Project Title:

Temporal Nonmonotonic description logic

Supported By:

RIIMA Laboratory
University of Science and Technology Houari Boumediene (USTHB)

TECHNICAL REPORT

Number: TR-2013-01
Date: March, 2013

Authors:

PHD Student. Ouarda Bettaz
Professor. Aicha Mokhtari
Assistant Professor. Narhimene boustia

Abstract

In this ongoing work, we present a new type of description logic, which is at the same time temporal and nonmonotonic. This kind of description logic allows representing both default and temporal features in concepts definition. We performed this task by extending the nonmonotonic description logic JClassic$_{\delta\varepsilon}$ with temporal aspects. JClassic$_{\delta\varepsilon}$ allows representing default and exception properties in concepts definition. It permits actually to go beyond the strict limitations on concept description and allows consequently to fully define them; by providing both necessary and sufficient conditions for their representation, which allows the classification process. Contrary to the use of strict knowledge that provides only necessary conditions leaving the concepts partially defined. However we frequently need to add the temporal aspect to the nonmonotonic feature as it's the case in causal reasoning, planning process, action theory and access control. Our aim is to extend this logic further with temporal connectives to grant the possibility to represent temporal properties of concepts and that by referring to temporal description logic.

Keywords: *Nonmonotonic description logic, Default knowledge, Temporal description logic, Temporal nonmonotonic description logic.*

1. Introduction

Description logics (DLs) are good formalisms for knowledge base representation [7]. However classical forms of description logics do not permit to represent neither default nor exception facts about concepts, for example: they do not allow representing the fact that all birds fly by default, but a penguin is a bird that exceptionally doesn't fly. The impossibility of representing this kind of information leaves the knowledge base partially defined which subsequently affects the inference process. The solution to represent such kind of knowledge is to rely on nonmonotonic reasoning that is based on the use of default description logic. Many approaches were proposed in the literature: Quantz and Royer [15], Padgham and Nebel [13], Padgham and Zhang [14], Baader and Hollunder [6]. The problem with these approaches is that they use a limited form of default reasoning; where concepts are defined only by using strict properties while default knowledge is represented using incidental rules, considering the fact that most of concepts can't be just defined by the use of strict properties, that will leave the knowledge base inevitably partially defined, consequently the classification process won't be complete. The approach that overcomes this problem was proposed by Coupey and Fouqueré [11]. In fact they developed a new nonmonotonic description logic named $\mathcal{AL}_{\delta\varepsilon}$ that permitted the introduction of the notion of default and exception in concepts definition, it was elaborated by adding to the description logic $\mathcal{AL}$ [7] two connectives: (δ) to represent default facts and (ε) to represent exception facts. This language was improved by the addition of connectors from C-classic which permitted to augment its expressivity and thus make it usable from a practical point of view. This new language was called JClassic$_{\delta\varepsilon}$ [8], [9]. Using this description logic we can define the concept Tree as having by default branches and always having a trunk and roots:

$$Tree \equiv \delta\ With\_branches \sqcap\ With\_trunk \sqcap With\_roots$$

Now if we want to define the concept Scion as being a tree that is by default one year old and exceptionally has branches we will write it this way:

$$Scion \equiv \delta\ One\_year\_old \sqcap Tree \sqcap With\_branches^{\varepsilon}$$

In this example the concept *Scion* that is subsumed by the concept *Tree* will only inherit the properties *With_trunk* and *With_roots*, but not the property *With_branches* since this property is an exception for the concept *Scion*.

In this work we developed a temporal nonmonotonic description logic and that by adding to the nonmonotonic description logic *JClassic*$_{\delta\varepsilon}$ temporal connectives. The purpose of doing so is to represent temporal concepts while having default knowledge. Differing from the existing temporal description logics where temporal components are added to classical description logics. This will permit to better manage the time aspect in a variety of domains such as reasoning about actions and plans and enhance natural languages

comprehension…etc, it will also allow us to improve access control, actually our future aim is to define a dynamic acces control model with the use of the temporal nonmonotonic description logic . Indeed we developed this description logic initially for that purpose. To reach our goal we referred to the use of temporal description logic (TDL) [1], [2], [3], [4], [7], [10]. In the research field on temporal description logic two approaches for modeling the notion of time were considered: the modal temporal logic and the reified temporal logic [12]. In our work we will employ modal temporal logic. In which the connectors □ and ◊ represents respectively the notion of (always in the future) and (sometimes in the future). The flow of time can be taken from two different angles, we can actually consider time as a set of points (instances) or as a set of intervals. In [1], [2], [7], [10], the different approaches on temporal logic based on points and intervals have been widely spread. For the purpose of our work we are interested in the use of interval based approach to define the specific interval at which a concept is valid. Concerning this approach many studies were undertaken. Artale and Franconi [1], [2], [3] put into evidence a TDL inspired by Schmiedel's [16] approach, that they restricted by discarding the □ operator for decidability matters. Example [5]:

◊ *(X Y) (Y starts X).(Student@ X ⊓ Bachelor_student @Y).*

In this example, we have two intervals *X* and *Y*, where *X* and *Y* start at the same time but *Y* is ended before *X*. So the described persons are students during the interval *X* and they are specifically Bachelor students for the initial sub-interval *Y* of *X*. The temporal part $\mathcal{TL}$ that we will be using for extending the nonmonotonic description logic is the one used in the TDL defined by Artale and Franconi [1], [2] and [3].

The rest of the paper is organized as follows: In the second section we introduce our temporal nonmonotonic description logic $TDL_{\delta\varepsilon}$ by providing its syntax, and we put forward an algebraic framework with both descriptive and structural point of view. We also describe the algorithm for subsumption calculation and provide elements that allows us to show in the future that t-$sub_{\delta\varepsilon}$ is correct, complete and with a polynomial complexity. Finally we define the inheritance algorithm. We conclude this work in section 3.

2. $TDL_{\delta\varepsilon}$

In this section we introduce our new temporal nonmonotonic description logic that we elaborated by the combination of the nonmonotonic description logic $JClassic_{\delta\varepsilon}$[8], [9] and the temporal component **@** from $\mathcal{TL}$ [1], [2], [3]. The result of this process is the temporal nonmonotonic description logic that we named $TDL_{\delta\varepsilon}$. It consists of: a set of atomic concepts *P* and atomic roles *R*, the two constants $\top$ (Top) and $\bot$ (Bottom) that represents respectively the universal and the bottom concept, a set of individuals *I* called 'classic individuals', the concepts *C* and *D*, the unary connectives δ *(Default)* and ε *(Exception)*, the binary conjunction $\sqcap$, the quantifier $\forall$ that enables universal quantification on role values, and the temporal qualifier @ to represent the interval '*X*' at which a concept *C* applies, *u* is a real number, *n* is an integer, *Ii* Are 'classic individuals'.

Table1. Syntax of $TDL_{\delta\varepsilon}$

C, D → P	(Atomic concept)
\| ⊤ | (Universal Concept) |
\| ⊥ | (Bottom concept) |
\| ¬P | (Atomic negation) |
\| C ⊓ D | (Intersection) |
\| Min u | (u is a real number) |
\| Max u | (u is a real number) |
\| ONE-OF{I1,…In} | (Concept in extension) |
\| R FILLS {I1,…In} | (Subset of value for R) |
\| R AT-LEAST n | Cardinality for R (minimum) |
\| R AT-MOST n | Cardinality for R (maximum) |
\| ∀ R.C | (Universal quantifier) |
\| δ C | (Concept C by default) |
\| C^{ε} | (Exception to the concept C) |
\| C@X | (Qualifier) |

Using this description logic we can represent temporal aspects, the properties: default, exception, exception of exception, and so on. For instance in the case of access control in the medical domain, we can define the concept *Doctor* as being a *Staff member* that *Exercises* officially his function by default and that has the right to Access the medical database records of patients during *Working hours*.

$Doctor \equiv Staff\text{–}Member \sqcap \delta Exercise \sqcap Access - Mdb\text{–}Records@(working\ Hours).$

Now we can define the concept *Resident* as a *Doctor* that exceptionally doesn't *Exercise* officially since he is still a student.

$Resident \equiv Doctor \sqcap Exercise^{\varepsilon}$

Here the concept *Resident* will inherit the property *Staff Member* and the right to *Access* the medical database records during working hours but not the property *Exercise* since it is an exception for the concept *Resident*. In the case where we have a context of *Emergency* another exception on the concept *Exercise* is applied for *Resident*.

$Resident \sqcap Emergency \equiv Doctor \sqcap (Exercise^{\varepsilon})^{\varepsilon}$

In this case, the exception over an exception omits the exception, therefore in an emergency context *Resident* has the right to *Exercise*.

The specificity of this logic is the use of an algebraic-based semantics unlike the classical practice where the semantics is based rather on a first order logic interpretation. Following this algebraic semantics, the concepts are written in a particular form called normal form.

2.1. Algebraic Framework

$TDL_{\delta\varepsilon}$ is endowed with an algebraic framework; it allows distinguishing and formalizing the different types of subsumption namely descriptive and structural.

2.1.1. Descriptive point of view

From a descriptive point of view the calculation of subsumption consists of the comparison of terms through an equational system called *EQ* which defines formally the main properties of connectors and determines the equivalence classes of terms.

EQ: An equational System for $TDL_{\delta\varepsilon}$

$\forall A, B, C, D \in TDL_{\delta\varepsilon}$

01: $(A \sqcap B) \sqcap C = A \sqcap (B \sqcap C)$
02: $A \sqcap B = B \sqcap A$
03: $A \sqcap A = A$
04: $\top \sqcap A = A$
05: $\bot \sqcap A = \bot$
06: $(\forall R:A) \sqcap (\forall R:B) = \forall R:(A \sqcap B)$
07: $\forall R:\top = \top$
08: $(\delta A)^{\varepsilon} = A^{\varepsilon}$
09: $\delta(A \sqcap B) = (\delta A) \sqcap (\delta B)$
10: $A \sqcap \delta A = A$
11: $A^{\varepsilon} \sqcap \delta A = A^{\varepsilon}$
12: $\delta\delta A = \delta A$
13: $C @ X \sqcap D @ X = (C \sqcap D)@X$
14: $(C@X1) @ X2 = (C@X1)$
15: $(C@X1 \sqcap D) @ X2 = C@ X1 \sqcap D@ X2$

The first twelve axioms correspond to $JClassic_{\delta\varepsilon}$ connectives [8], [9] and the axioms from thirteen to fifteen correspond to the temporal operator @. The axioms express that the conjunction of concepts is:
01: associative, 02: commutative, 03: idempotent.

04: the most general concept in the hierarchy top (⊤) is the neutral element of the conjunction.
05: the most specific concept bottom (⊥) is the absorbent element.
06: the connector ∀R:A is distributive over the conjunction.
07: represent a false restriction on roles.
08: an exception to the default concept is the same as an exception to the underlying concept.
09: a default on a conjunction of concepts is similar to the conjunction of two defaults.
10 and 11: express the fact that both A and A^{ε} are subsumed by δA.
12: Allows removal of redundant default chains.
13: @ operator is distributive over the conjunction.
14: Concept *C* is valid at the first interval it is related to.
15: @ operator is distributive over conjunction, but still C is valid at the first interval it is related to.

Descriptive subsumption denoted by $\sqsubseteq_d$ is a partial order relation to order terms. Equality (modulo the axioms EQ) between two terms is denoted $=_{EQ}$. which is a congruence relation that partitions the set of terms, i.e. $=_{EQ}$ allows to form equivalence classes between terms. The descriptive subsumption is defined using the congruence relation and conjunction of concepts as follows:

Definition 1. *(Descriptive subsumption)* Let *C* and *D* be two terms of $TDL_{\delta\varepsilon}$, $C \sqsubseteq_d D$, i.e. *D* subsumes descriptively *C*, iff $C \sqcap D =_{EQ} C$.

From an algorithmic point of view terms are not easily manipulated through subsumption. For this a structural point of view should be adopted which is closer to the algorithmic aspect of computing subsumption. This allows us to formalize calculation of subsumption in the implementation of $TDL_{\delta\varepsilon}$ and also to endow it with an intentional semantic.

2.1.2. Structural point of view

We present in this section the structural point of view for the subsumption in $TDL_{\delta\varepsilon}$. This provides a very closer vision to the algorithmic aspect and a formal framework to validate the algorithmic approach. For this purpose we define $CL_{\delta\varepsilon\text{-}t}$ an intentional semantic for $TDL_{\delta\varepsilon}$. Elements from $CL_{\delta\varepsilon\text{-}t}$ are the canonical intentional representation of terms of $TDL_{\delta\varepsilon}$ that allows representing the properties of concepts by using a normal form. These elements are formed of a pair of 7-uples, both having the same structure, the first is used to represent the strict properties and the second for the default properties, the six first fields of the 7-uples are the same as for the normal form defined for $JClassic_{\delta\varepsilon}$[8], [9] we introduced the 7th field to represent the temporal aspect. The structure of the elements of $CL_{\delta\varepsilon\text{-}t}$ is as follows:

Definition 2. An element of $CL_{\delta\varepsilon\text{-}t}$ corresponding to a term T of $TDL_{\delta\varepsilon}$ is a pair $<t_\theta, t_\delta>$, where t_θ is the strict part of T and t_δ the default part of T, t_θ and t_δ are 7-uples (*dom, min, max, π, r, ε, t*) defined as follows:

dom: is a set of individuals if the definition of T contains a property ONE-OF otherwise the special symbol UNIV.

min(resp. ***max***): its either a real if T contains a property MIN (resp MAX), or the special symbol MIN-R (resp. MAX-R) otherwise.

π: is the set of primitive concepts contained in T.

r: contains the following elements: <*R, fillers, least, most, c* > where:

R: is the role name.

fillers: is a set of individuals if T contains a property R FILLS or Ø otherwise.

least (resp. *most*): is an integer if T contains the property R AT-LEAST (resp. R AT-MOST) or 0 (resp. NOLIMIT) otherwise.

c: is the normal form of *C* if T contains the property $\forall R.C$.

ε: is a set of seven elements (*dom, min, max, π, r, ε, t*).

t: is the set of temporal concepts contained in T, it contains (*dom, min, max, π, r, ε, t*) where:

dom: is ONE-OF " *Saturday, Sunday, Monday, Tuesday, Wednesday, Thursday, Friday*" if the temporal context is a day of the week, otherwise the special symbol TIME if the temporal context is a time interval.

min (resp. ***max***): is a real that represents the beginning (resp. the end) of an interval if the context is a time interval otherwise Ø if the temporal context is a day of the week. Example the interval [8:00, 10:00] the min takes the value "8" and Max the value "10".

Notation: the complete structure is noted : $<(t_{\theta dom}, t_{\theta min}, t_{\theta max}, t_{\theta\pi}, t_{\theta r}, t_{\theta\varepsilon}, t_{\theta t}), (t_{\delta dom}, t_{\delta min}, t_{\delta max}, t_{\delta\pi}, t_{\delta r}, t_{\delta\varepsilon}, t_{\delta t}) >$.

We give an example about the normal form of a given concept *Permission*, in the domain of access control, *Permission(P1)*, is written as follows:

Permission(P1) ≡ *Permision(P2)* ⊓ *δpermission(P3)* ⊓ *Permission(p3)*$^{\varepsilon}$ ⊓ *Permission(P4)@i.*

The normal form is the following:

(<Univ, Min-R, Max-R, {Permission(P2)}, Ø, {Permission(P3)}, {Permission P4} >, < Univ, Min-R, Max-R, {Permission (P2), permission(P3)}, Ø, {Permission(P3)}, {Permission P4}>).

Definition 3. *(Homomorphism h)*

The interpretation of connectors and constants of $CL_{\delta\varepsilon\text{-}t}$ are denoted in **Table 2**, b_0 is a constant used as a denotation of ⊥.

Table2. Homomorphism h

$TDL_{\delta\varepsilon}$	**$CL_{\delta\varepsilon\text{-}t}$**
⊤	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
P	≺(UNIV, MIN-R, MAX-R, P, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, P, Ø, Ø, Ø)≻
ONE-OF E	≺(E, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (E, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
MIN u	≺(UNIV, u, MAX-R, Ø, Ø, Ø, Ø), (UNIV, u, MAX-R, Ø, Ø, Ø, Ø)≻
MAX u	≺(UNIV, MIN-R, u, Ø, Ø, Ø, Ø), (UNIV, MIN-R, u, Ø, Ø, Ø, Ø)≻
∀ R: C (C ≢⊤ and C ≢⊥)	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, $\|C_{\theta.dom}\|$, C≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, $\|C_{\theta.dom}\|$, C≻}, Ø, Ø)≻
R FILLS E (E ≠ Ø)	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, E, |E|, NOLIMIT, t≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, E, |E|, NOLIMIT, t≻}, Ø, Ø)≻
R FILLS Ø	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
R AT-LEAST 0	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
∀ R: C and C ≡ ⊤	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
R AT-LEAST n (n≥ 1)	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , n, NOLIMIT, t≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, n, NOLIMIT, t≻}, Ø, Ø)≻
R AT-MOST n (n≥ 1)	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, n, t≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, n, t≻}, Ø, Ø)≻
R AT-MOST 0	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, 0, b0≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, 0, b0≻}, Ø, Ø)≻
∀ R: C and C≡ ⊥	≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, 0, b0≻}, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, 0, b0≻}, Ø, Ø)≻
C ⊓ D	c ⊗ d
δC	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), c_{δ}≻
C^{ε}	≺ (UNIV, MIN-R, MAX-R, Ø, Ø, c_{δ}, Ø), ($c_{\delta.dom}$, $c_{\delta.min}$, $c_{\delta.max}$, $c_{\delta.\pi}$, $c_{\delta.r}$, $c_{\delta.\varepsilon}$∪c_{δ}, Ø) ≻
⊥	b_0
C@t	≺(UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺Dom_t, Min_t, Max_t≻}), (UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺Dom_t, Min_t, Max_t≻})≻
Dom_t≡ TIME	≺(UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺ TIME, Min_t, Max_t≻}), (UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺ TIME, Min_t, Max_t≻})≻
Dom_t≡ DATE	≺(UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺ ONE OF WEEK-DAYS, Ø, Ø ≻}),(UNIV, MIN-R, MAX-R, C, Ø, Ø, {≺ ONE OF WEEK-DAYS, Ø, Ø ≻})≻
δC@t	≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø) , c_{δ}@t≻
C^{ε}@t	≺(UNIV, MIN-R, MAX-R, Ø, Ø, c_{δ}, ≺Dom_t, Min_t, Max_t≻), ($c_{\delta.dom}$, $c_{\delta.min}$, $c_{\delta.max}$, $c_{\delta.\pi}$, $c_{\delta.r}$, $c_{\delta.\varepsilon}$∪c_{δ} , $c_{\delta.t}$) ≻

We should now define the union of two elements of $CL_{\delta\varepsilon\text{-}t}$ ($\otimes$), This operation completes the definition of the normal form

Definition 4. *(Union of two normal forms)*
For each normal form c of $CL_{\delta\varepsilon\text{-}t}$ we have: $b_0 \otimes c = c \otimes b_0$.
Given c and d two normal forms of $CL_{\delta\varepsilon\text{-}t}$ different from b_0 with the following form:
$c = <(c_\theta, c_\delta)>$ and $d = <(d_\theta, d_\delta)>$
$c \otimes d = <(c_\theta \oplus d_\theta), (c_\delta \oplus d_\delta)>$

This definition uses a union operation of two seven uples of $CL_{\delta\varepsilon\text{-}t}$ ($\oplus$), the union of the seven uples is performed on each field (dom, min, max, π, r, ε, t).
The calculation of the union is done through the semantic function Union-uple (A, B) (this is for $A \oplus B$)
We should now define the algorithm of the union of two 7 uples proper to $CL_{\delta\varepsilon\text{-}t}$.

Algorithm 1. Union-uple (A, B)

Require: two 7-uple A and B: (a_{dom}, a_{min}, a_{max}, a_π, a_r, a_ε, a_t) and (b_{dom}, b_{min}, b_{max}, b_π, b_r, b_ε, b_t)
Ensure: (u_{dom}, u_{min}, u_{max}, u_π, u_r, u_ε, u_t) the result uple of the union of A and B
$u_{dom} \leftarrow a_{dom} \cup b_{dom}$
$u_{min} \leftarrow maxi(a_{min} \cup b_{min})$
$u_{max} \leftarrow mini(a_{max} \cup b_{max})$
$u_\pi \leftarrow a_\pi \cup b_\pi$
$u_\varepsilon \leftarrow a_\varepsilon \cup b_\varepsilon$
$u_r \leftarrow \emptyset$
$u_t \leftarrow a_t \cup b_t$
for $<R, fills1, least1, most1, c1> \in a_r$ **do**
  if $\exists <R, fills2, least2, most2, c2> \in b_r$ **then**
    $u_r \leftarrow u_r \cup <R, fills, least, most, c>$ with
    $c \leftarrow c1 \otimes c2$
    $fills \leftarrow fills1 \cup fills2$
    $least \leftarrow maxi(least1, least2, \left|fills\right|)$
    $most \leftarrow mini(most1, most2, \left|C_{\theta dom}\right|)$
  else
    $u_r \leftarrow u_r \cup <R, c1>$
  end if
end for
for $<R, c2> \in b_r$, such as $\nexists$ element with the name R in a_r **do**
    $u_r \leftarrow u_r \cup <R, c2>$
end for
for $< Dom_{t1}, Min_{t1}, Max_{t1}> \in a_t$
  if $\exists < Dom_{t2}, Min_{t2}, Max_{t2}> \in b_t$ **then**
    $u_t \leftarrow u_t \cup <Dom_t, Min_t, Max_t>$ with
    $Dom_t \leftarrow Dom_{t1} \cup Dom_{t2}$
    $Min_t \leftarrow maxi(Min_{t1} \cup Min_{t2})$
    $Max_t \leftarrow mini(Max_{t1} \cup Max_{t2})$
  else
    $u_t \leftarrow u_t \cup < Dom_{t1}, Min_{t1}, Max_{t1}>$
  end if
end for
for $< Dom_{t2}, Min_{t2}, Max_{t2}> \in b_t$, such as $\nexists$ elements $< Dom_{t1}, Min_{t1}, Max_{t1}>$ in a_t **do**
    $u_t \leftarrow u_t \cup < Dom_{t2}, Min_{t2}, Max_{t2}>$
end for

Structural subsumption: two terms of $TDL_{\delta\varepsilon}$ are structurally equivalent iff their normal forms are equal. The notation $\sqsubseteq_s$ for structural subsumption is a partial order relation.

The structural equality of two terms of $TDL_{\delta\varepsilon}$ is noted $=_{CL}$ it's a congruence relation as $=_{EQ}$ in descriptive subsumption. We define the structural subsumption using the congruence relation and conjunction of concepts as follows:

Definition 5. *(Structural subsumption)* Let *C* and *D* two terms of $TDL_{\delta\varepsilon}$, $C \sqsubseteq_s D$; i.e. *D* subsumes structurally *C*, iff $C \sqcap D =_{CL} C$.

Theorem 1. *(Equivalency between descriptive subsumption and structural subsumption).*
Let C and D two terms of $TDL_{\delta\varepsilon}$
$C \sqsubseteq_s D \Leftrightarrow C \sqsubseteq_d D$

To infer new knowledge in this system, the subsumption relation is used. In the next section, we outline the subsumption algorithm $t\text{-}sub_{\delta\varepsilon}$ used for $TDL_{\delta\varepsilon}$. The other inference operation in concern is inheritance it will be presented in the upcoming sections.

2.2. Subsumption algorithm $t\text{-}sub_{\delta\varepsilon}$

We present in this section the algorithm for subsumption calculation, this algorithm lays on the calculation of structural subsumption described previously, we will define the algorithm $t\text{-}sub_{\delta\varepsilon}$ and the procedures it uses, we also provide elements that allows us to show that $t\text{-}sub_{\delta\varepsilon}$ is correct, complete and with a polynomial complexity.

2.2.1. Presentation of the algorithm $t\text{-}sub_{\delta\varepsilon}$.

$t\text{-}sub_{\delta\varepsilon}$ is composed of two stages. The first is a normalization of descriptions. The second is a syntactic comparison between normal forms. Let *C* and *D* be two terms of $TDL_{\delta\varepsilon}$. To answer the question "Is C subsumed by D?" we apply the next procedure. The normal form of *C* and $C \sqcap D$ are calculated with the procedure of normalization. If the two normal forms are equal, the algorithm returns "Yes" otherwise it returns "No".

- *Procedure of normalization of description.*

  This procedure uses the semantics functions union-uple and $\otimes$ which compute respectively the union of two 7-uples ($\oplus$) and two normal forms. The normalization (denoted in Algorithm 3.) permits to calculate the normal form of a concept *C* from its given description denoted by *fn(C).*

- *Procedure of comparison of normal forms.*

  The procedure *Compar* (defined in Algorithm 7.) which checks equality between two 7- uples t_1 and t_2 . t_1 resp t_2 have the form $(t_{i.dom}, t_{i.min}, t_{i.max}, t_{i.\pi}, t_{i.r}, t_{i.c}, t_{i.t})$ with $i=1$ (resp $i=2$). *Compar* calls the procedure *Compar-roles* (defined in Algorithm 8.) which allows checking equality of sets which denote roles. This procedure is denoted *Compar(fn(C1), fn(C2), rep).*

2.2.1.1. subsumption algorithm $t\text{-}sub_{\delta\varepsilon}$

Algorithm 2. Algorithm $t\text{-}sub_{\delta\varepsilon}$

Require: C and D two description of concepts of $TDL_{\delta\varepsilon}$
Ensure: Response "Yes" or "No" to question "Is C subsumed by D?"
{Compute normal form}
fn(C) ← Normalization (C)
fn(C ⊓ D) ← Normalization (C ⊓ D)
{Treatment of bottom}
if fn(C)=b_0 **then**
  Response ← "Yes"
else
  if fn(C ⊓ D)= b_0 **then**
    Response ← "No"
  else
    {Comparison of the obtained normal forms}
    Compar(fn(C) $_\theta$, fn(C ⊓ D) $_\theta$, rep1)
    if rep1="Yes" **then**
      Compar(fn(C) $_\delta$, fn(C ⊓ D) $_\delta$, rep1)
      Response ← rep2
    else
      Response ← "No"
    end if

end if
end if

2.2.1.2. Normalization Procedure

The Normalization algorithm that will be described next puts into evidence the homomorphism described previously, it handles simple and composed terms, therefore it calls two other algorithms namely simpleterm and composedterm. The procedure of Normalization allows the calculation of the normal form (denotation) of a concept *C* given its description. This procedure of normalization uses the semantic functions union-uple ($\oplus$) and ($\otimes$), which calculates respectively the union of two 7 uples and two normal forms.

Algorithm 3. Normalization procedure

Require: D is a description of a concept of $TDL_{\delta\varepsilon}$
Ensure: Normal form of D in $CL_{\delta\varepsilon\text{-}t}$ (i.e. its normal form) fn(D), D has the form D ≡ T1 ⊓ T2 ⊓ …⊓Tn with n ≥ 1
If (D is ⊤) or (D is ⊥) or (D is a primitive concept) or (D is ONE-OF E) or (D is MIN u) or (D is MAX u) or (D is R FILLS E (E ≠ Ø)) or (D is R FILLS Ø) or (D is R AT-LEAST n (n≥1)) or (D is R AT-LEAST 0) or (D is R AT-MOST n (n≥1)) or (D is R AT-MOST 0) or (D is ∀ R: ⊤) or (D is ∀ R: ⊥) or (D is C@t) or (D isDom_t≡ TIME) or (D is Dom_t≡ DATE) **then**
  simpleterm
else
  composedterm
end if

Algorithm 4. Simpleterme procedure

if D is ⊤**then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
end if
if D is ⊥**then**
  fn(D) ←b_0
end if
if D is a primitive concept **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, {P}, Ø, Ø, Ø), (UNIV, MIN-R, MAX-R, {P}, Ø, Ø, Ø)≻
end if
if D is ONE-OF E **then**
  fn(D) ← ≺(E, MIN-R, MAX-R, Ø, Ø, Ø, Ø), (E, MIN-R, MAX-R, Ø, Ø, Ø, Ø)≻
end if
if D is MIN u **then**
  fn(D) ← ≺(UNIV, u, MAX-R, Ø, Ø, Ø, Ø), (UNIV, u, MAX-R, Ø, Ø, Ø, Ø)≻
end if
if D is MAX u **then**
  fn(D) ← ≺(UNIV, MIN-R, u, Ø, Ø, Ø, Ø), (UNIV, MIN-R, u, Ø, Ø, Ø, Ø)≻
end if
if D is R FILLS E (E ≠ Ø) **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, E, |E|, NOLIMIT, t≻}, Ø, Ø),
    (UNIV, MIN-R, MAX-R, Ø, {≺R, E, |E|, NOLIMIT, t≻}, Ø, Ø)≻
end if
if (D is R FILLS Ø) or (D is R AT-LEAST 0) or (D is ∀ R: ⊤) **then**
  fn(D) ← t
end if
if D is R AT-LEAST n (n≥ 1) **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , n, NOLIMIT, t≻}, Ø, Ø),
    (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, n, NOLIMIT, t≻}, Ø, Ø)≻
end if
if D is R AT-MOST n (n≥ 1) **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, n, t≻}, Ø, Ø),
    (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, n, t≻}, Ø, Ø)≻
end if

if D is R AT-MOST 0 **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, 0, b_0≻}, Ø, Ø),
  (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, 0, b_0≻}, Ø, Ø)≻
end if
if D is ∀ R: ⊥**then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, 0, b_0≻}, Ø, Ø),
  (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø, 0, 0, b_0≻}, Ø, Ø)≻
end if
if D is C@t **then**
  fn(D) ← ≺(UNIV, MIN-R, MAX-R, {C}, Ø, Ø,{≺Dom_t, Min_t, Max_t≻}),
  (UNIV, MIN-R, MAX-R, {C}, Ø, Ø, {≺Dom_t, Min_t, Max_t≻})≻
end if
if D is Dom_t≡ TIME **then**
  fn(D) ←≺(UNIV, MIN-R, MAX-R, {C}, Ø, Ø, {≺ TIME, Min_t, Max_t≻}),
  (UNIV, MIN-R, MAX-R, {C}, Ø, Ø, {≺ TIME, Min_t, Max_t≻})≻
end if
if D is Dom_t≡ DATE **then**
  fn(D) ←≺(UNIV, MIN-R, MAX-R, {C}, Ø, Ø, {≺ ONE OF WEEK-DAYS, Ø, Ø ≻}),
  (UNIV, MIN-R, MAX-R, {C}, Ø, Ø, {≺ ONE OF WEEK-DAYS, Ø, Ø ≻})≻
end if

Algorithm 5. Composedterm procedure

if D is ∀ R: C **then**
  fn(C) ←Normalization(C)
 if fn(C) = t **then**
   fn(D) =t
 else
  if fn(C) =b0 **then**
   fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, 0, b_0≻}, Ø, Ø),
   (UNIV, MIN-R, MAX-R, Ø,{≺R, Ø, 0, 0, b_0≻}, Ø, Ø)≻
  else
   fn(D) ← ≺(UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, $|C_{\theta\cdot dom}|$, C≻}, Ø, Ø),
   (UNIV, MIN-R, MAX-R, Ø, {≺R, Ø , 0, $|C_{\theta\cdot dom}|$, C≻}, Ø, Ø)≻
  end if
 end if
end if
if D is δC **then**
  fn(C) ←Normalization(C)
  fn(D) ←≺(UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), $fn(C)_\delta$≻
end if
if D is C^ε **then**
  $fn(D)_{\theta dom}$ ← UNIV
  $fn(D)_{\theta Min}$ ← MIN-R
  $fn(D)_{\theta Max}$← MAX-R
  $fn(D)_{\theta\pi}$← Ø
  $fn(D)_{\theta r}$← Ø
  $fn(D)_{\theta\varepsilon}$← Index-calcul$(fn(C))_\delta$
  $fn(D)_{\delta dom}$ ← $fn(C)_{\delta dom}$
  $fn(D)_{\delta Min}$ ← $fn(C)_{\delta Min}$
  $fn(D)_{\delta Max}$← $fn(C)_{\delta Max}$
  $fn(D)_{\delta\pi}$← $fn(C)_{\delta\pi}$
  $fn(D)_{\delta r}$← $fn(C)_{\delta r}$
  $fn(D)_{\delta\varepsilon}$← $fn(D)_{\theta\varepsilon}$∪$fn(C)_{\delta\varepsilon}$
end if
if D is A ⊓ B **then**
  fn(A) ←Normalization(C)
  fn(B) ←Normalization(B)
  fn(D) ← fn(A)⊗fn(B) {Union of two normal forms}
end if

if D is δC@t **then**
fn (C) ←Normalization(C@t)
fn (D) ←≺ (UNIV, MIN-R, MAX-R, Ø, Ø, Ø, Ø), fn (c@t)$_{\delta}$≻
end if
if D is C$^{\varepsilon}$@t **then**
fn(D)$_{\theta dom}$ ← UNIV
fn(D)$_{\theta Min}$ ← MIN-R
fn(D)$_{\theta Max}$← MAX-R
fn(D) $_{\theta\pi}$← Ø
fn(D)$_{\theta r}$← Ø
fn(D)$_{\theta\varepsilon}$← Index-calcul(fn(C))$_{\delta}$
fn(D)$_{\theta t}$← ≺Dom$_t$, Min$_t$, Max$_t$≻
fn(D)$_{\delta dom}$ ← fn(C)$_{\delta dom}$
fn(D)$_{\delta Min}$ ← fn(C)$_{\delta Min}$
fn(D)$_{\delta Max}$← fn(C)$_{\delta Max}$
fn(D)$_{\delta\pi}$← fn(C)$_{\delta\pi}$
fn(D)$_{\delta r}$← fn(C)$_{\delta r}$
fn(D)$_{\delta\varepsilon}$← fn(D)$_{\theta\varepsilon}$∪fn(C)$_{\delta\varepsilon}$
fn(D)$_{\delta t}$← fn(C@t) $_{\delta}$
end if

To compute normal forms of ε field, we call a function that represents the exceptional concept. For this purpose, we make an index table which includes pairs (number, 7-uple).The index-calculation procedure is denoted in Algorithm 6.

2.2.1.3. Index calculation procedure

Algorithm 6. Procedure index-calcul(s) (s is a 7-uple)

If ∃ a pair (n,s) in index table **then**
Index-Calcul(s)
else
We create a new pair in index table(m, s)
where m=number of pairs in index table + 1
Index-calcul(s) ← m
end if

The computation of the index of a seven uple returns a number that corresponds to the code of the seven uple that will be used in the normal form; this also allows updating the index table.
The saving of space obtained is important in the case where the description handles a series of nested ε.

2.2.1.4. Procedure of comparison of normal forms

We first define the procedure *Compar* which checks equality between two 7- uples *t1* and *t2* .*t1* resp *t2* have the form *($t_{1.dom}$, $t_{i.min}$, $t_{i.max}$, $t_{i.\pi}$, $t_{i.r}$, $t_{i.\varepsilon}$, $t_{i.t}$)* with *i=1* (resp *i=2*). *Compar* calls the procedure *Compar-roles* which allows checking equality of sets which denote roles. This procedure is denoted *Compar(fn(C1), fn(C2), rep)* in algorithm *t-sub$_{\delta\varepsilon}$*.

Algorithm 7. Procedure Compar (t1, t2, Response)

if ($t_{1.dom} \neq t_{2.dom}$) or ($t_{1.min} \neq t_{2.min}$) or ($t_{1.max} \neq t_{2.max}$) or ($t_{1.\pi} \neq t_{2.\pi}$) or
($t_{1.\varepsilon} \neq t_{2.\varepsilon}$) or ($t_{1.t} \neq t_{2.t}$) **then**
Response ← "No"
else
{Comparison of roles field}
Compar-roles($t1_r$, $t2_r$,Rep)
Response ←Rep
end if

Algorithm 8. Compar-roles

Repr ← "Yes"
If $|A_r| \neq |B_r|$ **then**
Repr ← "No"
endif
while($A_r \neq \emptyset$) and (Repr = "Yes") **do**
let e ∈ (e=≺ R, fills1, least1, most1, c1 ≻; search e' in B*r* with the same name R
if e' ∉ B_r **then**
Repr← "No"
else
let e' ∈ B_r (e' = ≺ R, fills2, least2, most2, c2 ≻
endif
if (fills1 ≠ fills2) **or** (least1 ≠ least2) or (most1 ≠ most2) **then**
Repr ← "No"
else
{Comparison of strict and default 7-uples of normal forms c1 and c2}
Compar($c1_\theta$, $c2_\theta$, reps)
If reps= "Yes" **then**
Compar($c1_\delta$, $c2_\delta$, repd)
reps ← repd
endif
else
reps ← "No"
endif
$A_r \leftarrow A_r - e$
$B_r \leftarrow B_r - e'$
endwhile

Corollary 1 . *(Correction of t-sub$_{\delta\varepsilon}$)*
The algorithm that computes the subsumption t-sub$_{\delta\varepsilon}$ is correct since for any concept C and D:
$C \sqsubseteq_s D \Rightarrow C \sqsubseteq_d D$.
t-sub$_{\delta\varepsilon}$ is complete in accordance to $CL_{\delta\varepsilon\text{-}t}$ since any descriptive subsumption implies necessarily a structural subsumption. Thus:

Corollary 2. *(Completeness of t-sub$_{\delta\varepsilon}$)*
The algorithm that computes the subsumption t-sub$_{\delta\varepsilon}$ is complete with regard to $CL_{\delta\varepsilon\text{-}t}$ since for any concept C and D:
$C \sqsubseteq_d D \Rightarrow C \sqsubseteq_s D$

To use the notion of default, we use two approaches: definitional (subsumption) point of view and inheritance point of view. In the next section we describe the inheritance point of view.

2.3. Inheritance point of view

We remind here that the principal objective of the definitional point of view is the subsumption, which refers to concepts classification.

The user actually defines its concepts based on atomic concepts and then the classifier automatically organizes the graph. From inheritance point of view we describe the inherited properties of a concept, which are considered to be the basis of an inferential system.

If for example we consider *T* as being an instance of the concept *Tree* weather directly or not, that has a property *With_branches* by default (which is not excepted). Then *T* is recognized as being an instance of *δWith_branches* and *With_branches* is an inherited property of this instance. If a given user needs to know the number of *Trees* corresponding to the property *With_branches*, then *T* is considered as one of them, however if *T* is also an exception to this property (e.g. It is an instance of the concept *Scion*), then the precedence knowledge is still true (Monotonicity of the classifier), but *T* is not anymore considered as a *Tree* with the property *With_branches* (the non monotonicity of the inheritance process).

The aim of the inheritance process is thus to realize a preference of exception over default. From definitional point of view an exception is subsumed by default, but the latter it is not omitted by the classifier thought it should be retracted by the inheritance process.

If we go back to the example cited previously in this paper concerning the Concept *Penguin*; *Fly* is not an inherited property since it is excepted, however the property *Animal* and the temporal property *Mortal* are inherited.

For this purpose we describe an inheritance map from $CL_{\delta\varepsilon\text{-}t}$ to $CL_{\delta\varepsilon\text{-}t}$. This inheritance procedure is the bases for retrieving the inherited properties. It also helps distinguishing default from strict properties and answering questions concerning conflicts and inconsistencies.

The major task of the inheritance procedure is to deduce the exception from concepts denotations (the two parts ε from their normal form). The scenario for the inheritance calculation procedure for a concept C is the following:

1- Replace each exception at an even level by a default in the denotation of *C*.
2- For each role of *C*, recursively call inheritance with the role value restriction.
3- Suppress P (resp. P^o) in $c_{\delta\pi}$ if P^o (resp. P) is in $c_{\theta\pi}$.

The resulting denotation is called the inheritance form of *C*.

Algorithm 9. Inheritance Map

inheritance: $CL_{\delta\varepsilon\text{-}t} \rightarrow CL_{\delta\varepsilon\text{-}t}$, such that
inheritance(a) =
res ← <($a_{\theta dom}$, $a_{\theta min}$, $a_{\theta max}$, $a_{\theta\pi}$, Ø, Ø, $a_{\theta t}$), ($a_{\delta dom}$, $a_{\delta min}$, $a_{\delta max}$, $a_{\delta\pi}$, Ø, Ø, $a_{\delta t}$) >
for all y ∈ $a_{\theta\varepsilon} \cup a_{\delta\varepsilon}$ **do**
  res ← res ∪ transform(y, $a_{\theta\varepsilon}$)
end for
for all <r, p> ∈ $a_{\theta r}$ **do**
  res ← res ∪ <(Ø, Ø, Ø, Ø, <r, inheritance(p) >, Ø, Ø),(Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø) >
end for
for all <r, p> ∈ $a_{\delta r}$ **do**
  res ← res ∪ <(Ø, Ø, Ø, Ø, Ø, Ø, Ø), (Ø, Ø, Ø, Ø, <r, inheritance(p) >, Ø, Ø) >
end for
let res be <($res_{\theta dom}$, $res_{\theta min}$, $res_{\theta max}$, $xres_{\theta\pi}$, $res_{\theta r,}$ $res_{\theta\varepsilon}$, $res_{\theta t}$), ($res_{\delta dom}$, $res_{\delta min}$, $res_{\delta max}$, $res_{\delta\pi}$, $res_{\delta r,}$ $res_{\delta\varepsilon,}$ $res_{\delta t)}$ >
for all x ∈ $res_{\theta\pi}$ **do**
  suppress x^o from $res_{\delta\pi}$
end for
for all x^o ∈ $res_{\theta\pi}$ **do**
  suppress x from $res_{\delta\pi}$
end for
return res

The inherited properties are the ones found in the inheritance map, the strict properties (resp. the default properties) are the ones found in the strict part (resp. Default part).

Example: *pe (resp. an, fl, mo)* are the denotation of *Penguin (resp. animal, fly, mortal)* in the above example.

pe^{inh} =< *(Univ, Min-R, Max-R,{an, mo}, Ø, Ø, Ø), (Univ, Min-R, Max-R, Ø, Ø, Ø, Ø) >*, bi^{inh} =< *(Univ, Min-R, Max-R {an, mo}, Ø, Ø, Ø), (Univ, Min-R, Max-R, fl, Ø, Ø, Ø) >*

The strict inherited properties of *pe* are *an* and the temporal property *mo*, this is shown in the first set, in the second set the field corresponding to the default part is empty since *pe* doesn't have any default properties. Intuitively, *pe* is subsumed by *bi* (i.e. a Penguin is a Bird), however *Fly* is inherited by *bi* but not by *pe* (i.e. without further information we say that a bird fly and a penguin doesn't fly).

Formally, $pe \sqsubseteq bi$ and $pe \sqsubseteq pe^{inh}$, $bi \sqsubseteq bi^{inh}$ but $pe^{inh} \not\sqsubseteq bi^{inh}$ and $bi^{inh} \not\sqsubseteq bi^{inh}$. The subsumption $pe \sqsubseteq pe^{inh}$ signifies that a concept *pe* is more specific then a concept pe^{inh} since it includes masked properties (the excepted property *Fly*) which is essential for a correct classification process (w. r. t. *bi*), but cannot be an inherited property.

3. Conclusion

In this work we presented a new notion of description logic namely the temporal nonmonotonic description logic that we formed by the combination of the nonmonotonic description logic $JClassic_{\delta\varepsilon}$ [8], [9] and the temporal component @ from $\mathcal{TL}$ [1], [2], [3]. We named this description logic $TDL_{\delta\varepsilon.}$ Our work is summarized in what follows:

- We introduced $TDL_{\delta\varepsilon}$, outlined its syntax, and provided the definition of each of its elements. We showed how this description logic is appropriate for representing temporal aspects, and illustrated this fact by examples from the domain of access control.

- We endowed $TDL_{\delta\varepsilon}$ with an algebraic framework that allows distinguishing and formalizing the descriptive and structural subsumption. In fact the descriptive point of view allows the calculation of subsumption by comparing terms trough an equational system that defines formally connector's main properties and determines the equivalence classes of terms. However From an algorithmic point of view terms are not easily manipulated through subsumption. For this we need to provide a very closer vision to the algorithmic aspect and a formal framework to validate the algorithmic approach. For this purpose we defined a structural point of view that allows representing the properties of concepts by using a normal form. Two terms of $TDL_{\delta\varepsilon}$ are structurally equivalent iff their normal forms are equal.

- We described the algorithm for subsumption calculation and the procedures it uses; this algorithm lays on the calculation of structural subsumption. We also provided elements that will allow us to show in the future that $t\text{-}sub_{\delta\varepsilon}$ is correct, complete and with a polynomial complexity.

- We define the inheritance algorithm that describes the inherited properties of a concept, which are considered to be the basis of an inferential system.

The implementation of the corresponding reasoner is in progress. This work is a preamble for our onward objective: The use of the temporal nonmonotonic description logic for enhancing access control mechanisms.

References

1. A. Artale and E. Franconi, "A survey of temporal extensions of description logics", In Ann. of Mathematics and Artificial Intelligence, 1-4, pp. 171-210, 2000.
2. A. Artale and E. Franconi, "A Temporal Description Logic for reasoning about actions and plans", In J. of Artificial Intelligence Research, 9, pp. 463-506, 1998.
3. A. Artale and E. Franconi, "Introducing Temporal Description Logics", Invited paper at the sixth International Workshop on Temporal Representation and Reasoning (TIME'99), IEEE Computer Society Press, 1999.
4. A. Artale and E. Franconi, "Temporal Description Logics", In Handbook of Time and Temporal Reasoning in Artificial Intelligence, The MIT Press, 2001.
5. A. Artale and C. Lutz, "A correspondence between temporal description logics", in: Workshop Notes of the Int. Workshop on Description Logics, DL-99, Linköping, Sweden, July 1999, pp. 145–149.
6. F. Baader and B. Hollunder, "Embedding defaults into terminological knowledge representation formalisms", In Principles of knowledge representation and reasoning: 3rd international conference, pp 306– 317, 1992.
7. F. Baader, D.L. McGuiness, D. Nardi and P.F. Schneider, "The Description logic handbook: Theory, Implementation and Applications", Cambridge university press, 2008.
8. N. Boustia and A. Mokhtari "Modeling Disjunctive Context in Access Control", In International Journal on Advances in Software, volume 5 no1& 2, 2012. N.
9. Boustia and A. Mokhtari, «A dynamic access control model». Applied Intelligence Journal, 36(1):190-207, 2012.
10. M. Bouzid, C. Combi, M. Fisher, and G. Ligozat, "Temporal Representation and Reasoning", Annals of Mathematics in Artificial Intelligence 46(3):231-234. Springer, March 2006.
11. P. Coupey and C. Fouqueré, "Extending conceptual definitions with default Knowledge", Comput Intell 13(2), 1997.
12. J. MA and B. Knight, "Reified Temporal Logics: An Overview," In journal of Artificial Intelligence Review archive, Volume 15 Issue 3, May 2001.
13. L. Padgham and B. Nebel, "Combining Classification and No Monotonic Inheritance Reasoning: a First Step", In t77th International Symposium on Methodologies for Intelligent Systems, pp. 15- 18, Norway. 1993.
14. L. Padgham and T. Zhang, "A Terminological Logic with Defaults: a Definition and an Application", In 13th International Joint Conference on Artificial Intelligence, pp. 663- 668, Chambery, France. 1993.
15. J. Quantz and V. Royer, "Preference Semantics for Defaults in Terminological Logics", In Principals of knowledge Representation and Reasoning: 3rd International Conference, pp. 294- 305, Cambridge, 1992.
16. A. Schmiedel, "A Temporal Terminological Logic", In Proc. of AAAI-90, pp. 640–645, Boston, MA (1990).