended@x ⊓ ⋆E: Good_employee@y ⊑ ⋆E: Take_bonus@z)

In the previous rule the (starts now z) could be omitted, as z can be either x or y. Before the action execution, time interval x has the maximum value [now, ∞). Therefore it is safe to assume that the minimum time interval among x and y is y, as the time interval updated in this rule is y from fluent Good_employee. Therefore in the above rule z=y.

S$_1$ = {$\Diamond^+$ (x) (starts now x) (⋆E: Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: Good_employee@x, ⋆E: ¬Illegal@x)}

The situation does not change until timepoint 4, when the second action (Misdemeanor) takes place and causes Illegal to become true from the next 5 time units. From the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule(1) we have the situation:
S$_2$' = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 5) (⋆E :Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: Good_employee@x, ⋆E: Illegal@y)}

As we observe the following static rules will be evaluated:

$\Diamond^+$ (x)(**starts now** x)(⋆Employee:Illegal@x ⊑ ⋆Employee:Suspended@x),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee:Suspended@x ⊑ ⋆Employee: ¬Take_salary@x).

x has duration 5 so we get:

$\Diamond^+$ (x)(**starts now** x)(= x 5)(⋆Employee:Illegal@x ⊑ ⋆Employee:Suspended@x),
$\Diamond^+$ (x)(**starts now** x)(= x 5)(⋆Employee:Suspended@x ⊑ ⋆Employee: ¬Take_salary@x),

After the evaluation of the static rules we have the situation:

S_2 = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 5) (⋆E :Take_bonus@x, ⋆E: ¬Take_salary@y, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@y, ⋆E: Good_employee@x, ⋆E: Illegal@y)}

This situation does not change until the time point 6, when the third action (Bad_grade) executes. From the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule (3) we have the situation:

S_3' = {$\Diamond^+$ (x y) (starts now x) (starts now y)(= y 3) (⋆E :Take_bonus@x, ⋆E: ¬Take_salary@y, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@y, ⋆E:¬Good_employee@x, ⋆E: Illegal@y)}

The following static rule:

$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬Good_employee@x ⊑ ⋆Employee: ¬Take_bonus@x), will be evaluated and the situation will become:

S_3 = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 3) (⋆E:¬Take_bonus@x, ⋆E:¬Take_salary@y, ⋆E:¬Take_promotion@x, ⋆E: Suspended@y, ⋆E: ¬Good_employee@x, ⋆E: Illegal@y)}

This situation does not change until time point 8, when the fourth action (Good_Grade) takes place. From the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule (4), we have the situation:

S_4' = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 1) (⋆E : ¬Take_bonus@x, ⋆E:¬Take_salary@y, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@y, ⋆E: Good_employee@x, ⋆E: Illegal@y)}

No static rule is executed. Therefore the situation does not change. At time point 9 no action takes place, but the situation changes because the following default axioms hold:
$\Diamond^+$ (x y) (**starts** now x)(**starts** now y) (= x 0) (⋆Employee: Illegal@x ⊓ ⋆Employee:¬Illegal@x ⊑ ⋆Employee: ¬Illegal@y)

$\Diamond^+$ (x y) (**starts** now x)(**starts** now y)(= x 0) (⋆Employee: Take_salary@x ⊓ ⋆Employee: ¬Take_salary@x ⊑ ⋆Employee: Take_salary@y)

$\Diamond^+$ (x y) (**starts** now x)(**starts** now y) (= x 0) (⋆Employee: Suspended@x ⊓ ⋆Employee: ¬Suspended@x ⊑ ⋆Employee: ¬Suspended@y)

The situation is:

S_5' = {$\Diamond^+$ (x) (starts now x) (⋆E: ¬Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E:¬Suspended@x, ⋆E: Good_employee@x, ⋆E:¬Illegal@x)}

Now the static rule:

$\Diamond^+$ (x y z)(**starts** now x)(**starts** now y)(= z min(x, y)) (⋆Employee:¬Suspended@x ⊓ ⋆Employee: Good_employee@y ⊑ ⋆Employee: Take_bonus@z), is executed with x=y and therefore z=x=y. After the evaluation of the rule we have:

S_5 = {$\Diamond^+$ (x) (starts now x) (⋆E: Take_bonus@x, ⋆E:Take_salary@x, ⋆E:¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: Good_employee@x, ⋆E:¬Illegal@x)}

At time point 10 the action Misdemeanor executes, resulting the situation:

S_6’ = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 5)(⋆E : Take_bonus@x, ⋆E:Take_salary@x, ⋆E:¬Take_promotion@x, ⋆E:¬Suspended@x, ⋆E: Good_employee@x, ⋆E: Illegal@y)}

$\Diamond^+$ (x)(**starts** now x)(= x 5)(⋆Employee:Illegal@x ⊑ ⋆Employee:Suspended@x),
$\Diamond^+$ (x)(**starts** now x) (= x 5)(⋆Employee:Suspended@x ⊑ ⋆Employee: ¬Take_salary@x).

S_6 = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 5)(⋆E: Take_bonus@x, ⋆E: ¬Take_salary@y, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@y, ⋆E: Good_employee@x, ⋆E: Illegal@y)}

The last action (Take_pardon) occurs at time point 12. The new situation is

S_7’ = {$\Diamond^+$ (x y) (starts now x)(starts now y)(= y 3)(⋆E : Take_bonus@x, ⋆E: ¬Take_salary@y, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@y, ⋆E: Good_employee@x, ⋆E:¬Illegal@x)}

Finally the situation changes again at time point 15, because the following default axioms hold:

$\Diamond^+$ (x y) (**starts now** x)(**starts now** y)(= x 0) (⋆Employee: Take_salary@x ⊓ ⋆Employee: ¬Take_salary@x ⊑ ⋆Employee: Take_salary@y)

$\Diamond^+$ (x y) (**starts now** x)(**starts now** y) (= x 0) (⋆Employee: Suspended@x ⊓ ⋆Employee: ¬Suspended@x ⊑ ⋆Employee: ¬Suspended@y)

Now the situation is:

S_8 = {$\Diamond^+$(x) (starts now x) (⋆E : Take_bonus@x, ⋆E: Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: Good_employee@x, ⋆E: ¬Illegal@x)}

This is the end of execution. As we observe from the set R, for each pair (F, ¬F) it holds that $G_F \sqcap K_F \equiv$ FALSE, when $G_F \sqsubseteq F$ and $K_F \sqsubseteq \neg F$. More specifically the set of static rules is:

R = {$\Diamond^+$ (x)(**starts now** x)(⋆Employee: Illegal@x ⊑ ⋆Employee: Suspended@x),
$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z max(x,y))(⋆Employee:Illegal@x ⊔ ⋆Employee: ¬Good_employee@y ⊑ ⋆Employee: ¬Take_promotion@z),

$\Diamond^+$ (x)(**starts now** x)(⋆Employee: Suspended@x ⊑ ⋆Employee: ¬Take_salary@x),
$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z min(x, y)) (⋆Employee:¬Suspended@x ⊓ ⋆Employee: Good_employee@y ⊑ ⋆Employee: Take_bonus@z),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬ Good_employee@x ⊑ ⋆Employee:¬Take_bonus@x),
$\Diamond^+$ (x)(**starts now** x)(⊥ ⊑ ⋆Employee:¬Suspended @x),
$\Diamond^+$ (x)(**starts now** x)(⊥ ⊑ ⋆Employee: Take_promotion@x)
$\Diamond^+$ (x)(**starts now** x)(⊥ ⊑ ⋆Employee: Illegal@x)
$\Diamond^+$ (x)(**starts now** x)(⊥ ⊑ ⋆Employee:¬Illegal@x)
}, where ⊥ is the symbol for FALSE.

As we observe, for the fluent that there is not a static rule, we add the rule FALSE ⊑ F, because they cannot become true by static rules, but only by dynamic rules (this means that the truth value changes only as the direct effect of some action). Now we have:

$\Diamond^+$ (x y)(**starts now** x)(**starts now** y)((⋆Employee: ¬Suspended@x ⊓ ⋆Employee: Good_employee@y) ⊓ ⋆Employee:¬Good_Employee@x) for (Take_bonus, ¬Take_bonus)

$\Diamond^+$ (x y)(**starts now** x) (⋆Employee: Suspended@x ⊓ ⋆Employee: ¬Suspended@x) for (Take_salary, ¬Take_Salary)

$\Diamond^+$ (x)(**starts now** x)(⋆Employee:Illegal@x ⊓ ⊥) for (Suspended, ¬Suspended)

$\Diamond^+$ (x y)(**starts now** x)(**starts now** y)((⋆Employee:Illegal@x ⊔ ⋆Employee: ¬Good_employee@y) ⊓ ⊥) for (Take_promotion, ¬Take_promotion)

⊥ ⊓ ⊥ for (Illegal, ¬Illegal)

This assumption $G_f \sqcap K_f \equiv \bot$ is very important in order to ensure that, always after the execution of action there is a consistent situation. Now we show with an example, that if this assumption does not hold, the situation is not consistent after the execution of some sequence of actions.

Consider the above example with the public worker and assume that there is another integrity constraint specifying that when a public worker is Good_employee, then s/he takes promotion. Now the set of static rules is:

R = {$\Diamond^+$ (x)(**starts now** x)(⋆Employee:Illegal@x ⊑ ⋆Employee:Suspended@x),
$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z max(x,y))(⋆Employee:Illegal@x ⊔ ⋆Employee: ¬Good_employee@y ⊑ ⋆Employee: ¬Take_promotion@z),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee:Suspended@x ⊑ ⋆Employee: ¬Take_salary@x),
$\Diamond^+$ (x y z)(**starts now** x)(**starts now** y)(= z min(x, y)) (⋆Employee: ¬Suspended@x ⊓ ⋆Employee: Good_employee@y ⊑ ⋆Employee: Take_bonus@z),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬ Good_employee@x ⊑ ⋆Employee: ¬Take_bonus@x),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: ¬Suspended@x ⊑ ⋆Employee: Take_salary@x),
$\Diamond^+$ (x)(**starts now** x)(⋆Employee: Good_employee@x ⊑ ⋆Employee: Take_promotion@x)

}.

As we observe for the pair (Take_promotion, ¬Take_promotion), the above assumption does not hold, because

$\Diamond^+$ (x)(**starts now** x)(starts now y)(Good_employee@x ⊓ (Illegal@x ⊔ ¬Good_employee@y)) can be true when Good_employee ⊓ Illegal holds.

Assume now that we have a public worker ⋆E and the initial situation is:

S_0 = {$\Diamond^+$ (x) (starts now x) (⋆E:¬Take_bonus@x, ⋆E:Take_salary@x, ⋆E:¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E:¬Good_employee@x, ⋆E:¬Illegal@x)}

As we have mentioned before, x has the maximum value possible for a future time interval which is [now, ∞). Assume that the following actions occur at the following time points, assuming time starts at 0 and time granularity is that of months.

Misdemeanor@4
Good_grade@6

At time point 4 after the execution of the action Misdemeanor we have the situation:

S_1' = {$\Diamond^+$ (x) (starts now x) (⋆E:¬Take_bonus@x, ⋆E:Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: ¬Suspended@x, ⋆E: ¬Good_employee@x, ⋆E: Illegal@x)}

After the evaluation of the static rules we have:

S_1 = {$\Diamond^+$ (x) (starts now x) (⋆E:¬Take_bonus@x, ⋆E:¬Take_salary@x, ⋆E: ¬Take_promotion@x, ⋆E: Suspended@x, ⋆E: ¬Good_employee@x, ⋆E: Illegal@x)}

At time point 6, after the execution of the action Good_grade we have the situation:

S_2' = {$\Diamond^+$ (x) (starts now x) (⋆E:¬Take_bonus@x, ⋆E:¬Take_salary@x, ⋆E:¬Take_promotion@x, ⋆E: Suspended@x, ⋆E: Good_employee@x, ⋆E: Illegal@x)}

Now the static rule $\Diamond^+$(x)(**starts now** x)(⋆Employee: Good_employee@x ⊑ ⋆Employee: Take_promotion@x) must be evaluated, and after that Take_promotion@x must hold. But if Take_promotion@x holds, then we must examine if the static rule $\Diamond^+$(x y z)(**starts now** x)(**starts now** y)(= z max(x,y))(⋆Employee: Illegal@x ⊔ ⋆Employee:¬Good_employee@y ⊑ ⋆Employee:¬Take_promotion@z), must be evaluated. We observe that we must evaluate this static rule as well. As we see, those two static rules will be evaluated one after the other for ever (infinitely). This means that the situation is not consistent. This happened because there is a mistake in the integrity constraints, and therefore the above assumption does not hold. The algorithm can run without the above assumption, but we must determine the preconditions of each action, in order to avoid the above problem. We have proved the following theorems:

Theorem 1: Each time unit, the algorithm is terminated at a finite number of steps.
Theorem 2: *The above algorithm always returns a legal situation.*

4 Conclusion

In this paper we have presented the basics of Description Logics, as well as several approaches to Temporal Description Logics found in Literature. We also presented a Temporal Description Logics representation, used in a thorough example to come up with a solution to the ramification problem in Temporal Settings. In order to accomplish that, we present algorithms that utilize Integrity constraints along with static and dynamic rules, all expressed in Temporal Description Logics. In this particular example time intervals are not only variables but can have enumerated values, in contrast with most examples found in Literature.

As we showed it is possible to deal with the ramification problem with rules and algorithms expressed in whole in Temporal Description Logics. Our implementation can also work with actions taking place in the past, by applying the same algorithms to evaluate static and dynamic rules and to come up with a consistent situation. The Temporal Description Logics representation and algorithms we presented, could also work with non instant actions (actions with duration).

References

1. Allen, J. F. & Ferguson, G.: Actions and Events in Interval Temporal Logic (1994)
2. Artale, A. & Franconi, E.: A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence* **30** (1-4), pp 171--210 (2000)
3. Artale, A. & Franconi, E.: A Temporal Description Logic for Reasoning about Actions and Plans. *J. Artif. Int. Res.* **9** (1), pp 463--506 (1998)
4. Artale, A. & Franconi, E.: Introducing Temporal Description Logics. In TIME '99: Proceedings of the Sixth International Workshop on Temporal Representation and Reasoning (pp. 2). IEEE Computer Society. (ISBN: 0-7695-0173-7.) (1999)
5. Artale, A. & Franconi, E.: Temporal Description Logics. In Handbook of Time and Temporal Reasoning in Artificial Intelligence. MIT Press (2000)
6. Artale, A., Franconi, E., C, M. M., Frank, W. & Zakharyaschev, M.: The DLR_US Temporal Description Logic. In Handbook of Time and Temporal Reasoning in Artificial Intelligence (pp. 96-105). MIT Press (2001)
7. Baader, F. & Nutt, W.: Basic Description Logics. , pp 43--95 (2003)
8. Calvanese, D., Giacomo, G. D., Nardi, D. & Lenzerini, M.: Reasoning in Expressive Description Logics. , pp 1581--1634 (2001)
9. Franconi, E. & Toman, D.: Fixpoint Extensions of Temporal Description Logics. In Description Logics (2003)
10. Lutz, C.: Interval-based Temporal Reasoning with General TBoxes. In *IJCAI* pp. 89--96 (2001)
11. Papadakis, N. & Plexousakis, D.: Actions with Duration and Constraints: the Ramification Problem in Temporal Databases. In *ICTAI* pp. 83--90 (2002)
12. Wolter, F. & Zakharyaschev, M.: Temporalizing Description Logics. In *In Proceedings of FroCoS'98* pp. 104--109 (1999)
13. Zhang, J.: Description Logics and Time (2006)