Objects as Structuring Units for Incorporating Dynamics in Deductive Conceptual Modeling

Cristina Sernadas, Paula Gouveia, Luísa Silva and Antónia Lopes

Secção de Ciência da Computação, Departamento de Matemática, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa

Following the deductive approach, the conceptual schema is a theory in a chosen logic framework whose assertions reflect the relevant static and dynamic aspects of the UoD. The problem with a flat deductive approach is the lacking of abstractions to structure the conceptual schema. Herein, we propose the object concept as the abstraction that allows the structuring of the dynamic aspects in the conceptual schema. Moreover, we recognize that interaction and inheritance between objects are new mechanisms for putting together conceptual schemata (theories).

1. Introduction

Following the *deductive approach* to conceptual modeling, the *conceptual schema* is defined as a *theory* in some logic framework which provides a set of generic axioms, valid for every conceptual schema, and a set of inference rules. In the conceptual schema we introduced new axioms representing the relevant knowledge about the UoD. The deductive approach has been followed by the so called deductive database school [Rei78,MiNi83,Rei84,GMN84,ChDe86,Oli89,DIB90] and also by some researchers of the conceptual modeling school [WMW89,FSMS90,Wie90]. The two perspectives differ basically in the adopted logic framework. In the former, the database is defined as a first-order logic theory. In the field of conceptual modeling, the conceptual schema is usually seen as a collection of formulae in a temporal/deontic logic. Among the main advantages of the deductive approach are the neat mathematical definitions, the uniform treatment of issues like querying and constraints and the possibility of proving new assertions.

In order to make the deductive approach more effective for conceptual modeling, where a flat deductive approach is very difficult because thousands of assertions have to be defined and in the end the designer would have difficulty in understanding and working with the whole conceptual schema, mechanisms for *structuring* theories are needed so that we can capture the structure of the conceptual schema. On the other hand, more complex applications, like engineering and office automation [KaLo86,BDL84], put a pressure to have mechanisms that allow us to define the conceptual schema in an *incremental* way.

Hence for adopting the deductive way of doing things, we need abstraction mechanisms for putting theories together. Moreover, in these applications, *prototyping* is becoming very useful, even at the conceptual schema level, and we would like to have the possibility of detecting inconsistencies not only for the whole conceptual schema but also for fragments. Due to the advances in the deductive paradigm it seems that the time has come to go towards *structured and incremental deductive conceptual modeling approaches.*

In [SeSe85,SeSe86,SeSe87,SeSe88,FiSe88,FSS88,SFS89,SFS90b], the *semantic primitives* of the conceptual modeling approaches are identified as one of the adequate mechanisms for *structuring* theories. In this context, the semantic primitives are theory morphisms that allow the building of a conceptual schema (a theory) from another conceptual schema (another theory). The complete conceptual schema is then a theory from which we can recapture the structure of its construction. In this research, operations, like enrichment and union, are identified for putting theories together.

However, the structuring and the incremental aims are not totally dealt with since an emphasis was put on the static side of the UoD description. According to the 100% and the conceptualization principles [Gri82] the dynamic and static aspects are of the same importance. And it is well known that a lot of work is being produced in order to tackle with dynamic aspects [Ser80,SBO85,SeSe85,Oli82,RoRi82,RBL88,SaLi89,WMW89, WiRi90].

Taking advantage of the recent effort to put the deductive and the object-oriented paradigms together [KNN89], we discuss in this paper how the object concept can be used to *structure dynamics in a deductive-oriented way* of defining the conceptual schema. The main idea is that the conceptual schema is a collection of objects, each of them seen as a theory in some logic framework, which can interact with each other. Moreover, we identify the *inheritance and interaction* abstractions as providing *new ways of putting theories* (conceptual schemata) together. We indicate how the approach can be used in the incremental definition of conceptual schemata namely illustrating useful assertions to be proved in order to get inconsistency-free conceptual schemata.

The object-oriented approach that we adopt has its roots in the work going on in the Esprit BRA IS-CORE (Information Systems: REusability and COrrectness) namely in [SeEh90,SeFi90,SFSE89a,SFSE89b,SFS90a,SSE87] for the object concept, [EhSe89, ESS88,ESS89,ESS90] for the semantic fundamentals, [FiMa90a,FiMa90b,FiSe90, FSMS90] for the calculi and [SSS90] for a model-theoretic perspective of inheritance.

2. Motivation

Objects as Theories

As an example consider a simplified *employee-department* conceptual schema. The two basic objects are of course *employee* and *department*. Adopting an incremental deductive approach we can start by defining them as theories in the following way:

```
theory employee
    importing data types natural,|DEPT|
    events
        hired(nat,nat),new-salary(nat),new-birthday,set-dept(|DEPT|),fired
    attributes
        name: string,
        dept:|DEPT|,salary,age:nat
    constraints
        (C1)   always(salary≤next(salary));
        (C2)   always(age≤next(age));
        (C3)   always((dept≠next(dept)∧salary=N)⇒sometimef(salary>N));
        (C4)   always(age=next(age)∨age+1=next(age));
        (C5)   sometimef(salary=2000)
    safety
        (S1)   always({age≥18}new-dept(D))
```

```
theory department
    importing data type |EMPLOYEE|,set(|EMPLOYEE|),natural
    events
        creation,set-budget(nat),new-employee(|EMPLOYEE|),close
    attributes
        budget:nat,employees:set(|EMPLOYEE|)
    valuation
        (V1)   always([set-budget(N)]budget=N);
        (V2)   always([new-employee(E)]employees=insert(E,employees))
    safety
        (S1)   always({number(employees<200)}new-employee(E))
```

The logic framework that we use is a mixture of a temporal, a dynamic and a safety logics over the signature of data types (note that for example *|EMPLOYEE|* is a data type that corresponds to the surrogate space of the identifications of the employees), events and attributes. The temporal logic is essentially used for expressing constraints (like *always (salary≤next (salary))* and goals (like *sometimef(salary=2000)*), the dynamic logic is used for expressing effects of the events over the attributes and the safety logic allow us to constrain the occurrence of events. For instance, the dynamic logic formula *[set-budget(N)]budget=N* informally indicates that after the occurrence of an event *set-budget(N)* the budget of the department will be *N*. On the other hand, the formula *{number(employees)<200}new-employee(E)* indicates that an employee can come to the department assuming that the previous number of employees in the department is less than 200.

At this point of the definition of the *employee-department* conceptual schema we can prove several things. For instance we can show that

$employee \vdash always((age=N) \Rightarrow alwaysf(age \geq N))$

meaning that from the axioms in theory *employee* together with the adopted logic framework we can prove the assertion above. More interesting aspects are related to inconsistencies. Assume that we want to enrich the theory *employee* with the axiom

$always([new\text{-}salary(N)]salary=salary-10)$

which makes the theory *employee* inconsistent since we also have

$always(salary \leq next(salary))$

Such inconsistency of an object in *isolation* corresponds to the non-existence of a model for such theory.

Note that so far the theories *employee* and *department* are independent and as a consequence we cannot prove anything related to the relationship between the employee and the department.

Interaction as a Mechanism for Putting Theories Together

Assume that the relationship between *employee* and *department* as is as follows:

```
interaction E:|EMPLOYEE|,D:|DEPT|
    E.set-dept(D)=D.new-employee(E)
```

meaning that the two objects synchronize when the employee is enrolled in the department. In a sense, the two events are the *same* but they are seen with different names by the two objects. As a consequence of the interaction, we can define a new conceptual schema which corresponds to *putting the two objects together taking into account the* *interaction*. In order to get a perspective of the desired operation, let us assume that we introduce a new theory, *interaction*, where we include the event

transference(|EMPLOYEE|,|DEPT|)

as well as the data types */EMPLOYEE/* and */DEPT/*. Such theory is related to the theories *employee* and *department*. The relationship is given by two *maps* (rigorously morphisms)

ϕ_1: *interaction* $\rightarrow$ *employee*
ϕ_2: *interaction* $\rightarrow$ *department*

which, in this case, are trivial since they only state

ϕ_1*(transference(E,D))=E.set-dept(D)*
ϕ_2*(transference(E,D))=D.new-employee(E)*

ie they say that the event *transference* is seen by the employee as *set-dept* and by the department as *new-employee*. The result of putting the objects together taking into account the interaction is given by aggregating (rigorously the colimit of the diagram composed by the three objects and the two morphisms), which is described by the theory *department-employee1* (acting like a *community* of two objects)

```
theory  department-employee1
    importing data types  natural,|DEPT|,|EMPLOYEE|,set(|EMPLOYEE|)
    events
        hired(nat,nat),new-salary(nat),new-birthday,set-budget(nat),
        transference(|EMPLOYEE|,|DEPT|),fired,creation,close
    attributes
     employees:set(|EMPLOYEE|),dept:|DEPT|,budget,salary,age:nat,name:string
    constraints
        (C1)    always(salary≤next(salary));
        (C2)    always(age≤next(age));
        (C3)    always((dept≠next(dept)∧salary=N)⇒sometimef(salary>N));
        (C4)    always(age=next(age)∨age+1=next(age));
        (C5)    sometimef(salary=2000)
    safety
        (S1)    always({age≥18∧number(employees)<200}transference(E,D))
    valuation
        (V1)    always([set-budget(N)]budget=N);
        (V2)    always([transference(E,D)]employees=insert(E,employees))
```

and two theory morphisms

φ_1*:employee* $\rightarrow$ *department-employee1*
φ_2*:department* $\rightarrow$ *department-employee1*

such that

$\varphi_1(E.set\text{-}dept(D)) = transference(E,D)$
$\varphi_2(D.new\text{-}employee(E)) = transference(E,D)$

according to the following diagram

$$\begin{tikzcd}
& \textit{employee} \arrow[dr, "\varphi_1"] & \\
\textit{interaction} \arrow[ur, "\phi_1"] \arrow[dr, "\phi_2"'] & & \textit{department-employee1} \\
& \textit{department} \arrow[ur, "\varphi_2"'] &
\end{tikzcd}$$

Note that the event *transference* is more constrained than the corresponding event in both *employee* and *department* when considered in isolation. It can only take place if $age \geq 18$ (which comes from *employee*) and $number(employees) < 200$ (which comes from *department*).

The theory morphisms φ_1 and φ_2 are more complex since they are not only changes of vocabulary. We want them to provide a *safe import mechanism* in the sense that we want to identify in it the sub-theories *employee* and *department*, ie an employee in department-employee is still an employee and a department in department-employee is still a department. Namely, we want the assertions for the isolated employee and department to be assertions when they are in the community *department-employee1*, that the events of *departament* do not change the values of the attributes of *employee* and vice-versa.

When the objects interact the inconsistency problem that we can get is not with the attributes, since they are not involved, but with the events. As an example, assume that in in object obj_1 we can prove for an event e_1

$obj_1 \vdash sometimef(after(e_1))$

Assume that we define the following interaction:

interaction obj_1, obj_2 $\qquad e_1 = e_2\ ,\ e_1' = e_2'$

and that we have the following safety axioms

$\{sometimep(after(e_1'))\}\ e_1$
$\{sometimep(after(e_2))\}\ e_2'$

Let *obj* be the object that results from putting obj_1 and obj_2 together taking into account the interaction. Let *e*, *e'* be the events in *obj* that correspond in obj_1 to e_1 and e_1' and in obj_2 to e_2 and e_2'. Note that the events *e* and *e'* cannot happen in *obj* and as a consequence we cannot prove the assertion above.

Inheritance as a Mechanism for Putting Theories Together

Assume now that we define the relationship between *employee* and *department* through inheritance in the following way:

```
theory department-employee2
    inheriting employee into department
```

As a consequence of such inheritance, we can define a new conceptual schema which corresponds to *putting the objects employee* and *department together* taking into account the inheritance. The result is the theory *department-employee2*, containing all the axioms of *employee*, *department* plus the map (morphism)

φ: *employee* $\rightarrow$ *department-employee2*

as a formula, see below, in *department-employee2*. Hence, it makes sense to say whether or not such a formula is satisfied. Note that if we add the axiom

always([set-budget(N)]salary=salary+100)

we do not have the desired relationship between *employee* and *department* since the formula above is no longer an assertion in *department-employee2*. The intuitive idea is that we have the event *set-budget(N)* of *department* changing the attribute *salary* of *employee*. This would mean that *employee* is not an object (informally a collection of attributes and all the events that can change the attributes).

3. Logical and Categorial Setting

The main concepts to be introduced are the theory (an object) and the theory morphism for putting theories together. Intuitively, a theory is a signature (a vocabulary) and a set of assertions whereas a theory morphism is a map between theories. A logic framework must be adopted in order to write assertions and a category must be defined in order to have theory morphisms and the possibility of putting objects together.

Definition 3.1 *Object signature* Σ_{obj}

An object signature Σ_{obj} is a quadruple $(S\cup\{e\}, OP, EVT, ATT)$ where

- S is a set of data sorts and $\{e\}$ is a single set with an event sort e;
- OP is a $S^*\times S$-indexed family of sets of operation symbols;
- EVT is a $S^*\times\{e\}$-indexed family of sets of event symbols;
- ATT is a $S^*\times S$-indexed family of sets of attribute symbols. □

Example:
In the theory *employee* we have
 nat $\in S$, *new-salary(nat)* $\in EVT_{nat,e}$, *salary* $\in ATT_{nat}$ □

We must be able to interpret the symbols in the signature. For this purpose we introduce the concept of model.

Definition 3.2 Σ_{obj}-*Model* M for the object signature Σ_{obj}

A Σ_{obj}-model M is a triple $(\mathcal{B},\Gamma,\vartheta)$ where

- $\mathcal{B}$ is a $S\cup\{e\}$-indexed family of sets called carriers of the sorts;
- Γ is a subset of $\mathcal{B}_e{}^{\sigma}$ whose elements are called life cycles;
- ϑ is a $S^*\times(S\cup\{e\})$-indexed family of maps
 $\{\vartheta_{s_1\ldots s_n,s}:H_{s_1\ldots s_n,r}\rightarrow$
 $[\Gamma\rightarrow[2^{[\mathbb{N}_\circ\rightarrow\mathcal{B}_{s_1}]}\times\ldots\times 2^{[\mathbb{N}_\circ\rightarrow\mathcal{B}_{s_n}]}\rightarrow 2^{[\mathbb{N}_\circ\rightarrow\mathcal{B}_{s_r}]}]\}_{<s_1\ldots s_n>\in S^*,r\in S\cup\{e\}}$
 where $H_{s_1\ldots s_n,r}=OP_{s_1\ldots s_n,s}\cup EVT_{s_1\ldots s_n,e}\cup ATT_{s_1\ldots s_n,s}$ □

Example:
A $\Sigma_{employee}$-model $M_{employee}$ is the following
 $\mathcal{B}_{nat}=\mathbb{N}_\circ$, $\mathcal{B}_{IDEPTI}=\mathbb{N}_\circ$,
 $\mathcal{B}_e=\{$*hired(i,j),new-salary(i),new-birthday,set-dept(i),fired,i,j* $\in\mathbb{N}_\circ\}$
 $\Gamma=\{\eta=<$*hired(i,j),new-salary(n),set-dept(m),fired*$>$,*i,j,m,n*$\in\mathbb{N}_\circ\}$

$\vartheta_{nat}(salary)(\eta)=\{f:f(1)=i,f(2)=n,f(3)=n,f(4)=n,f(i)=\mathit{undefined},i\geq 5\}$

□

Before introducing the formulae we must start by introducing the terms of data and event sort.

Definition 3.3 *Sets of terms*

Let X be a S-indexed set of variables. T(X) is a S∪{e}-indexed family of sets of terms defined inductively as follows:

- every variable $x\in X_s$ is a term of data sort s
- if $g\in EVT_{s_1,\dots,s_n,e}$ and $t_i\in T_{s_i}(X)$, i=1,...,n then $g(t_1,\dots,t_n)\in T_e(X)$
- if $o\in OP_{s_1,\dots,s_n,s}$ and $t_i\in T_{s_i}(X)$, i=1,...,n then $o(t_1,\dots,t_n)\in T_s(X)$
- if $a\in ATT_{s_1,\dots,s_n,s}$ and $t_i\in T_{s_i}(X)$, i=1,...,n then $a(t_1,\dots,t_n)\in T_s(X)$
- if $t\in T_s(X)$ and $u\in T_e(X)$ then $[u]t\in T_s(X)$
- if $t\in T_s(X)$ then $\mathit{next}(t)\in T_s(X)$ □

Example:

next(salary),budget $\in T_{nat}(X)$

set-budget(N) $\in T_e(X)$

[set-budget(N)]budget $\in T_{nat}(X)$ □

We will consider $\mathcal{B}_e=\{g(t_1,\dots,t_n),g\in EVT_{s_1\dots s_n,e},t_i\in T_{s_i}(X),i=1,\dots,n\}$ as the carrier for the event symbols. We must now define the interpretation of the terms.

Definition 3.4 *Non-deterministic interpretations of terms*

Assume that M is a Σ_{obj}-model, X is a S-indexed family of variables of data sort, ρ is an assignment, ie a S-indexed family of maps $\rho_s:X_s\rightarrow 2^{\mathcal{B}_s}$ and Ω the set of all assignments. The interpretation I of the terms is a S∪{e}-indexed family of maps

$$I_r:T_r(X)\rightarrow[\Omega\rightarrow[\Gamma\rightarrow[\mathbb{N}_*\rightarrow 2^{\mathcal{B}_r}]]]$$

such that

- $I_s(x)(\rho)(\eta)(k)=\rho_s(x)$
- Let

  $F=\vartheta_{s_1\dots s_n,r}(c)(\eta)(F_1,\dots,F_n)$ $F_i=\{f_i:f_i(k)\in I_{s_i}(t_i)(\rho)(\eta)(k)\}$

  Then

$I_s(c(t_1,\ldots,t_n))(\rho)(\eta)(k)$
$\quad = \{f(k): f\in F\}$ if F is *defined*
$\quad =$ *undefined* if F is *undefined*

- $I_s([u]t)(\rho)(\eta)(k)$
  $\quad = I_s(t)(\rho)(\eta)(k+1)$ if $\eta_k\in I_e(u)(\rho)(\eta)(k)$
  $\quad =$ *undefined* otherwise
- $I_s(\mathit{next}(t))(\rho)(\eta)(k)=I_s(t)(\rho)(\eta)(k+1)$

where

$x\in X_s$, $k\in \mathbb{N}_0$, η_k is the k-element of η
$t_i\in T_{s_i}(X)$, $i=1,\ldots,n$, $t\in T_s(X)$, $u\in T_e(X)$
$c\in OP_{s_1,\ldots,s_n,s}\cup EVT_{s_1,\ldots,s_n,e}\cup ATT_{s_1,\ldots,s_n,s}$
$s_1,\ldots,s_n,s\in S$, $r\in S\cup\{e\}$ □

Note that the terms are interpreted as sets of values in the carrier set of the respective sort.

Example:
$I_{nat}(\mathit{next}(\mathit{salary}))(\rho)(\eta)(0)=I_{nat}(\mathit{salary})(\rho)(\eta)(1)=\{i\}$
$I_{nat}(\mathit{next}(\mathit{salary}))(\rho)(\eta)(1)=I_{nat}(\mathit{salary})(\rho)(\eta)(2)=\{n\}$
$I_{nat}(\mathit{next}(\mathit{salary}))(\rho)(\eta)(2)=I_{nat}(\mathit{salary})(\rho)(\eta)(3)=\{n\}$
$I_{nat}(\mathit{next}(\mathit{salary}))(\rho)(\eta)(3)=I_{nat}(\mathit{salary})(\rho)(\eta)(4)=\{n\}$
$I_{nat}(\mathit{next}(\mathit{salary}))(\rho)(\eta)(i)=I_{nat}(\mathit{salary})(\rho)(\eta)(i+1)=\mathit{undefined}, i\geq 4$ □

Definition 3.5 *Formulae*

The set of formulae F(X) over an object signature Σ_{obj} is defined inductively as follows:

- if $t_1,t_2\in T_s(X)$ then $t_1=t_2$, $t_1<t_2$, $t_1>t_2$ are formulae;
- if $u\in T_e(X)$ then *after*(u) is a formula;
- if $\mathcal{C}$ is a formula then $\mathit{sometimep}(\mathcal{C})$, $\mathit{sometimef}(\mathcal{C})$, $[u]\mathcal{C}$, $\mathit{next}(\mathcal{C})$ are formulae;
- if $\mathcal{C}$ and $\mathcal{D}$ are formulae then $\neg\mathcal{C}$, $\mathcal{C}\vee\mathcal{D}$ are also formulae;
- if $\mathcal{C}$ is a formula and $u\in T_e(X)$ then $\{\mathcal{C}\}u$ is a formula. □

Example:
$\mathit{always}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))$ is a formula where
$\mathit{always}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))=$
$\quad\mathit{alwaysf}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))\wedge\mathit{salary}\leq\mathit{next}(\mathit{salary})\wedge\mathit{alwaysp}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))$
$\mathit{alwaysf}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))=\neg(\mathit{sometimef}(\neg(\mathit{salary}\leq\mathit{next}(\mathit{salary}))))$
$\mathit{alwaysp}(\mathit{salary}\leq\mathit{next}(\mathit{salary}))=\neg(\mathit{sometimep}(\neg(\mathit{salary}\leq\mathit{next}(\mathit{salary}))))$ □

Definition 3.6 *Non-deterministic satisfaction of formulae*

Assume that M is a model for a signature Σ_{obj}. M satisfies the formula

- $t_1=t_2$ in $k\in \mathbb{N}_\circ$, $t_1,t_2\in T_s(X)$, if for every $\rho\in\Omega$, $\eta\in\Gamma$
  $\mathcal{I}_s(t_1)(\rho)(\eta)(k)=\mathcal{I}_s(t_2)(\rho)(\eta)(k)=\{b\}$ where $b\in \mathcal{B}_s$
- $t_1<t_2$ in $k\in \mathbb{N}_\circ$, $t_1,t_2\in T_s(X)$, if for every $\rho\in\Omega$, $\eta\in\Gamma$
  $\max(\mathcal{I}_s(t_1)(\rho)(\eta)(k))<\min(\mathcal{I}_s(t_2)(\rho)(\eta)(k))$
- $t_1>t_2$ in $k\in \mathbb{N}_\circ$, $t_1,t_2\in T_s(X)$, if for every $\rho\in\Omega$, $\eta\in\Gamma$
  $\min(\mathcal{I}_s(t_1)(\rho)(\eta)(k))>\max(\mathcal{I}_s(t_2)(\rho)(\eta)(k))$
- *after*(u) in $k\in \mathbb{N}_\circ^+$, $u\in T_e(X)$, if for every $\rho\in\Omega$, $\eta\in\Gamma$
  $\eta_{k-1}\in \mathcal{I}_e(u)(\rho)(\eta)(k-1)$
- *sometimep*($\mathfrak{C}$) in $k\in \mathbb{N}_\circ^+$, $\mathfrak{C}$ is a formula, if for every $\rho\in\Omega$, $\eta\in\Gamma$ there is
  $i<k$, $i\in \mathbb{N}_\circ$, such that M satisfies $\mathfrak{C}$ in i
- *sometimef*($\mathfrak{C}$) in $k\in \mathbb{N}_\circ$, $\mathfrak{C}$ is a formula, if for every $\rho\in\Omega$, $\eta\in\Gamma$ there is
  $i>k$, $i\in \mathbb{N}_\circ$, such that M satisfies $\mathfrak{C}$ in i
- $[u]\mathfrak{C}$ in $k\in \mathbb{N}_\circ$, $u\in T_e(X)$, $\mathfrak{C}$ is a formula,
  if M satisfies $\mathfrak{C}$ in k+1 providing that $\eta_k\in \mathcal{I}_e(u)(\rho)(\eta)(k)$, for every $\rho\in\Omega$, $\eta\in\Gamma$
- *next*($\mathfrak{C}$) in $k\in \mathbb{N}_\circ$, $\mathfrak{C}$ is a formula,
  if M satisfies $\mathfrak{C}$ in k+1
- $\neg\mathfrak{C}$ in $k\in \mathbb{N}_\circ$, $\mathfrak{C}$ is a formula,
  if M does not satisfy $\mathfrak{C}$ in k
- $\mathfrak{C}\vee\mathfrak{D}$ in $k\in \mathbb{N}_\circ$, $\mathfrak{C}$, $\mathfrak{D}$ are formulae,
  if M satisfies either $\mathfrak{C}$ or $\mathfrak{D}$ in k
- $\{\mathfrak{C}\}u$ in $k\in \mathbb{N}_\circ^+$, $u\in T_e(X)$, $\mathfrak{C}$ is a formula, for every $\rho\in\Omega$, $\eta\in\Gamma$
  if $\eta_k\in \mathcal{I}_e(u)(\rho)(\eta)(k)$ then M satisfies $\mathfrak{C}$ in k-1.

We denote by $M\models\mathfrak{C}$ the satisfaction of $\mathfrak{C}$ by M. □

Example: Consistency of *always(salary<next(salary)∧*
[new-salary(N)]salary=salary-10)

Let M be a model for the signature Σ_{obj}. Let *salary* $\in T_s(X)$, $\rho\in\Omega$, $\eta\in\Gamma$.

Assume that $\mathcal{I}_s(salary)(\rho)(\eta)=\mathfrak{f}$

(i) M satisfies *always(salary≤next(salary)∧*
[new-salary(N)]salary=salary-10) iff for every $k\in \mathbb{N}_\circ$

M satisfies $\mathit{salary} \leq \mathit{next}(\mathit{salary})$ in k
and M satisfies $[\mathit{new\text{-}salary}(N)]\mathit{salary}=\mathit{salary}-10$ in k

(ii) M satisfies $\mathit{salary} \leq \mathit{next}(\mathit{salary})$ in k
iff M satisfies $(\mathit{salary} < \mathit{next}(\mathit{salary}) \vee \mathit{salary} = \mathit{next}(\mathit{salary}))$ in k
iff M satisfies either $\mathit{salary} < \mathit{next}(\mathit{salary})$
or $\mathit{salary} = \mathit{next}(\mathit{salary})$ in k

(iii) M satisfies $\mathit{salary} < \mathit{next}(\mathit{salary})$ in k iff
$\max(f(k)) < \min(f(k+1))$ (*)

(iv) M satisfies $[\mathit{new\text{-}salary}(N)]\mathit{salary}=\mathit{salary}-10$ in k iff
$I_s([\mathit{new\text{-}salary}(N)]\mathit{salary})(\rho)(\eta)(k) = I_s(\mathit{salary}-10)(\rho)(\eta)(k)$ iff
$I_s(\mathit{salary})(\rho)(\eta)(k+1) = I_s(\mathit{salary}(\rho)(\eta)(k)-10$ (**)
if $\eta_k \in I_e(\mathit{new\text{-}salary}(N))(\rho)(\eta)(k)$

(v) M satisfies $\mathit{always}(\mathit{salary} < \mathit{next}(\mathit{salary}) \wedge [\mathit{new\text{-}salary}(N)]\mathit{salary}=\mathit{salary}-10)$
assuming that $\eta_k \in I_e(\mathit{new\text{-}salary}(N))(\rho)(\eta)(k)$
Taking into account (**) we have that (*) becomes $f(k) < (f(k+1)$ (***)
Consider k=1 in (**). We have $f(2) = f(1)-10$ (****)
Combining (***) and (****) for k=1 we have $f(1) < f(1)-10$
which is impossible.

(vi) In a similar way we can prove a similar result for the satisfaction of
$\mathit{salary} = \mathit{next}(\mathit{salary})$ in k.

Hence there is *at least one* model that does
not satisfy
$\mathit{always}(\mathit{salary} < \mathit{next}(\mathit{salary}) \wedge [\mathit{new\text{-}salary}(N)]\mathit{salary}=\mathit{salary}-10)$. □

Definition 3.7 *Object*

An object *obj* is a pair $(\Sigma_{obj}, \mathcal{F})$ where

- Σ_{obj} is an object signature
- $\mathcal{F}$ is a set of formulae □

Definition 3.8 *Consistency.*

An object $obj=(\Sigma_{obj},\mathcal{F})$ is consistent if for each model M we have

$M\models\mathfrak{C}$ for every $\mathfrak{C}\in\mathcal{F}$ □

Example:

employee is a consistent object □

We now must introduce the concept of morphism between theories so that we can define the mechanisms for inheritance and interaction. The morphism that we use is similar to the one presented in [FiMa90a] but based on another model concept. We start by defining what is an object signature morphism.

Definition 3.9 *Signature morphism*

An object signature morphism $\sigma:\Sigma_{obj}\rightarrow\Sigma_{obj}'$ is a pair (α,β) where

- $\alpha:S\cup\{e\}\rightarrow S'\cup\{e\}$;
- β is a $S^*\times(S\cup\{e\})$-indexed family of maps $H_{s_1\ldots s_n,r}\rightarrow H'_{s_1'\ldots s_n',r'}$ where
  $H_{s_1\ldots s_n,r}=OP_{s_1\ldots s_n,s}\cup EVT_{s_1\ldots s_n,e}\cup ATT_{s_1\ldots s_n,s}$,
  $H'_{s_1'\ldots s_n',r'}=OP'_{s_1'\ldots s_n',r'}\cup EVT'_{s_1'\ldots s_n',r'}\cup ATT'_{s_1'\ldots s_n',r'}$ such that
  $\beta(c)=c'$, $c\in H_{s_1\ldots s_n,r}$ and $c'\in H'_{\alpha(s_1),\ldots,\alpha(s_n),\alpha(r)}$ □

Definition 3.10 *Morphism between objects*

Let *obj* and *obj'* be objects. A **morphism** $\sigma:obj\rightarrow obj'$ exists between the objects *obj* and *obj'* iff there is a signature morphism from the signature of *obj* into the signature of *obj'* and for every model of *obj'* there is a model for *obj* such that:

$\mathcal{B}_{obj,s}=\mathcal{B}_{obj',s}$ for $s\in S$

$\mathcal{B}_{obj,e}=\mathcal{B}_{obj',e}\downarrow EVT_{s_1\ldots s_n,e}{}^{r}$

where $EVT_{s_1\ldots s_n,e}{}^{r}=\{g(b_1,\ldots,b_n):b_i\in\mathcal{B}_{obj,s_i},i=1,\ldots,n,g\in EVT_{s_1\ldots s_n,e}\}$

$\Gamma_{obj}=\Gamma_{obj'}\downarrow\mathcal{B}_{obj,e}$

$\vartheta_{obj,s_1\ldots s_n,s}(c)(\eta_{obj})(F_1,\ldots,F_n)(k)=\vartheta_{obj',s_1\ldots s_n,s}(c)(\eta_{obj'})(F_1,\ldots,F_n)(k)$

where $\eta_{obj}=\eta_{obj'}\downarrow\mathcal{B}_{obj,e}$

$c\in OP_{obj,s_1,\ldots,s_n,s}\cup EVT_{obj,s_1,\ldots,s_n,e}\cup ATT_{obj,s_1,\ldots,s_n,s}$. □

Example:

A $\Sigma_{department\text{-}employee2}$-model $M_{department\text{-}employee2}$ is the following

$\mathcal{B}_{nat}=\mathbb{N}_0$, $\mathcal{B}_{|DEPT|}=\mathbb{N}_0$, $\mathcal{B}_{|EMPLOYEE|}=\mathbb{N}_0$, $\mathcal{B}_{set(|EMPLOYEE|)}=2^{\mathbb{N}}$

$\mathcal{B}_{department\text{-}employee2,e}=\{$*hired(i,j),new-salary(n),new-birthday,set-dept(m),fired, creation,set-budget(p),new-employee(q),close*,i,j,n,m,p,q$\in\mathbb{N}_0\}$

$\Gamma_{department\text{-}employee2}=\{\eta_{department\text{-}employee2}=$ *<creation,hired(i,j),new-salary(n),set-dept(m),set-budget(p),fired>*$\}$

such that $\eta_{department\text{-}employee2}=<\eta_0,\eta_1,\eta_2,\eta_3,\eta_4,\eta_5>$

$\vartheta_{department\text{-}employee2,nat}(salary)(\eta_{department\text{-}employee2})=$

$\{f:f(2)=i,f(3)=n,f(4)=n*1.1,f(5)=n*1.3,f(i)=undefined,i\geq6\}$

There is a morphism

$\sigma:M_{employee}\rightarrow M_{department\text{-}employee2}$

since in particular

$\Gamma_{employee}=\{\eta_{employee}=$

<hired(i,j),new-salary(n),set-dept(m),fired>$\}$

such that $\eta_{employee}=<\eta_1,\eta_2,\eta_3,\eta_5>$

$\vartheta_{employee,nat}(salary)(\eta_{employee})(k)=$

$\vartheta_{department\text{-}employee2,nat}(salary)(\eta_{department\text{-}employee2})(k)$

Note that when we take the restriction we get the model $M_{employee}$.

In other words, we can say that the model $M_{department\text{-}employee2}$ of *department-employee2* satisfies the formula $\sigma:M_{employee}\rightarrow M_{department\text{-}employee2}$. □

Example:

The putting together of *employee* and *department* through the *interaction* is (*department-employee1*,

φ_1:*employee* → *department-employee1*,φ_2:*department* → *department-employee1*)=

pushout$_{object}$(ϕ_1:*interaction* → *employee*,ϕ_2:*interaction* → *department*)

The putting together of *employee* and *department* through the *inheritance* is (*department-employee2*,φ:*employee* → *department-employee2*)

4. Conclusions

The *object* concept is introduced as the abstraction that allows the *structuring of dynamics* in deductive conceptual modeling. Hence the conceptual schema is described as a collection of objects (theories) that can interact through events. In each object (corresponding to the objects in the UoD) we include the relevant events and attributes, as well as formulae stating constraints on the values of the attributes, effects of the events upon the attributes and constraints upon the occurrence of the events. For this purpose, we use a logic framework which is a mixture between a temporal logic, a dynamic logic and a safety logic. Examples of assertions that can be proved are presented namely the ones related to the inconsistency of an object in isolation.

Sharing of events and *inheritance* are recognized as mechanisms for *putting theories together* and are explained through colimits in a category of objects. Again examples are presented of inconsistency problems with the objects when considered in the community.

Acknowledgments

The work reported herein was partially supported by Junta Nacional de Investigação Científica e Tecnológica through the research project OBCALC (Logics of Objects).

References

[BDL84]
Batini, C., Demo, B. and Di Leva, "A Methodology for Conceptual Design of Office Data Bases", *Information Systems*, 9[3], 1984

[ChDe86]
Cholvy, L. and Demolombe, R., "Querying a Rule Base", *First Conference on Expert Database Systems*, Charleston, 1986

[DIB90]
Demolombe, R., Illarramendi, A. and Blanco, J.-M., "Semantic Optimization in Data Bases Using Artificial Intelligence Techniques", R. Meersman, Shi, Z. and C. Kung (eds), *The Role of Artificial Intelligence in Databases and Information Systems*, North-Holland, 1990, 519-528

[EhSe89]
Ehrich, H.-D. and Sernadas, A., "Algebraic Implementation of Objects over Objects", *REX89: Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, Springer Verlag, to be published

[ESS88]
Ehrich, H.-D., Sernadas, A. and Sernadas, C., "Abstract Object Types for Databases", *Advances in Object-Oriented Database Systems*, K. Dittrich (ed), *Advances in Object-oriented Database Systems*, Springer Verlag, 1988

[ESS89]
Ehrich, H.-D., Sernadas, A. and Sernadas, C.,, "Objects, Object Types and Object Identity", *Categorical Methods in Computer Science with Aspects from Topology*, H. Ehrig et al, Springer Verlag, 1989, 142-156

[ESS90]
Ehrich, H.-D., Sernadas, A. and Sèrnadas, C., "From Data Types to Object Types", *Journal of Information Processing and Cybernetics*, EIK 26(1/2), 1990, 33-48

[FiMa90a]
Fiadeiro, J. and Maibaum, T., "Temporal Reasoning Over Deontic Specifications", *Journal of Logic and Computation*, to be published

[FiMa90b]
Fiadeiro, J. and Maibaum, T., "Describing, Structuring and Implementing Objects", *REX90: Foundations of Object-oriented Languages*, Springer-Verlag, to be published

[FiSe88]
Fiadeiro, J. and Sernadas, A., "Structuring Theories on Consequence", D. Sanella and A. Tarlecki (eds), *Recent Trends in Data Type Specification*, Springer-Verlag, 1988, 44-72

[FiSe90]
Fiadeiro, J. and Sernadas, A., "Logics of Modal Terms for Systems Specification", *Journal of Logic and Computation*, to be published

[FSMS90]
Fiadeiro, J., Sernadas, C., Maibaum, T. and Saake, G., "Proof-theoretic Semantics of Object-oriented Specification Constructs", R. Meersman and B. Kent (eds), *Object-oriented Databases: Analysis, Design and Construction*, North-Holland, to be published

[FSS88]
Fiadeiro, J., Sernadas, A. and Sernadas, C., K. Nori (ed), "Knowledge Bases as Structured Theories", *Foundations of Software Technology and Theoretical Computer Science*, Springer-Verlag, 1988, 469-486

[GMN84]
Gallaire, H., Minker, J. and Nicolas, J.-M., "Logic Databases: A Deductive Approach", *Computing Surveys*, 16[2], 1984, 153-185

[Gri82]
Van Griethuysen, J. (ed) *Concepts and Terminology for the Conceptual Schema and the Information Base*, ISO/TC97/SC5-N695, 1982

[KaLo86]
Karl, S. and Lockermann., "Design of Engineering Databases: A Case for More Varied Semantic Modeling Concepts", *Information Systems*, 13(4), 1988, 335-258

[KNN89]
First International Conference on Deductive and Object-oriented Databases, W. Kim, J.-M. Nicolas and S. Nishio (eds), 1989

[MiNi83]
Minker, J. and Nicholas, J.-M., "On Recursive Axioms in Deductive Databases", *Information Systems*, 8[1], 1983

[Oli82]
Olivé, A., "Dades: A Methodology for Specification and Design of Information Systems", Olle, W., Sol, H. and Verrijn-Stuart, A., *Information Systems Design Methodologies: A Comparative Review*, North-Holland, 1982, 285-334

[Oli89]
Olivé, A., *Deductive Conceptual Modeling*, IFIP WG8.1 Meeting, Sesimbra, June, 1989

[RBL88]
Rolland, C., Bodart, F. and Leonard, M. (eds), *Temporal Aspects of Information Systems*, North-Holland, 1988

[Rei78]
Reiter, R., "On Closed World Data Bases", H. Gallaire and J. Minker (eds), *Logic and Databases*, Plenum Press, 1978, 55-76

[Rei84]
Reiter, R., "Towards a Logical Reconstruction of Relational Database Theory", M. Brodie, J. Mylopoulos and J. Schmidt (eds), *On Conceptual Modeling*, Springer-Verlag, 1984, 191-233

[RoRi82]
Rolland, C. and Richard, C., "The Remora Methodology for Information Systems Design and Management", W. Olle, H. Sol and A. Verrijn-Stuart (eds), *Information Systems Design Methodologies: A Comparative Review*, North-Holland, 1982, 369-426

[SaLi89]
Saake, G. and Lipeck, U., "Using Finite-Linear Temporal Logic for Specifying Database Dynamics", *Proc. CSL'88 2nd Workshop Computer Science Logic*, Borger, E. Kleine Buening, H. and Richter, M., LNCS 385, Springer Verlag, 1989, 288-300

[SBO85]
Sernadas, A., Bubenko, J. and Olivé, A. (eds), *Information Systems: Theoretical and Formal Aspects*, North-Holland, 1985

[SeEh90]
Sernadas, A. and Ehrich, H.-D., "What is an Object, After All", *Object-oriented Databases: Analysis, Design and Construction*, R. Meersman and Kent, B. (eds), North-Holland, to be published

[SeFi90]
Sernadas, C. and Fiadeiro, J., *Towards Object-oriented Conceptual Modeling*, INESC Research Report, 1990

[Ser80]
Sernadas, A., "Temporal Aspects of Logical Procedure Definition", *Information Systems* 5, 1980, 167-187

[SeSe85]
Sernadas, A. and Sernadas, C., "The Use of E-R Abstractions for Knowledge Representation", P. Chen (ed), *Entity-Relationship Approach: The Use of ER Concept in Knowledge Representation*, North-Holland, 1985, 224-231

[SeSe86]
Sernadas, C. and Sernadas, A., "Conceptual Modeling Abstraction Mechanisms as Parameterized Theories in Institutions", T. Steel and R. Meersman (eds), *Database Semantics*, North-Holland, 1986, 121-140

[SeSe87]
Sernadas, A. and Sernadas, C. , "Conceptual Modeling for Knowledge-Based DSS Development", C. Holsapple and A. Whinston (eds), *Decision Support Systems: Theory and Application*, Springer-Verlag, 1987, 91-135

[SeSe88]
Sernadas, A. and Sernadas, C. "Abstraction and Inference Mechanisms for Knowledge Representation", J. Schmidt and C. Thanos (eds), *Foundations of Knowledge-Based Management*, Springer-Verlag, 1988, 91-111

[SFS89]
Sernadas, C., Fiadeiro, J., Sernadas and A., "Proof-theoretic Conceptual Modeling: The Niam Case Study", *Information System Concepts: An In-depth Analysis*, Falkenberg, E. and Lindgreen, P. (eds), North Holland, 1989, 1-30

[SFS90a]
Sernadas, C., Fiadeiro, J. and Sernàdas, A., Object-oriented Conceptual Modeling from Law, R. Meersman, Shi, Z. and C. Kung (eds), *The Role of Artificial Intelligence in Databases and Information Systems*, North-Holland, 1990, 305-327

[SFS90b]
Sernadas, C., Fiadeiro, J., Sernadas and A., "Modular Construction of Logic Knowledge Bases: An Algebraic Approach", *Information Systems*, 15[1], 1990, 37-59

[SFSE89a]
Sernadas, A., Fiadeiro, J., Sernadas, C. and Ehrich, H.-D., "The Basic Building Blocks of Information Systems", *Information System Concepts: An In-depth Analysis*, Falkenberg, E. and Lindgreen, P. (eds), North Holland, 1989, 225-246

[SFSE89b]
Sernadas, A., Fiadeiro, J., Sernadas, C. and Ehrich, H.-D.,"Abstract Object Types: A Temporal Perspective", *Temporal Logic in Specification*, Banieqbal, B., Barringer, H. and Pnueli, A. (eds), Springer Verlag, 1989, 324-350

[SSE87]
Sernadas, A., Sernadas, C. and Ehrich, H.-D., "Object-Oriented Specification of Databases: An Algebraic Approach", *Proc. 13th Conference on Very Large Data Bases*, VLDB, Hammersley, P. (ed), 1987, 107-116

[SSS90]
Sernadas, C., Saake, G. and Sernadas, A., *Algebraic Approach to Inheritance*, INESC Research Report, 1990

[Wie90]
Wieringa, R., "Equational Specification of Dynamic Objects" , R. Meersman and B. Kent (eds), *Object-oriented Databases: Analysis, Design and Construction*, North-Holland, to be published

[WiRi90]
Wieringa, R, and van de Riet, R, "Algebraic Specification of Object Dynamics in Knowledge Base Domains", R. Meersman, Shi, Z. and C. Kung (eds), *The Role of Artificial Intelligence in Databases and Information Systems*, North-Holland, 1990, 411-436

[WMW89]
Wieringa, R., Meyer, J.-J. and Weigand, H., "Specifying Dynamics and Deontic Integrity Constraints", *Data and Knowledge Engineering* 4(2), 1989, 157-190