Navigating through Multiple Temporal Granularity Objects

Elisa Bertino[1] Elena Ferrari[1] Giovanna Guerrini[2] Isabella Merlo[1]

[1]DSI - Università degli Studi di Milano - Italy
{bertino,ferrarie,merloisa}@dsi.unimi.it

[2]DISI - Università di Genova - Italy
guerrini@disi.unige.it

Abstract

Managing and relating temporal information at different time units is an important issue in many applications and research areas, among them temporal object-oriented databases. Due to the semantic richness of the object-oriented data model, the introduction of multiple temporal granularities in such a model poses several interesting issues. In particular, object-oriented query languages provide a navigational approach to data access, performed via path expressions. In this paper we present an extension to path expressions to a multi-granularity context. The syntax and semantics of the proposed path expressions are formally defined.

1 Introduction

Managing and relating temporal information at different time units is an important issue in many applications and research areas, among them temporal databases. Some interesting issues arise in extending a database model to store and query data with *multiple temporal granularities* [3].

In particular, the introduction of multiple temporal granularities in an object-oriented data model poses several interesting issues, due to the semantic richness of such a model. Most of the temporal object-oriented data models proposed so far do not deal with temporal granularities. The only ones dealing with temporal granularities [5, 12, 13, 15] support multiple temporal granularities as extensions to the set of types of the temporal model. However, the specification and management of different granularities, e.g., how to convert from a granularity to another, is completely left to the user.[1]

Object-oriented database systems provide, in addition to traditional query languages, a *navigational* approach to data access. Given an oid, the system directly accesses the corresponding object and navigates through objects referred to by its components. This access modality can be combined with the query-based (e.g., SQL-like) access. Thus, conditions in a query can be imposed on the nested properties, denoted by *path expressions*, of the queried objects.Most of the temporal object-oriented query languages do not consider navigational acces to data .

Temporal path expressions [1, 7] are obtained as an extension of classical path expressions of object-oriented languages, in that for each property access a time can be specified, in addition to the property name. In this paper, the notion of temporal path expression is extended to a multi-granularity context. Thus, for each property access a set of granules has to be specified in order to access the property. A specific set of granules can be denoted either *explicitly* or *implicitly*. To explicitly denote a set of granules we refer to the notion of *temporal element*. To symbolically denote a set of granules we use *temporal expression*, which extends the notion of temporal expression presented in [1], which has been in turn inspired by the one proposed by Gadia and Nair in [7].

The paper is organized as follows. In Section 2 the temporal multi-granularity reference object model is presented. Section 3 presents how multi-granularity temporal values can be combined and compared, and the syntax and semantics of temporal expressions. Section 4 presents temporal path expressions. Finally, Section 5 concludes the paper.

2 Preliminaries: The Reference Temporal Object Model

In this section we introduce the temporal object model we refer to throughout the paper [9, 11].

We adopt the classical notion of *temporal granularity* [3]. Given a time domain $(\mathbb{N},\leq)$[2] and an index set $\mathcal{IS}$, a granularity G is a mapping from $\mathcal{IS}$ to $2^{\mathbb{N}}$ such that *(i)* if $i < j$ and $G(i)$ and $G(j)$ are non-empty, then each element of $G(i)$ is less than all elements of $G(j)$ and *(ii)* if

[1] An exception is represented by a previous work of ours [10] dealing with expressions involving data with multiple granularities.

[2] $\mathbb{N}$ is the set of natural numbers and represents the set of *time instants*, and $\leq$ is the order on $\mathbb{N}$.

$i < k < j$ and $G(i)$ and $G(j)$ are non-empty, then $G(k)$ is non-empty. Intuitively, a granularity defines a countable set of granules, each *granule* $G(i)$ is identified by an integer. The set of granularities is denoted by $\mathcal{G}$.

The usual collections $days$, $months$, and $weeks$ are granularities. For each non-empty granule, we use a "textual representation", termed as *label*. For example, $days$ are in the form $mm/dd/yyyy$. By contrast, when we refer to a generic granularity G and an index i, l_i^G denotes the label corresponding to the ith granule of G. When we refer to usual labels, such as the ones for $days$, i_l denotes the index corresponding to the granule denoted by label l.

A granularity G is said to be *finer than* a granularity H [3], denoted $G \preceq H$, if for each index i, there exists an index j such that $G(i) \subseteq H(j)$.[3] For example, $days \preceq months$. The finer than relationship will be used to evaluate path expressions involving several granularities.

We extend now the notion of *temporal interval* [8] and *temporal element* [6] to a multi-granularity model.

Definition 1 (*Temporal Interval*). Let $G \in \mathcal{G}$ be a granularity and $i, j \in \mathcal{IS}$ be two indexes such that $i \leq j$. Then $[i,j]^G = \{G(k) \mid i \leq k \leq j, k \in \mathcal{IS}\}$ is called *temporal interval*, with respect to granularity G. $\square$

Definition 2 (*Temporal Element*). Let $G \in \mathcal{G}$ be a granularity, then every subset of the set of granules associated to G, $\{G(i) \mid i \in \mathcal{IS}\}$, is called temporal element with respect to granularity G. $\square$

For instance, $\{months(i) \mid i \in \mathcal{IS}$ and $i_{01/1999} \leq i \leq i_{12/1999}\}$ is a temporal element representing the set of the months of year 1999. Every temporal element can be represented as a finite union of intervals. That is, let $[i_1, j_1]^G, \ldots, [i_n, j_n]^G$ be temporal intervals. Then, $[i_1, j_1]^G \cup \ldots \cup [i_n, j_n]^G$ represents the temporal element which includes all the granules included in each interval.[4] For instance, $[i_{01/1999}, i_{07/1999}]^{months} \cup [i_{10/1999}, i_{12/1999}]^{months}$ represents a temporal element.

In what follows temporal elements will be frequently denoted through the symbol Υ^G where G is the granularity of the temporal element. The set of all temporal elements with respect to a granularity G is denoted by $\mathcal{TELEM}_G$; in addition, $\mathcal{TELEM} = \bigcup_{G \in \mathcal{G}} \mathcal{TELEM}_G$. In [6], where temporal elements are subsets of the time domain, it is proved that the set of all temporal elements is closed under union, intersection, difference, and complementation and thus forms a boolean algebra. Such a result similarly applies to $\mathcal{TELEM}_G$. However, we often need to apply these operations to temporal elements at different granularities. In this case the sets are first *converted* to a common granularity: the greatest lower bound (glb) of their granularities with respect to the finer than relationship.[5] Thus, we now formalize the notion of *conversion* of a temporal interval and a temporal element from a granularity to another.

Definition 3 (*Conversion of a Temporal Interval*). Let $H \in \mathcal{G}$ be a granularity and $i, j \in \mathcal{IS}$ be two indexes such that $i \leq j$. $G([i,j]^H)$ denotes the conversion of the temporal interval $[i,j]^H$ to granularity G. $G([i,j]^H) = [h,k]^G$ such that $\bigcup_{i \leq p \leq j} H(p) \supseteq \bigcup_{h \leq p \leq k} G(p)$ and $\nexists [h',k']^G$ such that $\bigcup_{i \leq p \leq j} H(p) \supseteq \bigcup_{h' \leq p \leq k'} G(p) \supseteq \bigcup_{h \leq p \leq k} G(p)$. $\square$

If $G = H$, obviously $G([i,j]^G) = [i,j]^G$. The conversion can be applied if either $G \prec H$ (*downward conversion*) or $H \prec G$ (*upward conversion*). For instance, $days([i_{1995}, i_{2000}]^{years}) = [i_{01/01/1995}, i_{31/12/2000}]^{days}$ is a downward conversion, whereas $years([i_{01/01/1995}, i_{31/12/2000}]^{days}) = [i_{1995}, i_{2000}]^{years}$ is an upward conversion. The conversion of a temporal element Υ^H is computed by repeatedly applying the conversion to each temporal interval composing the temporal element.

In what follows, given a set of granularities $\{G_1, \ldots, G_n\}$, $glb(G_1, \ldots, G_n)$ denotes the granularity which is the greatest lower bound of $G_1, \ldots, G_n$ in $\mathcal{G}$ with respect to the finer than relationship. Similarly, $lub(G_1, \ldots, G_n)$ denotes the lowest upper bound.

We introduce now some notations. Given an interval $[i,j]^G$, $min([i,j]^G)$ denotes the lower bound i, whereas $max([i,j]^G)$ denotes the upper bound j. In addition, we define a projection operation $\Pi(\Upsilon^G, n)$, that takes as input a temporal element Υ^G and a natural number n. Π orders the elements in Υ^G in increasing order, with respect to their upper bound, and returns the n-th interval in the ordering. If $\mid \Upsilon^G \mid \leq n$,[6] $\Pi(\Upsilon^G, n)$ is undefined.

We can now introduce the notion of temporal types related to different granularities. We refer the interested reader to [2, 9, 11] for a detailed description of the reference model. In our model object types can be defined through classes. We consider a classical notion of class [4] where, in order to store temporal information, the type of a property can be a temporal one. Figure 1 presents an example of a temporal object database schema involving multiple granularities.

We assume that a set $\mathcal{T}_R$ of types is given. Such set includes class and literal types.[7] For each type $\tau \in \mathcal{T}_R$ and granularity $G \in \mathcal{G}$, a corresponding temporal type, $temporal_G(\tau)$, is defined.[8] For instance, $temporal_{months}(\texttt{Person})$ is an example of a temporal

[3] The symbol "$\prec$" denotes the anti-reflexive finer than relationship.

[4] A granule itself is obviously a temporal element.

[5] Note that two granularities in $\mathcal{G}$ are not guaranteed to have a glb in $\mathcal{G}$. Here and in the remainder of this paper the approach is that two granularities can be "used together" only if they admit a glb in $\mathcal{G}$.

[6] Given a generic set S, $\mid S \mid$ denotes the cardinality of S.

[7] Example of types belonging to $\mathcal{T}_R$ are `short`, `Person`, and so on.

[8] Note that temporal types cannot be nested.

```
class Course {...;
            attribute temporal_years(short) room;
            relationship temporal_years(Professor) T_prof
                        inverse Professor::teaches;
            relationship temporal_semesters(Researcher) T_assist
                        inverse Researcher::assists;
            relationship temporal_years(set<Student>) P_stud
                        inverse Student::attends;}

class I_Course extends Course {ref relationship temporal_years(Researcher) T_assist
                                            inverse Researcher::assists;
                               ref relationship temporal_days(set<Student>) P_stud
                                            inverse Student::attends count_students;
                               attribute temporal_weeks(string) lab;}

class E_Course extends Course {ref relationship temporal_months(Researcher) T_assist
                                            inverse Researcher::assists main;
                               attribute temporal_months(string) lab;}

class T_Course extends Course {ref attribute temporal_months(short) room all;}

class Person {...};

class UnivEmployee extends Person {attribute short emp#;
                                   attribute temporal_months(short) salary;
                                   attribute temporal_years(short) room;};

class Student extends Person {...;
                              relationship temporal_years(Professor) supervisor
                                          inverse Professor::supervises;}

class Professor extends UnivEmployee {attribute temporal_years(string) business_hours;
                                      relationship temporal_years(Course) teaches
                                           inverse Course::T_prof;
                                      relationship temporal_years(set<Student>) supervises
                                           inverse Student::supervisor;}

class Researcher extends UnivEmployee {ref attribute temporal_months(short) room main;
                                       attribute temporal_weeks(string) supervised_lab;
                                       relationship temporal_months(set<Course>) assists
                                            inverse Course::T_assist;}
```

Figure 1. Example of database schema

type provided that $\texttt{Person} \in \mathcal{T}_R$ and $months \in \mathcal{G}$. The set of types provided by our model, which includes temporal, literal, and object types, is denoted as $\mathcal{T}$.

Given a non-temporal type τ and a time instant t, $[\![\tau]\!]_t$ denotes the extent of type τ at time t. If τ is a literal type $[\![\tau]\!]_t$ simply denotes the set of values of that type, whereas if τ is a class $[\![\tau]\!]_t$ returns the set of objects belonging to type τ at time t. Indeed, no literal value can be explicitly created or deleted, whereas objects belonging to classes are dynamically created and deleted, thus the extent of a class depends on time.

The set of values of a non-temporal type with respect to a time instant is generalized to a granularity G as follows. $[\![\tau]\!]_i^G$ denotes the extension of type τ with respect to the ith granule of G, that is, $[\![\tau]\!]_i^G = \bigcap_{t \in G(i)} [\![\tau]\!]_t$.[9] The idea behind this is that if an object o belonging to class c exists only during a portion of a granule, it does not belong to the extent of c related to such a granule. The set of legal values of a temporal type $temporal_G(\tau)$ is defined as follows:

$$[\![temporal_G(\tau)]\!] = \{f \mid f : \mathcal{IS} \to \bigcup_{i \in \mathcal{IS}} [\![\tau]\!]_i^G \text{ is a partial func. s.t. } \forall i \in \mathcal{IS} \text{ if } f(i) \neq \perp \text{ then } f(i) \in [\![\tau]\!]_i^G\}.$$

Example 1 Let `Course` be a class such that: $D = \bigcup_{i \in \mathcal{IS}} [\![\texttt{Course}]\!]_i^{years} = \{\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3, \ldots, \mathrm{d}_n\}$, then examples of functions, denoted as set of pairs, in $[\![temporal_{years}(\texttt{Course})]\!]$ are $v_1 = \{\langle i_{1992}, \mathrm{d}_1\rangle, \langle i_{1993}, \mathrm{d}_4\rangle\}$; $v_2 = \{\langle i_{1992}, \mathrm{d}_1\rangle, \langle i_{1993}, \mathrm{d}_1\rangle, \langle i_{1994}, \mathrm{d}_1\rangle\}$. $\diamond$

When the function representing a temporal value is constant for a set of contiguous granules, that

[9] We frequently use $[\![\tau]\!]_{l_i^G}$ to denote the legal values of a type with respect to a granule.

is, an interval, we denote temporal values as set of pairs $\langle temporal\ interval, value\rangle$. For instance, $\{\langle[i_{1992}, i_{1994}]^{years}, \mathtt{d}_1\rangle\}$ is a compact notation for value v_2 of Example 1.

We denote with $\mathcal{V}$ the set of legal values for types in $\mathcal{T}$. Let v be a value of type $temporal_G(\tau)$, we denote with $v(i)$ the value of v in ith granule of G. We assume that, for each granularity H such that $H \preceq G$, and for each $i, j \in \mathcal{IS}$ such that $H(j) \subseteq G(i)$, the value of v in granule j of H is the one in the ith granule of G. This assumption is known in the temporal reasoning community [14] as *downward hereditary* property. We simply denote such value as $v^H(j)$. For instance, let $v_1 \in [\![temporal_{years}(\mathtt{Department})]\!]$ be the temporal value presented in Example 1, then $v_1^{months}(i_{01/1992}) = \mathtt{d}_1$.

One can argue that downward hereditary property is not always appropriate. We believe that this property is realistic in most cases. In fact, storing the value of an attribute with respect to a granularity is somehow deciding the temporal precision associated with the information. If the attribute granularity is $months$, then for each granularity finer than $months$, such as $days$, this information is imprecise. However, since a value has been associated with each month, this information is as close as possible to the value of each day.

3 Temporal Expressions

In this section we first briefly discuss how temporal values related to different temporal granularities can be combined, then we present the syntax and semantics of temporal expressions. Temporal expressions are the mean by which the set of granules, that is, the temporal element, with respect to which a query is evaluated, are implicitly specified.

Combining temporal values expressed with respect to different granularities raises several interesting issues [10]. The intuitive meaning of an operator *op* applied to two temporal values, v_1 and v_2, is the "point to point", that is, "granule to granule" in our context, evaluation of the operation denoted by *op*. For each comparison operator *op* we introduce a *temporal* variation (op_T), whose intuitive meaning is to answer the following question: "when the relationship denoted by *op* holds?" These operators are used in *temporal expressions*. In case the two values are expressed with respect to different granularities two cases can be devised. If one granularity is finer than the other, the "coarser" value is converted to the finer granularity. If the previous condition is not verified, but the two granularities are in some way comparable, that is, a granularity K finer than both of them exists, the two values are converted to K. If none of the previous conditions is verified an undefined value is returned, that corresponds to an error detection. For lack of space we give only some examples of expressions involving temporal values, we refer the interested reader to [9, 10] for further details.

Example 2 Let v_1= $\{\langle i_{01/1999}, 100\rangle, \langle i_{02/1999}, 150\rangle, \langle i_{03/1999}, 300\rangle\}$ and v_2= $\{\langle i_{01/1999}, 400\rangle, \langle i_{03/1999}, 50\rangle\}$ be two values of type $temporal_{months}(\mathtt{short})$. Then: $v_1 + v_2 = \{\langle i_{01/1999}, 500\rangle, \langle i_{03/1999}, 350\rangle\}$, $v_1 > v_2 = \{\langle i_{01/1999}, false\rangle, \langle i_{03/1999}, true\rangle\}$. In addition, $v_1 >_T v_2 = [i_{03/1999}, i_{03/1999}]^{months}$. Consider now $v_1 = \{\langle i_{12/1999}, 5\rangle, \langle i_{01/2000}, 7\rangle\} \in [\![temporal_{months}(\mathtt{short})]\!]$ and $v_2 = \{\langle i_{01/12/1999}, 5\rangle, \langle i_{02/12/1999}, 10\rangle, \langle i_{03/12/1999}, 8\rangle, \langle i_{04/12/1999}, 1\rangle, ...\} \in [\![temporal_{days}(\mathtt{short})]\!]$. Then, $v_1 <_T v_2 = v_1^{days} <_T v_2 = \{days(i) \mid v_1(i) < v_2(i) = true\} = \{days(i_{02/12/1999}), days(i_{03/12/1999}), ...\}$. $\diamond$

3.1 Syntax

Temporal expressions evaluated on an object return the temporal element in which a boolean condition is satisfied on the object. They are somehow the temporal counterpart of boolean expressions. Indeed, they answers the query: "*when*" is a certain condition verified? Temporal expressions are built by combining simple expressions which denote values in $\mathcal{V}$. Their syntax in BNF form is reported in Figure 2. Terminal symbol `nat` represents a natural number, `granularity` represents a granularity in $\mathcal{G}$, `value` represents an element of $\mathcal{V}$, `count` denotes the usual aggregate function, `att_name`, `rel_name` represent an attribute, relationship name, respectively.

Example 3 The following are examples of temporal expressions which can be evaluated with respect to an object of class `Course` of the database schema of Figure 1:

`room=`$_T$`107 and T_assist.room=`$_T$` 207,`

`first(count(P_stud)>`$_T$`25 or T_prof.salary>`$_T$`70),`

$years$`(T_assist.salary>`$_T$`55k).` $\diamond$

3.2 Semantics

The semantics of temporal expressions is built on top of the semantics of simple expressions (cf. Figure 2). A simple expression, evaluated on an object, denotes a value. It can be a value itself, the navigation through a path, the count operator applied to an expression, and an operation applied to two expressions. For instance, `T_prof.salary`, is a simple expression which evaluated on an object of class `Course` returns a value storing the history of the salaries of the professors who have ever taught that course.

In case of paths, which represent the navigation through objects, the evaluation of a path expression $p_1.p_2.\cdots.p_n$ is the value of property p_n starting to navigate from the given object through properties $p_1, \ldots, p_{n-1}$. Each property p_i,

$\langle$temp_expr$\rangle$ ::= $\langle$simple_expr$\rangle$ $\langle$temp_comp_op$\rangle$ $\langle$simple_expr$\rangle$ | $\langle$temp_expr$\rangle$ $\langle$bool_op$\rangle$ $\langle$temp_expr$\rangle$ |
$\langle$single_op$\rangle$ ($\langle$temp_expr$\rangle$) | $\langle$slice_op$\rangle$ ($\langle$temp_expr$\rangle$, nat) | granularity($\langle$temp_expr$\rangle$)

$\langle$simple_expr$\rangle$ ::= value | $\langle$path$\rangle$ | count($\langle$simpl_expr$\rangle$) | $\langle$simpl_expr$\rangle$ $\langle$op$\rangle$ $\langle$simpl_expr$\rangle$

$\langle$path$\rangle$ ::= att_name | rel_name | $\langle$path$\rangle$.att_name | $\langle$path$\rangle$.rel_name

$\langle$temp_comp_op$\rangle$ ::= $>_T$ | $<_T$ | $>=_T$ | $<=_T$ | $=_T$ | $!=_T$ | $\in_T$ | $\notin_T$ $\langle$op$\rangle$ ::= + | - | * | / | $\cup$ | $\cap$ | \ $\langle$bool_op$\rangle$ ::= and | or

$\langle$single_op$\rangle$ ::= not|first|last|first_instant|last_instant $\langle$slice_op$\rangle$ ::= slice|inst_slice

Figure 2. Syntax of the language for expressing temporal expressions

$i \in [1, n]$ can be expressed with respect to a different granularity. Consider indeed a path $p_1.p_2.\cdots.p_n$ such that p_1 is a property of the class of the object on which the expression is evaluated whose domain is $temporal_{G_1}(\tau_1)$ and p_i, for $i \in [2, n]$, is a property of class τ_i whose domain is $temporal_{G_i}(\tau_i)$. The value denoted by this expression is a temporal value of granularity $glb(G_1, \ldots, G_n)$ whose value in each granule is obtained by evaluating the value of the path at that granule. If $glb(G_1, \ldots, G_n)$ in $\mathcal{G}$ does not exist, then the value of such expression is not defined. For instance, the path supervisor.teaches.P_stud, evaluated on an object of class Student of Figure 1 whose supervisor has ever taught an introductory course, denotes a temporal value of type $temporal_{days}(\texttt{set} < \texttt{Student} >)$.

In addition, in agreement with OQL, we impose the restriction that path navigation across multivalued properties is not allowed. In what follows, given an object o, $\nu(o)$ denotes the value of its properties. If p is a property of o, then $\nu(o).p$ denotes the value of p for o. The semantics of simple expressions is formally defined as follows.

Definition 4 (*Semantics of Simple Expressions*). Let $\mathcal{OI}$ be a set of objects, $\mathcal{V}$ be the set of values, and Exp_{simple} be the set of well-formed simple expressions (cf. Figure 2). Then the semantics of simple expressions is defined through function: $\mathcal{E}_{simple} : Exp_{simple} \rightarrow (\mathcal{OI} \rightarrow \mathcal{V})$, such that, given an expression $e \in Exp_{simple}$ and an object $o \in \mathcal{OI}$:

- if $e = v, v \in \mathcal{V}$, then $\mathcal{E}_{simple} [\![v]\!] o = v$;
- if $e = p_1.p_2.\cdots.p_n$, with $p_1, p_2, \ldots, p_n$ properties, $\mathcal{E}_{simple} [\![p_1.p_2.\cdots.p_n]\!] o = \{\langle i, (\mathcal{E}_{simple} [\![p_2.\cdots.p_n]\!] \nu(o).p_1^K(i))^K(i)\rangle \mid K = glb(G_1, \ldots, G_n), i \in \mathcal{IS}\}$ if the type of property p_1 for o is $temporal_{G_1}(c_1) \in \mathcal{TT}$ and the type of property p_j in c_{j-1} is $temporal_{G_j}(c_j) \in \mathcal{TT}, j \in [2, n]\}$;
- if $e = \texttt{count}(e')$, $e' \in Exp_{simple}$, then $\mathcal{E}_{simple} [\![\texttt{count}(e)]\!] o = \mid \mathcal{E}_{simple} [\![e]\!] o \mid$;
- if $e = e_1 \ op \ e_2$, then $\mathcal{E}_{simple} [\![e_1 \ op \ e_2]\!] o = (\mathcal{E}_{simple} [\![e_1]\!] o) \ op \ (\mathcal{E}_{simple} [\![e_2]\!] o)$. □

Intuitively, the resulting value of a temporal expression is the temporal element in which all the conditions expressed by each expression composing it are verified.

Example 4 The evaluation of the first temporal expression of Example 3, room $=_T$ 107 and T_assist.room $=_T$ 207, on an object of class I_Course, returns a set of temporal intervals of granularity $months$, obtained as the intersection of the intervals in which the two conditions are satisfied. ◊

The semantics of a temporal expression is formalized by means of function $\mathcal{E}_{temp} : Exp_{temp} \rightarrow (\mathcal{OI} \rightarrow \mathcal{TELEM})$ defined in Figure 3. In Figure 3 symbols e, e_1, and e_2 denote temporal expressions, o denotes an object, n denotes a natural number, and G denotes a granularity.

In case of temporal expressions involving the negation operator not, the semantics is defined using the set operation complement. The universe with respect to which the complement is computed is the set of granules in which expression e is defined on object o.

Note that the only meaningful use of the conversion operator G (cf. Definition 3) in temporal expressions is for upward conversion, that is, to convert to a coarser granularity, to obtain a granule-valued expression, that is, an expression whose evaluation returns a single granule.

According to the notion of conversion of temporal interval and temporal element we adopt, the semantics of upward conversion we consider, converts expressions that denote a superset of the set of instants associated to a set of granules of the coarser granularity. Actually, upward conversion could be extended to arbitrary intervals, with different alternative semantics. For lack of space we do not deal with alternative semantics, we refer the interested reader to [9] .

4 Temporal Path Expressions

Usually, internal nodes of a path expression must produce a single object to which subsequent accesses are applied. In a multi-granularity context, to ensure that property,

$$\mathcal{E}_{bool} [\![e_1 op_T e_2]\!] o = (\mathcal{E}_{simple} [\![e_1]\!] o) \ op_T \ (\mathcal{E}_{simple} [\![e_2]\!] o)$$

$$\mathcal{E}_{temp} [\![e_1 op e_2]\!] o = \begin{cases} \mathcal{E}_{temp} [\![e_1]\!] o \cup \mathcal{E}_{temp} [\![e_2]\!] o & \text{if } op = \texttt{or} \\ \mathcal{E}_{temp} [\![e_1]\!] o \cap \mathcal{E}_{temp} [\![e_2]\!] o & \text{if } op = \texttt{and} \end{cases}$$

$$\mathcal{E}_{temp} [\![op(e)]\!] o = \begin{cases} (\mathcal{E}_{temp} [\![e]\!] o)^c & \text{if } op = \texttt{not} \\ \Pi(\mathcal{E}_{temp} [\![e]\!] o, 1) & \text{if } op = \texttt{first} \\ \Pi(\mathcal{E}_{temp} [\![e]\!] o, m) & \text{if } op = \texttt{last} \text{ and } m = \mid \mathcal{E}_{temp} [\![e]\!] o \mid; \\ [i,i]^G & \text{if } op = \texttt{first\_instant} \text{ and } i = min(\Pi(\mathcal{E}_{temp} [\![e]\!] o, 1)) \\ [i,i]^G & \text{if } op = \texttt{last\_instant}, i = max(\Pi(\mathcal{E}_{temp} [\![e]\!] o, m)), \text{ and} \\ & \text{if } m = \mid \mathcal{E}_{temp} [\![e]\!] \mid \end{cases}$$

$$\mathcal{E}_{temp} [\![op(e,n)]\!] o = \begin{cases} \Pi(\mathcal{E}_{temp} [\![e]\!] o, n) & \text{if } op = \texttt{slice} \\ [i,i]^G & \text{if } op = \texttt{instant\_slice} \\ & \text{and } i \text{ is the } i\text{th granule in } \mathcal{E}_{temp} [\![e]\!] o \end{cases}$$

$$\mathcal{E}_{temp} [\![G(e)]\!] o = G(\mathcal{E}_{temp} [\![e]\!] o)$$

Figure 3. Semantics of temporal expressions

whichever the granularities of the accessed property and of the granule are, we rely on the use of *coercion functions*. Coercion functions allow one to convert values from a given granularity into values of a coarser granularity in a meaningful way. Thus, a coercion function C is a partial function such that $C : [\![temporal_H(\tau)]\!] \rightarrow [\![temporal_G(\tau)]\!]$, where $H \preceq G$. In [11] coercion functions were associated with property definitions allowing one to specialize property domains in a type with a granularity finer than the one in the property domain to be redefined.

Example 5 In Figure 1, the type $temporal_{years}(\texttt{short})$ of attribute `room` of class `Course` is specialized in class `T_Course` sub_class of `Course` in $temporal_{months}(\texttt{short})$. Since $months \preceq years$, coercion function `all` is associated with it in order to correctly perform object accesses. Coercion function `all`, for each granule in the coarser granularity, returns the value which always appears in the included granules of the finer one if this value exists, the null value otherwise. ◇

Coercion functions have been inspired by *semantic assumptions* [3], a way of deriving implicit information from explicitly stored (relational) data. In our approach we adopt somehow the use of semantic assumption to temporal object-oriented databases.

In what follows, we first discuss why we need coercion functions and then how they are used. Consider an access to a property p of an object o at a granule l_i^G. Let the property domain be $temporal_H(\tau)$. If $G \preceq H$, we can access the value of property p at granule l_i^G, and this access uniquely denotes a value. We denote such access as $\nu(o).p^G(i)$. If, by contrast, $H \prec G$, property p may assume different values in the granule identified by l_i^G. For instance, consider the class `E_Course` of Figure 1. If we access property `lab` of an `E_Course` object specifying a year, since that property may vary every month, the property may take different values over the year, corresponding to the fact that the laboratory has been changed during that year. In this case we make use of coercion functions to obtain a single value starting from those taken by property p in the granule l_i^G. Thus, here we allow one to attach such functions to property accesses, and not only to property definitions. If a coercion function is specified for a property access this function is used to evaluate the expression, even if a different coercion function has been specified in the schema for the accessed property. If, by contrast, no coercion function is specified for a property access, the one associated with the property is employed.

Wherever they are specified, coercion functions are employed to convert a temporal value to a coarser granularity. In [11] we have devised different kinds of coercion functions. Here, we only refer to selective and user-defined coercion functions. Let $\{i_1, \ldots, i_k\}$ be the set of indexes such that $H(i_p) \subseteq G(j)$ and let $v \in [\![temporal_H(\tau)]\!]$ such that $v(i_p) = v_p$, $p \in [1,k]$. Then, intuitively, in case of selective coercion functions, one of the possible values among $\{v_1, \ldots, v_k\}$ is chosen for a generic granule j. Function `main` of Example 5 is a selective coercion function. In case of user-defined coercion functions, the method to convert from one granularity to the other is completely specified by the user.

4.1 Syntax

The syntax of temporal path expressions is presented in Figure 4. Terminal symbol `var` denotes a variable, `granule` denotes a granule label, and `temp_element` denotes a temporal element. In addition, the rules of the non-terminal symbols ⟨`temp_expr`⟩ and ⟨`path`⟩ can be found in Figure 4.

Example 6 Consider the schema of Figure 1 and let `X` be a variable of type `Course`. `X.T_assist` ↓ `1st_sem_1999` denotes the assistant of the course during the first semester of 1999; `X.T_prof`↓`first_instant(count(P_stud)` $>_T$`25)`.

```
⟨path_expr⟩ ::= ⟨simple_path_expr⟩ | ⟨term_path_expr⟩

⟨simple_path_expr⟩ ::= var | var.⟨path⟩ | ⟨simple_path_expr⟩ ↓ granule | ⟨simple_path_expr⟩ ↓^⟨coerc_func⟩ granule |
                       ⟨simple_path_expr⟩ ↓ ⟨temp_expr⟩ | ⟨simple_path_expr⟩ ↓^⟨coerc_func⟩ ⟨temp_expr⟩

⟨coerc_func⟩ ::= ⟨selective_coerc_func⟩ | ⟨user-def_coerc_func⟩
⟨selective_coerc_func⟩ ::= first | last | Proj(index) | main | all
⟨user-def_coerc_func⟩ ::= meth_inv

⟨term_path_expr⟩ ::= ⟨simple_path_expr⟩ ↓ temp_element | ⟨simple_path_expr⟩ ↓^⟨coerc_func⟩ temp_element
```

Figure 4. Syntax of the language for expressing temporal path expressions

`business_hours` $\downarrow$ `(count(supervises)`$>_T$`5)` denotes the business hours, in time intervals in which he/she supervised more than five students, of the professor who was teaching the course in the first instant in which it was attended by more than 25 students. $\diamond$

4.2 Semantics

The semantics of a path expression can only be specified starting from an object-assignment and depends on the temporal specifications it contains.

Consider first simple path expressions, for which a granule l_i^G is specified (either implicitly or explicitly), that is, path expressions of the form $e.p \downarrow l_i^G$ with a coercion function optionally specified ($e.p \downarrow^C l_i^G$). Let o be an object to which e evaluates, and let the granularity of property p for object o be H. Moreover, given a temporal property p and a granule identified by l_i^G, $p(i)$ denotes the value of p in the granule l_i^G. Let $K \preceq G$, $p^K(i)$ denotes the value of property p with respect to granularity K, such value is equal to $p(j)$ where $K(i) \subseteq G(j)$. The value denoted by the path expression with respect to an object o is:

- $\nu(o).p(i)$, if $G = H$ (the eventually specified coercion function is irrelevant);
- $\nu(o).p^G(i)$, if $G \prec H$ (the eventually specified coercion function is irrelevant);
- $C(\nu(o).p)(i)$ if $H \prec G$, where C is the coercion function specified for the access, if any, and the one associated with property p in the class to which o belongs otherwise;[10]
- $C(\nu(o).p)^K(j)$ if H and G are not comparable under $\preceq$ but they have a least upper bound K, $G(i) \subseteq K(j)$, and C is the coercion function determined as in the case above. In this case, C is used to coerce values of property p to granularity K and the value of the property in the K-granule containing $G(i)$ is accessed.

[10] If $H \prec G$ and no coercion function is specified for the access, nor attached to property p in the class to which o belongs, the value of the expression is undefined.

Example 7 Let $\mathtt{o_i}, \mathtt{o_e}, \mathtt{o_t}$ be objects of classes `I_Course`, `E_Course`, `T_Course` of Figure 1, respectively. $\mathtt{o_t}$`.T_assist` $\downarrow$ `1st_sem_1999` denotes the value stored in that granule; $\mathtt{o_i}$`.T_assist` $\downarrow$ `1st_sem_1999` denotes the value stored in the $year$-granule 1999; $\mathtt{o_e}$`.T_assist` $\downarrow$ `1st_sem_1999` denotes the value obtained by applying coercion function `main` to the values stored in the $months$-granules of the first semester of 1999; $\mathtt{o_e}$`.T_assist` $\downarrow^{\mathtt{first}}$ `1st_sem_1999` denotes the value obtained by applying coercion function `first` to the values stored for the relationship in the $months$-granules corresponding to the first semester of 1999. $\diamond$

Consider now the terminal path expression $e.p \downarrow \Upsilon^G$, where Υ^G is a temporal element (either explicitly or implicitly denoted). The value associated with this expression is the restriction of the temporal value $\nu(o).p$ to the time instants in Υ^G. Different interpretations of this path expression are however possible if the value and the temporal element are expressed at different granularities. Let o be the object to which e evaluates and the granularity of property p for o be H. Different cases can be distinguished:

- $G = H$: the expression denotes a temporal value of granularity G which is the restriction of $\nu(o).p$ to Υ^G (cf. Definition 5);
- $G \prec H$: the expression denotes a temporal value of granularity G which is $\nu(o).p$ seen as a value of granularity G and restricted to Υ^G;
- $H \prec G$: two alternative interpretations are possible:
  - no coercion function is specified for the access: the expression denotes a temporal value of granularity H obtained by restricting $\nu(o).p$ to the H-granules in Υ^G;

- a coercion function C is specified for the access: the expression denotes a temporal value of granularity G obtaining by restricting $C(\nu(o).p)$ to Υ^G;

- G and H are not comparable: two alternative interpretations are possible:
  - no coercion function is specified for the access: the expression denotes a temporal value of granularity $J = glb(G, H)$ which is the restriction of $\nu(o).p$ (seen as a value of granularity J) to the J-granules in Υ^G;
  - a coercion function C is specified for the access: the expression denotes a temporal value of granularity G obtaining by restricting the coercion $C(\nu(o).p)$ of $\nu(o).p$ at granularity $K = lub(G, H)$, to Υ^G.

Example 8 Consider again Figure 1 and let $\mathtt{o_a}$ be the identifier of a `Researcher` object for which $\mathtt{count(assists)} >_{\mathtt{T}} 2$ denotes a set of intervals at granularity $months$.

- $\mathtt{o_a.room} \downarrow (\mathtt{count(assists)} >_{\mathtt{T}} 2)$ denotes a temporal value of type $temporal_{months}(\mathtt{short})$ obtained by restricting the value of the `room` attribute to those months in which the researcher was the assistant of more than two courses;

- $\mathtt{o_a.room} \downarrow [\mathtt{01/01/99}, \mathtt{18/01/99}]$ denotes a temporal value of type $temporal_{days}(\mathtt{short})$ obtained by restricting the value of the `room` attribute to those days belonging to specified temporal interval;

- $\mathtt{o_a.room} \downarrow^{\mathtt{first}} [\mathtt{1999}, \mathtt{2000}]$ denotes a temporal value of type $temporal_{years}(\mathtt{string})$ obtained by restricting the value of the `room` attribute corced to the $years$ granularity by means of the `first` function, to years 1999 and 2000;

- $\mathtt{o_a.supervised\_lab} \downarrow (\mathtt{count(assists)} >_{\mathtt{T}} 2)$ denotes a temporal value of type $temporal_{days}(\mathtt{string})$ obtained by restricting the value of the `supervised_lab` attribute to those days belonging to months in which the researcher was the assistant of more than two courses. $\diamond$

The restriction of a temporal value to a temporal element is formalized by the following definition.

Definition 5 *(Temporal Value Restriction).* Let $v \in [\![temporal_H(\tau)]\!]$ be a temporal value and let $\Upsilon^G \in \mathcal{TELEM}_G$ be a temporal element. The restriction of v to Υ^G, denoted as $v_{|\Upsilon^G}$, is defined as follows:

- if $G \preceq H$, $v_{|\Upsilon^G} \in [\![temporal_G(\tau)]\!]$ such that $v_{|\Upsilon^G}(i) = v(j)$ if $G(i) \in \Upsilon^G$ and $G(i) \subseteq H(j)$, undefined otherwise;

- if $H \prec G$, $v_{|\Upsilon^G} \in [\![temporal_H(\tau)]\!]$ such that $v_{|\Upsilon^G}(i) = v(i)$ if $\exists G(j) \in \Upsilon^G$ s.t. $H(i) \subseteq G(j)$, undefined otherwise. $\square$

We are now ready to define the semantics of a path expression. Let Exp_{path} be the set of well-formed path expressions (cf. Figure 4), $\vartheta : Var \rightarrow \mathcal{OI}$[11] be an object-assignment, and Θ be the set of all object-assignments. The semantics of a path expression e under object-assignment ϑ is defined through the following semantic function $\mathcal{E}_{path} : Exp_{path} \rightarrow (\Theta \rightarrow \mathcal{V})$ which is formally specified in Figure 5. In Figure 5 we use function γ to denote the granularity of a temporal value, that is, if $v \in [\![temporal_G(\tau)]\!]$, $\gamma(v) = G$. In addition, X denotes a variable, p denotes an object property, l_i^G denotes a granule label, te denotes a temporal expression, C denotes a coercion function, and Υ^G denotes a temporal element.

5 Concluding Remarks

In this paper we have investigated navigation through objects whose property values are expressed with respect to several granularities. This can be considered as the *core* of a language which extends OQL path expressions [4] to query data expressed with respect to different granularities. The proposed navigation has been implemented on top of ObjectStore Pse as part of a prototype we have developed. Such prototype implements T_ODMG, the temporal extension to ODMG supporting multiple temporal granularity data we have proposed [9]. Several different semantics could be devised in evaluating the presented expressions. We leave for future work the introduction in the language of syntactic constructs according to which different semantics could be supported. Finally, our future works include the definition of a full temporal query language and optimization.

References

[1] E. Bertino, E. Ferrari, and G. Guerrini. Navigational Access in a Temporal Object Model. *IEEE TKDE*, 10(4), 1998.

[2] E. Bertino, E. Ferrari, G. Guerrini, and I. Merlo. An ODMG Compliant Temporal Object Model Supporting Multiple Granularity Management. Technical Report DISI-TR-00-08, DISI, Università di Genova, 2000.

[3] C. Bettini, S. Jajodia, and X.S. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer-Verlag, 2000.

[11] Var is a set of object denoting variables.

$$\mathcal{E}_{path}\,[\![\, X \,]\!]\,\vartheta = \vartheta(X) \qquad\qquad \mathcal{E}_{path}\,[\![\, X.p \,]\!]\,\vartheta = \nu(\mathcal{E}_{path}\,[\![\, X \,]\!]\,\vartheta).p =$$

$$\mathcal{E}_{path}\,[\![\, X.path.p \,]\!]\,\vartheta = \nu(\mathcal{E}_{path}\,[\![\, X.path \,]\!]\,\vartheta).p$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow l_i^G \,]\!]\,\vartheta = \begin{cases} \nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p(i) & \text{if } G = \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \text{ or} \\ & (G \prec \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \\ \mathcal{E}_{path}\,[\![\, e.p \downarrow^C l_i^G \,]\!]\,\vartheta & \text{if } \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \prec G \\ & \text{and } C \text{ is the coerc. func. associated with } p \\ undefined & \text{otherwise} \end{cases}$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow^C l_i^G \,]\!]\,\vartheta = \begin{cases} \mathcal{E}_{path}\,[\![\, e.p \downarrow l_i^G \,]\!]\,\vartheta & \text{if } G \preceq \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \\ C(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p)^K(j) & \text{otherwise,} \\ & K = lub(G, \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p)), \text{ and} \\ & G(i) \subseteq K(j) \end{cases}$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow te \,]\!]\,\vartheta = \mathcal{E}_{path}\,[\![\, e.p \downarrow (\mathcal{E}_{temp}\,[\![\, te \,]\!]\,\vartheta(X)) \,]\!]\,\vartheta$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow^C te \,]\!]\,\vartheta = \mathcal{E}_{path}\,[\![\, e.p \downarrow^C (\mathcal{E}_{temp}\,[\![\, te \,]\!]\,\vartheta(X)) \,]\!]\,\vartheta$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow \Upsilon^G \,]\!]\,\vartheta = \begin{cases} \nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p_{|\Upsilon^G} & \text{if } \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \preceq G \\ \nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p_{|\Upsilon'^G} & \text{if } G \preceq \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \\ & \Upsilon'^G = \{G(i) \mid G(i) \in \Upsilon^G \\ \nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p_{|J(\Upsilon^G)} & \text{otherwise (i.e., } \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \text{ and } G \\ & \text{are not comparable under } \preceq), \\ & J = glb(\gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p)), G) \end{cases}$$

$$\mathcal{E}_{path}\,[\![\, e.p \downarrow^C \Upsilon^G \,]\!]\,\vartheta = \begin{cases} \mathcal{E}_{path}\,[\![\, e.p \downarrow^C \Upsilon^G \,]\!]\,\vartheta & \text{if } G \preceq \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p) \\ C(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p)_{|K(\Upsilon^G)} & \text{otherwise,} \\ & K = lub(G, \gamma(\nu(\mathcal{E}_{path}\,[\![\, e \,]\!]\,\vartheta).p)) \end{cases}$$

Figure 5. Semantics of temporal path expressions

[4] R. Cattel, D. Barry, M. Berler, J. Eastman, D. Jordan, C. Russel, O. Schadow, T. Stanienda, and F. Velez. *The Object Database Standard: ODMG 3.0*. Morgan-Kaufmann, 1999.

[5] C. Combi, G. Cucchi, and F. Pinciroli. Applying Object-Oriented Technologies in Modeling and Querying Temporally Oriented Clinical Databases Dealing with Temporal Granularity and Indeterminacy. *IEEE Transactions on Information Technology in Biomedicine*, 1(2), 1997.

[6] S.K. Gadia. A homogeneous relational model and query languages for temporal databases. *ACM TODS*, 13(4), 1988.

[7] S.K. Gadia and S.S. Nair. Temporal Databases: A Prelude to Parametric Data. In A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

[8] C.S. Jensen and C.E. Dyreson. The Consensus Glossary of Temporal Database Concepts. In *Temporal Databases: Research and Practice*, number 1399 in LNCS, 1998.

[9] I. Merlo. *Extending the ODMG Object Model with Temporal and Active Capabilities*. PhD thesis, Università di Genova, February 2001.

[10] I. Merlo, E. Bertino, E. Ferrari, S. Gadia, and G. Guerrini. Querying Multiple Temporal Granularity Data. In S. Goodwin and A. Trudel, editors, *IEEE Proc. TIME 2000*.

[11] I. Merlo, E. Bertino, E. Ferrari, and G. Guerrini. A Temporal Object-Oriented Data Model with Multiple Granularities. In C. Dixon and M. Fischer, editors, *IEEE Proc. TIME 1999*.

[12] M. T. Ozsu, R. Peters, D. Szafron, B. Irani, A. Lipka, and A. Munoz. TIGUKAT: A Uniform Behavioral Objectbase Management System. *VLDB Journal*, 4(3), 1995.

[13] E. Rose and A. Segev. TOODM - A Temporal Object-Oriented Data Model with Temporal Constraints. In *Proc. Tenth Int'l Conf. on the Entity-Relationship Approach*, 1991.

[14] Y. Shoham. Temporal Logics in AI: Semantical and Ontological Considerations. *Artificial Intelligence*, 33(1), 1987.

[15] G. Wuu and U. Dayal. A Uniform Model for Temporal and Versioned Object-Oriented Databases. In A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.