International Journal of Cooperative Information Systems,

Context-Based Naming in Information Bases

Manos Theodorakis
Department of Computer Science, University of Crete, and
Institute of Computer Science, Foundation for Research and Technology - Hellas
P.O.Box 1385, GR-71110, Heraklion, Greece
etheodor@ics.forth.gr

and

Panos Constantopoulos
Department of Computer Science, University of Crete, and
Institute of Computer Science, Foundation for Research and Technology - Hellas
P.O.Box 1385, GR-71110, Heraklion, Greece
panos@ics.forth.gr

ABSTRACT

In information bases following semantic and object-oriented data models logical names are used for the external identification of objects. Yet the naming schemes employed are not "natural" enough and several problems often arise: logical names can be ambiguous, excessively long, unrelated to or unable to follow the changes of the environment of the named object. In natural language, similar problems are resolved by the context within which words are used. An approach to introducing a notion of context in an information base is to provide structuring mechanisms for decomposing it into possibly overlapping parts. This paper focuses on developing a context mechanism for an information base and, in particular, exploiting this mechanism for naming purposes. Rules are developed for generating meaningful names for objects by taking their context into account. This context-based naming enhances name readability, resolves name ambiguities, saves a lot of redundant name substrings, and it localizes and thus facilitates consistency checking, query processing and update operations. In modeling, it supports systematic naming of objects, and thus enhances cooperation between the designers and the end-users in the sense that the contents of the information base are more understandable by both of them.

Keywords: Context, naming, conceptual modeling, semantic conflicts, information bases

1. Introduction

In natural language, words are used to identify concepts (concrete or abstract) and context is used to essentially enhance the expressiveness of a finite vocabulary. In information bases, in particular those following object-oriented or semantic data models, logical names are used to externally identify objects. Context mechanisms, however, are not offered in general. Introducing context in information bases would enhance expressiveness, understandability and flexibility, and would support cooperative applications, where contextual divisions of the information bases naturally arise.

A general approach to introducing a notion of context in an information base is to provide structuring mechanisms for decomposing it into possibly overlapping parts. This paper focuses on developing a context construct for an information base following an object-oriented semantic data model and, in particular, exploiting this construct for naming purposes.

At the conceptual level, the information base describes objects and relationships between them. The objects are identified by logical names. These names are usually derived from the names of the real world concepts that the objects represent. A variety of problems arise, especially in large databases, such as logical names that are ambiguous, excessively long, unrelated to the environment of the respective objects, or unable to follow the changes of that environment.

An abstract context model for partitioning information bases was proposed and its integration into a powerful object-oriented notation, namely the Telos knowledge representation language,[30] was addressed by Mylopoulos and Motschnig-Pitrik.[31,32] Inspired in large part by that work, as well as by the application requirements of the Semantic Index System, [7,8,9,10,11] in this paper we introduce a context construct for a data model essentially conformant with the structural part of Telos, and we define a context-based naming scheme.

Contexts are defined in a broad sense, in which they may overlap, and in a strict sense, in which they are disjoint. The disjoint strict contexts can be thought of as characterizing the objects they contain in a unique manner, whereas in general an object may be contained in more than one broad contexts. Objects are assigned unique names within their strict contexts. The latter are hierarchically organized, thus enabling the generation of unique path names for all objects in an information base. This scheme implements name relativism not only for relationships, as usual, but also for objects of independent standing. Meaningful object names are generated automatically through the respective contexts. Thus names are dynamic, related with and following the changes of the environment of the objects.

The paper is organized as follows: The next section briefly surveys manifestations of context and naming in different fields. Section 3 reviews the information representation framework. We show the naming problems that appear in this framework and discuss the need for using context. Section 4 defines a refined notion of path, necessary for introducing a versatile path-based name generating scheme. In section 5 the context construct and the context-based naming mechanism are developed. Section 6 discusses issues of using context-based naming. Section 7 addresses some implementation issues and section 8 contains general discussion and conclusions.

2. Naming and context: a brief survey

In linguistics and cognitive psychology the notion of context plays an important role in language comprehension and generation. People naturally use contextual information in order to convey a specific meaning. Langacker claimed that all linguistic units are context-dependent.[26] One of the basic characteristics of natural

language is *ambiguity.*[36] People frequently employ the same form to convey more than one meanings in everyday speech. This is the case of *homonyms*, which are either *homographs* (e.g. 'bat' the mammal or 'bat' the athletic implement) or *polysemous* words (e.g. the word 'line' in 'a line of code', 'a line on the blackboard', or 'a line in the bank'). On the other hand, different forms are often used to convey the same meaning (*synonyms*, e.g. 'cheap' and 'inexpensive'). These ambiguities are resolved by reference to a specific context. For example, the same meaning can be expressed by different words in the dialects of different cultural or professional groups.[23] Conversely, the same word can be used by different groups to convey different meanings, or even within one group to convey different meanings in different situations.

In artificial languages, the opposite of the previous assumption, that is, one meaning can have only one linguistic representation, is also true. This clearly requires using absolute or global 'names'. An interesting question in artificial language design is how to extend a name by suffixing or prefixing constructs in order to make it unique in a given context or, given a name, how to restrict the context so that this name may convey a unique meaning.

There are several manifestations of context-based naming: in programming languages, the parts of the program which are visible to a particular program segment are determined by the *scopes* and *scope rules* using *scope resolution operators.* More recently, in an object-oriented framework, *aspects*,[37] *roles* [17,38,2] and *conceptual slices* (views) [40] have been introduced to support multiple state and multiple behavior of the same objects, which is similar to handling the same object in different context or to moving an object into another context. In traditional databases, *views* present a consistent partition of the database.[16,3] Such mechanisms have been adopted in object-oriented databases [1,29] and semantic data models.[13]

In multidatabase environments and heterogeneous information systems, database integration has to deal with naming conflicts of two types, *homonyms* and *synonyms*, because the global schema of the integrated database is usually generated by merging one or more user-oriented schemas.[6,4] Some significant alternatives for representing context are the *semantic proximity proposal*,[22] where context is defined as a collection of meta-attributes for capturing the semantic similarity between objects, the *context building approach*,[33] where context is defined as the knowledge that is needed to reason about another system, for the purpose of answering a query, and the *context interchange approach*,[39] where context is defined as the meaning, content, organization and properties of data used for exchanging data and general, facilitating semantic interoperability between heterogeneous information systems.

In artificial intelligence contexts have been introduced as means of partitioning knowledge into manageable sets,[19] or they have been considered as logical constructs that facilitate reasoning activities.[27,18] Moreover, *contexts* have been proposed as a partitioning scheme for hypertext databases,[14] and *perspectives* as a mechanism for organizing and manipulating groups of nodes and links in a hypertext network.[35]

Furthermore a context is taken to defines a view of the objects in a repository[5]

and it is typically used to define a set of objects that an engineer is manipulating for a particular task. More recently, general abstractions for partitioning information bases with contexts have been proposed, which address issues of naming conventions, authorization, transaction execution and overlapping contexts.[31,32,28] Other approaches employ context as a way to face the complexity of information base update,[11] or to develop a naming mechanism for semantic data models.[43,44]

3. Information Representation

In this section we discuss the information representation framework, the naming scheme it employs and the problems entailed by that scheme, as well as the benefits of using context as a structuring mechanism in this framework.

In representing information we try to capture some of the important semantic information about the real word and represent it in a model world. Our approach closely follows that of semantic data modeling.[20,34] We assume information bases that follow an object-oriented data model conformant with the structural part of the Telos language.[30,25] This is a generalization of graph-theoretic structures used in semantic networks, semantic data models and object-oriented representations. An information base can be thought of as consisting of nodes and links, with nodes describing objects of independent existence and links representing relationships among such objects. Nodes and links are considered as objects and are organized along three dimensions: *classification* (instance of), *generalization* (isA) and *attribution*. Attributes represent object properties and binary relationships. Attributes of attributes can be defined. Multiple classification is allowed, supporting the separate representation of multiple modeling aspects. Classes of objects within a given instantiation level are also organized in terms of generalization relationships. These can be multiple and give rise to hierarchies that are directed acyclic graphs.

3.1. Naming scheme and naming problems

An *object* is associated with two identifiers: a *system-generated, globally unique identifier*, completely independent of any physical location (a *surrogate*[12]), which distinguishes it internally from all other objects, and an *atomic logical name* which supports logical reference to the object and identifies it externally. The logical name of a node is unique over the entire information base, and the logical name of a link is unique among the links emanating from the same object. This results in the naming of nodes being different from the naming of concepts in the real world. The latter are often named taking into account a specific context.

An example is illustrated in Fig. 1, where the information base represents articles and their structure. The nodes are depicted by boxes and the links by directed arrows. The text inside a box or above an arrow corresponds to the atomic logical name, and the text in square brackets to their internal object identifier. The class of articles (object x_{20}) is represented as a specialization of the class of documents and as a composite object consisting of a body, the body consisting of sections and

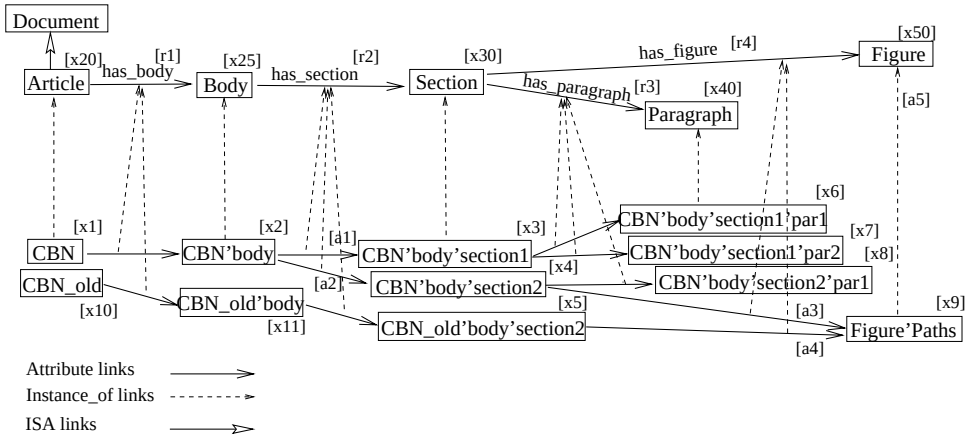

Fig. 1. Information base example.

sections consisting of paragraphs and figures. The objects x_1 and x_{10} represent two specific articles named *CBN* and *CBN_old* respectively. In order for their bodies (objects x_2 and x_{11}) to be externally identified uniquely, they are named *CBN'body* and *CBN_old'body*, that is by prefixing the name of the composite object each of them belongs to. The same naming convention is used to name sections and paragraphs of the articles, but not figures. So, as we can see, names are composed in an arbitrary manner. This produces several problems in communication between designers (who name objects) and users (who query the information base). Moreover, these composite names become excessively long, especially in large databases carrying redundant information, being inflexible in use and difficult in maintainance. If, for example, the name of the object x_1 were changed from *CBN* to *CBN_new*, all nested components (e.g. $x_2, x_3, x_4, \ldots$) which include the substring *CBN* would remain inconsistent. This can happen quite often in applications involving update operations.

Therefore, naming schemes like the one above are not "natural" enough and problems can arise. In particular, names turned out to be:

- **Ambiguous**, in the sense that names can be homonyms (e.g. section 2 of *CBN* and section 2 of *CBN_old*), or synonyms.
- **Excessively long** in order to resolve ambiguities. This results in composite names with redundant name substrings (e.g. *CBN* is redundantly repeated) and without having a systematic manner of composition (this is left to the designer's preference and perception of the concept being represented).
- **Inconsistent**, i.e. unrelated to or unable to follow the changes of the environment of the named object.

3.2. Benefits of using context

In order to resolve the above naming problems we need to capture the important semantics of relationships between objects and their environment. These environments can be thought of as *semantically meaningful partitions* of an information base. "Semantically meaningful" means that these partitions represent real world divisions such as background knowledge, personal beliefs, time periods or geographical areas, and so on. These partitions are referred to as *contexts*.

Representing context in information bases contributes to the efficient handling of information. Shoham [41] has discussed the benefits of representing context in artificial intelligence and knowledge-based systems including: *economy of representation* (context can be act as a focusing mechanism when accessing objects), *economy of reasoning* (reasoning within the context scope rather than the entire information base), *handling inconsistent information* (information is consistent within the context of the user query), and *flexible semantics* (the same object can have different representations in different contexts or different objects can have the same representation (e.g. the same name) in different contexts).

In this paper we define a partial representation of context by partitioning the information base into possibly overlapping conceptual slices. Based on context we build a naming mechanism resolving most of the naming problems discussed above. Moreover, most of the context benefits applying to naming are obtained. Specifically, contexts are defined over the information base and a given node may belong to several contexts. Our naming mechanism relaxes the unique node name assumption by assigning unique node names with respect to one or more contexts. Yet, we assume that one of these contexts can always be selected as more characteristic for a given node in a specific information base, and for naming purposes the node is associated exclusively with that context. Correspondingly, we assume that each of the real-world concepts modeled by nodes can be characterized uniquely by one of the real-world contexts of the domain of discourse. Thus, contexts are hierarchically organized enabling the use of path name expressions for object identification. In the next section the notions of path and path name expression are discussed in detail.

4. Paths and Names

In this section we introduce a refined notion of path, that explicitly differentiates between link classes, and corresponding notions of navigation and path names.

We use the symbols $\mathcal{O}$ for the set of objects, $\mathcal{ND}$ for the set of nodes, $\mathcal{A}$ for the set of links ($\mathcal{A} : \mathcal{O} \longrightarrow \mathcal{O}$). It holds that: $\mathcal{O} = \mathcal{ND} \cup \mathcal{A}$. We use $\mathcal{R}$ for the set of binary relations (classes of links, $\mathcal{R} \subseteq \mathcal{A}$), and $\mathcal{L}$ for the set of atomic names. The functions $from\,(a)$ and $to\,(a)$ give the object which the link a emanates from and points to, respectively. The function $inst\,(o)$ gives the set of instances of class $o \in \mathcal{O}$. We use the expression $link\,(x, y, r)$ (or $!link\,(x, y, r)$) to denote that there exists at least (or only) one link between the nodes x and y (either in the

forward or backward direction, i.e. $link\,(x,y,r) = link\,(y,x,r)$), which is instance of link class r. We use the expression $conn\,(a,x,y)$ to denote that link a connects the nodes x and y. As mentioned earlier, objects can have an atomic name. The binding of atomic names to objects is determined by the *atomic name function* $L : \mathcal{O} \longrightarrow \mathcal{L}$. Thus, for our purposes an information base ($\mathcal{IB}$) is defined as a triple: $\mathcal{IB} \equiv (\mathcal{O}, \mathcal{L}, L)$. Formal definitions of the expressions $link\,(x,y,r)$ and $conn\,(a,x,y)$ are given in the Appendix A. To improve the readability of this paper, all the sets, functions and expressions has been collected in different tables in the Appendix C.

Accessing information in such an information base often involves navigating from one object to another by following links (in the forward or backward direction).[24] Navigation relies on the notion of *path*; It begins at the first node of the path and reaches the last one following links of specific classes and passing through (visiting) specific nodes in a given order. The main difference between our notion of path and the dot-separated path expressions in object-oriented systems[24] is that we focus on the type of navigation (navigation through link classes) and not on the navigation itself (navigation through a specific link). The orientation of links is disregarded, since for every link an inverse one can be defined (e.g. "a Car includes an Engine" and "an Engine is part of a Car" describe the same real-world situation).

Definition 4.1 Path. *The set of paths $\mathcal{P}$ consists of finite sequences of "dot separated" nodes. Every two successive nodes must be connected to each other with at least one link. One of the link classes to which a path link belongs is used to characterize the link and appears as separator. The orientation of the link is disregarded.*

$$\mathcal{P} = \{x_1.[r_1].x_2.\ \cdots\ .x_{n-1}.[r_{n-1}].x_n \mid n \in I\!N_0 \ \wedge\ (\forall 1 \leq i \leq n,\ x_i \in \mathcal{ND}) \ \wedge\ (\forall 1 \leq i < n, link\,(x_i, x_{i+1}, r_i))\}.$$

The set of paths includes an identity path denoted as `Id`. *This does not correspond to any path in the information base, but is used to refer to empty paths.*

In Fig. 1, the path $p' = x_1.[r_1].x_2.[r_2].x_3$ denotes the navigation from the article "CBN" to the first of its sections and the path $p'' = x_4.[r_2].x_2.[r_1].x_1$ denotes the navigation from the second section to the article it belongs to.

A path can be composed by other paths as follows: $p_1 \odot_r p_2 \equiv p_1.[r].p_2$ where $p_1, p_2 \in \mathcal{P}$ and $r \in \mathcal{R}$. The functions $Root(p)$ and $Leaf(p)$ give the first and the last node of path p, respectively. In our example $Root(p') = x_1$ and $Leaf(p') = x_3$. The *length* ($len : \mathcal{P} \longrightarrow I\!N_0$) of a path is defined as the number of nodes contained in the path (e.g. $len(p') = 3$). Formal definitions of composition and length are given in the Appendix A.

Reaching a node requires either to explicitly specify the node (by its identifier) or to navigate the information base through a path leading to that node. The set of paths reaching a specific node x, $\mathcal{P}(x)$, is defined as follows: $\forall x \in \mathcal{ND} : \mathcal{P}(x) = \{p \in \mathcal{P} \mid Leaf(p) = x\}$. For example, in Fig. 1 we have:

$$\mathcal{P}(x_9) = \{x_9,\ x_4.[r_4].x_9,\ x_5.[r_4].x_9,\ x_2.[r_2].x_4.[r_4].x_9,\ x_1.[r_1].x_3.[r_2].x_4.[r_4].x_9,\ x_{11}.[r_2].x_5.[r_4].x_9,\ x_{10}.[r_1].x_11.[r_2].x_5.[r_4].x_9,\ x_{50}.[instof].x_9\}$$

(*instof* represents the internal identifier of the *instance-of* relation).

The communication between user and information base is effected through logical names. Therefore, a path should be referred externally by a logical name. A *logical name* (or simply *name*) is defined as a sequence of atomic names that can occur in a database, or that are allowed in a term of a query language. Since the atomic name of a link is unique only among the links emanating from the same object, a link is not fully identified externally by its atomic name. Full identification is achieved by extending its atomic name with the name of its from-object. In general, link names can be defined as follows:

Definition 4.2 Link name. *The set of link names $\mathcal{N}_A$ consists of finite sequences of "dot separated" atomic names: $\mathcal{N}_A = \{l_1. \ \cdots \ .l_n \mid n \in I\!N_0 \ \wedge \ l_1, \ \cdots \ , l_n \in \mathcal{L}\}$. This set includes an identity name, denoted* `Id`*, which refers to names of zero length.*

The binding of arbitrary link names to links is determined by the *link name function*.

Definition 4.3 Link name function. *The link name function $N_a : \mathcal{A} \longrightarrow \mathcal{N}_A$ assigns a name to each link.*

$$\forall a \in \mathcal{A}: \ N_a(a) = \begin{cases} L\left(from\left(a\right)\right).L\left(a\right) & \text{if } from\left(a\right) \in \mathcal{ND} \\ N_a(from\left(a\right)).L\left(a\right) & \text{if } from\left(a\right) \in \mathcal{A} \end{cases}$$

The link name function is defined recursively because links can emanate not only from nodes but also from other links. In our example, $N_a(r_1) = Article.has\_body$ and $N_a(r_2) = Body.has\_section$.

A node, on the other hand, can be accessed not only by its atomic name (which is, so far, a unique external identifier), but also by a path. Each path can be denoted externally by a name. The *path name* is defined as follows:

Definition 4.4 Path name. *The set of path names $\mathcal{N}_P$ is*

$$\mathcal{N}_P = \{l_1.[r_1].l_2. \ \cdots \ .l_{n-1}.[r_{n-1}].l_n \mid n \in I\!N_0 \ \wedge \\ l_1, \ \cdots \ , l_n \in \mathcal{L}, r_1, \ \cdots \ , r_{n-1} \in \mathcal{N}_A\}.$$

We suppose, as mentioned before, that for every (forward) relationship there is an inverse one which represents the same situation. That is, for each relationship r there exists the inverse r^{-1} with atomic logical name $L\left(r^{-1}\right)$. With this in mind, we define the *path name function*, which provides the binding of path names to paths.

Definition 4.5 Path name function. *The path name function $N_p : \mathcal{P} \longrightarrow \mathcal{N}_P$ associates a name with each path.*

$\forall p, p_1 \in \mathcal{P}, \ x \in \mathcal{ND}, \ r \in \mathcal{R}:$

$$N_p(p) = \begin{cases} \texttt{Id}, & p = \texttt{Id} \\ L\left(x\right), & p = x \\ L\left(x\right).[N_a(r)].N_p(p_1), & p = x \odot_r p_1 \ \wedge \ x \in inst\left(from\left(r\right)\right) \\ L\left(x\right).[N_a(r^{-1})].N_p(p_1), & p = x \odot_r p_1 \ \wedge \ x \in inst\left(to\left(r\right)\right) \end{cases}$$

For example, in Fig. 1, the name of the trivial path x_1 is $N_p(x_1) = CBN$ and the name of path $x_1.[r_1].x_2$ is $N_p(x_1.[r_1].x_2) = CBN.[Article.has\_body].CBN'body$. On the other hand, a path name which contains an inverse relationship name is $N_p(x_{50}.[instof].x_9) = Figure.[classof].Figure'paths$ (note that $L\left(instof^{-1}\right) = classof$).

Under certain circumstances a path name can characterize and identify a node uniquely. Then, this name can be considered as the external identification of the node. In the next section we define a mechanism which provides the necessary environment (context-based) for uniquely identifying nodes by path name expressions.

5. Context and Naming

We now introduce the notion of context and define a context-based naming scheme. Our aim is to relax the unique node name assumption for nodes, so that different nodes can be named with the same atomic name. On the other hand, we have to ensure that there is always a unique external identifier for each object. These are the *relative* and *global* names of the object, which are composite and generated by taking into account the context which the object belongs to. Meaningful node names are produced automatically, by using the connections of the object with its environment. Thus names can be frugal and dynamically follow the changes of the environment. Contexts can be considered as conceptual slices of the information base. A node can be contained in more than one contexts, yet we shall make the working hypothesis that a single context can best characterize each node. Thus, in the naming scheme developed here, contexts are considered as disjoint.

5.1. The notion of context

Definition 5.6 Context. *A context is a pair consisting of a node and a link class. Thus, if Cxt is the set of contexts then $Cxt \subseteq \mathcal{ND} \times \mathcal{R}$.*

We define two element selector functions on contexts: the first to retrieve the node ($nd : Cxt \longrightarrow \mathcal{ND}$) and the second to retrieve the link class of a context ($lc : Cxt \longrightarrow \mathcal{R}$). Hereafter, the node and the link class of a context will be referred to as *pivot elements* of that context.

Definition 5.7 Contents of a context. *The contents of a context is a part of the information base consisting of objects (nodes and links). The function $Cnts$ associates with each context its contents: $Cnts : Cxt \longrightarrow P(\mathcal{O})$*.

Contexts are defined in a broad sense, in which their contents may overlap, and in a strict sense, in which their contents are disjoint. Correspondingly, we distinguish the contents of a context into *broad* and *strict contents*. The broad contents of a context c are created by the constructor function $brCnts$ as follows:

$$brCnts(c) = pivotelements\,(c) \cup neighbors\,(c)$$
$$pivotelements\,(c) = \{nd\,(c)\,, lc\,(c)\}$$

*P stands for powerset.

$$neighbors(c) = NDneighbors(c) \cup Aneighbors(c)$$
$$NDneighbors(c) = \{x \in \mathcal{ND} \mid link(nd(c), x, lc(c))\}$$
$$Aneighbors(c) = \{a \in \mathcal{A} \mid \forall x \in NDneighbors(c) : conn(a, nd(c), x)\}.$$

We can see that the broad contents of a context $c = (y, r)$ consist of its *pivot elements* (node y and link class r) and its *neighbors*. Neighbors are distinguished into *node neighbors* ($NDneighbors(c)$) and *link neighbors* ($Aneighbors(c)$). The former is the set of all nodes connected with pivot node y through links which are instances of pivot link class r and the latter is the set of all those links.

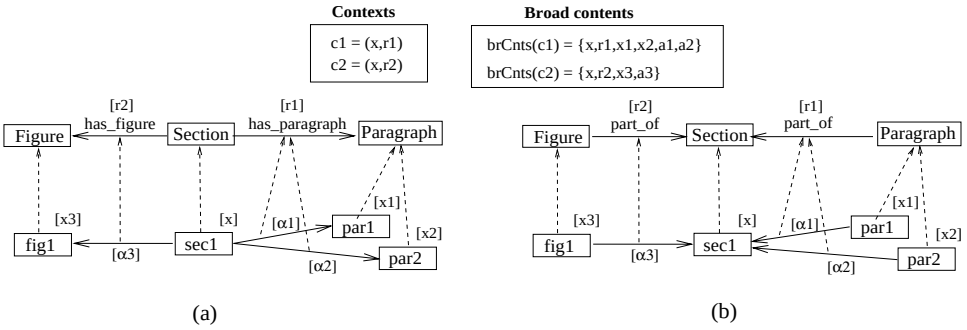

Fig. 2. Broad context.

For example, in Fig. 2, there are two contexts defined by the same pivot node x: $c_1 = (x, r_1)$ and $c_2 = (x, r_2)$ in both information bases (a) and (b). The broad contents of those contexts are: $brCnts(c_1) = \{x, r_1\} \cup \{x_1, x_2, a_1, a_2\}$ and $brCnts(c_2) = \{x, r_2\} \cup \{x_3, a_3\}$. The first context semantically represents all the possible paragraphs of the first section and the second all the possible figures. Fig. 2(a) and 2(b) represent the same real word situation. In Fig. 2(a) we use *has_part* relations (*has_paragraph*, *has_figure*) and in Fig. 2(b) *part_of* relations to represent that a section may consist of paragraphs or figures. On the other hand, contexts and their contents are the same in both cases, because they represent environmental information which remains the same regardless of the specific representation.

Broad contexts are possibly overlapping in the sense that their contents may contain common neighbors. For example, in Fig. 3, contexts c_1 and c_2 overlap containing in common the neighbor node x_2. Semantically the overlapping node x_2 represents a paragraph shared by sections 1 and 2 of two different articles.

Strict contexts, on the other hand, are defined to be disjoint. The strict contents of a context are subset of the corresponding broad contents, that is: $\forall c \in Cxt : strCnts(c) \subseteq brCnts(c)$, and they are specified by the *scope function* in a way such as to avoid overlapping among the neighbors of different contexts.

Definition 5.8 Scope function. *A scope function* $S : \mathcal{ND} \longrightarrow Cxt$ *associates*

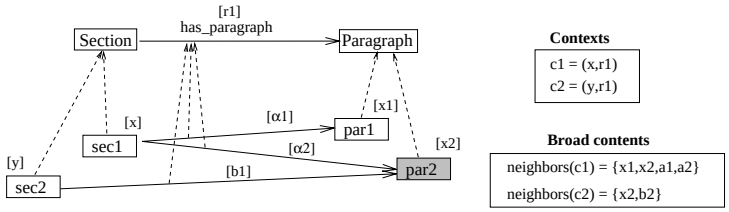

Fig. 3. Overlapping broad contexts.

with each node a context and denotes that the node is in the strict contents of that context.

The constructor of strict contents is identical with the constructor of broad contents except for the definition of node neighbors, which are defined as follows:

$$NDneighbors\left(c\right) = \{x \in \mathcal{ND} \mid link\left(nd\left(c\right), x, lc\left(c\right)\right) \wedge S\left(x\right) = c\}.$$

Strict contexts are organized in a tree hierarchy in a sense that neighbor nodes of contexts may be pivot nodes for other contexts. The root of all contexts is the *information base context*, which is defined as: $IBcontext = (individual, instof)$ and is built in the information base. By default, all nodes belong to information base context.

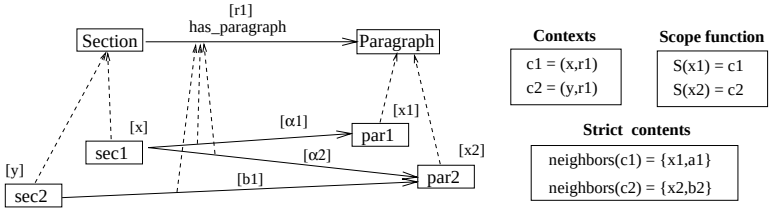

Fig. 4. Strict context.

For example, Fig. 4 shows the information base of Fig. 2 with a scope function defined over it. By virtue of that scope function, $S\left(x_1\right) = c_1$ and $S\left(x_2\right) = c_2$. The strict contents of the contexts c_1 and c_2 are $strCnts(c_1) = \{x, r_1\} \cup \{x_1, a_1\}$ and $strCnts(c_2) = \{y, r_1\} \cup \{x_2, b_1\}$, respectively. By contrast to broad contexts, strict contexts c_1 and c_2 do not overlap. Node x_2 belongs to strict context c_2, which means that it is better characterized by that context (possibly paragraph 2 has originally been published in section 2 of the respective article).

Because of the hierarchical organization of strict contexts, the scope function can be applied recursively on a node.

Definition 5.9 Recursive scope function. *The scope function is defined recursively as follows:*

$$\forall i \in \mathbb{N}_0,\ x \in \mathcal{ND}:\ S^i(x) = \begin{cases} \mathtt{Id} = (\mathtt{Id}, \mathtt{Id}), & i = 0 \\ S(x), & i = 1 \\ S\left(nd\left(S^{i-1}(x)\right)\right), & otherwise \end{cases}$$

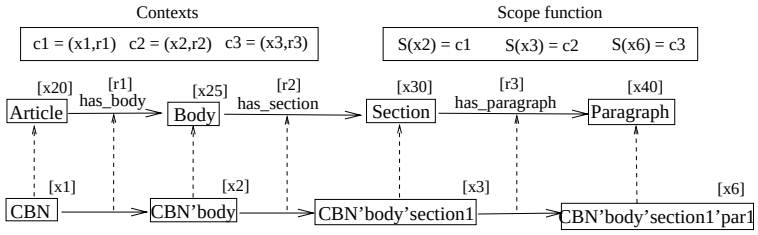

Fig. 5. Recursive context definition

For example, in Fig. 5 contexts has been defined recursively. According to the table of values at the top of the figure, $S(x_2) = c_1$, $S(x_3) = c_2$, $S^2(x_3) = S(nd(c_2)) = c_1$. It also holds that: $S^3(x_6) = c_1$ and $S^4(x_6) = IBcontext$.

Hereafter, we use the words context and contents instead of strict context and strict contents, respectively. We also say that a node x *is included in* a context c iff $\exists i \in \mathbb{N}:\ S^i(x) = c$.

5.2. Naming using context

Definition 5.10 Context-based naming mechanism. *A context-based naming mechanism $\mathcal{CNM}$ is defined over an information base if it obeys the following axioms 5.1 - 5.4.*

Axiom 5.1 *There is a unique relationship between the pivot node and each of the neighbor nodes of a context:*

$$\forall x \in \mathcal{ND}, c \in Cxt:\quad x \in NDneighbors(c) = c \Rightarrow\ !link(nd(c), x, lc(c)).$$

Axiom 5.2 *Every node is included in the information base context.*

$$\forall x \in \mathcal{ND}\quad \exists k \in \mathbb{N}:\ S^k(x) = IBcontext.$$

Axiom 5.3 *The atomic names of nodes are unique with respect to their context.*

$\forall x_1, x_2 \in \mathcal{ND}, c \in Cxt:$

$$x_1 \in NDneighbors(c)\ \wedge\ x_2 \in NDneighbors(c)\ \wedge\ L(x_1) = L(x_2)\ \Rightarrow x_1 = x_2.$$

Axiom 5.4 *Every node belongs to one context (this is ensured by the definition of scope function).*

$\forall x \in \mathcal{ND}, c_1, c_2 \in Cxt:$

$$x \in NDneighbors(c_1)\ \wedge\ x \in NDneighbors(c_2) \Rightarrow c_1 = c_2.$$

This axiom is a strict interpretation of our assumption that one of the contexts can always be selected as more characteristic for a given node in an information base.

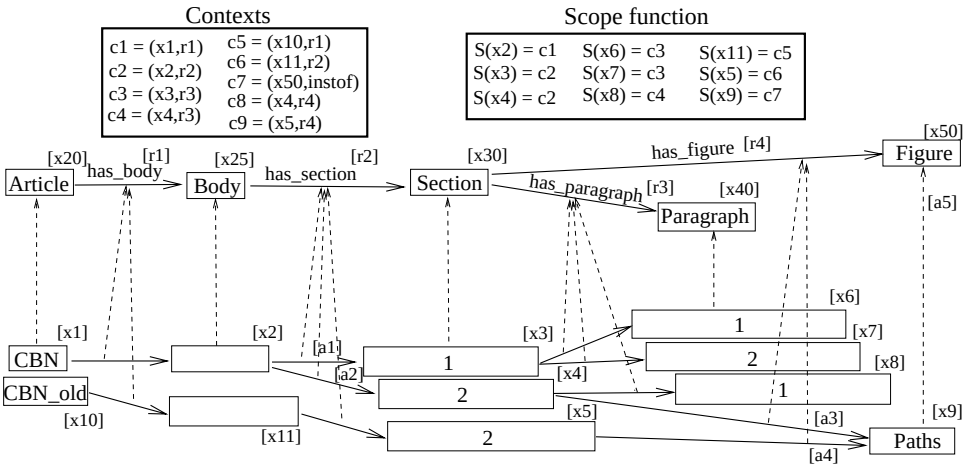

Fig. 6. Context and naming: an example

The information base of Fig. 6 satisfies the axioms of the context-based naming mechanism. The nodes $x_1, x_{10}, x_{20}, x_{25}, x_{30}, x_{40}, x_{50}$ belong to the information base context. On the other hand the nodes x_2 and x_3, which are components of the composite objects x_1 and x_2 belong to the contexts c_1 and c_2, respectively. Their atomic names are unique among the node neighbors contained in these contexts. This is the reason why the atomic names of these two objects are much shorter than their counterparts in Fig. 1. Comparing Fig. 6 with Fig. 1 we observe that in Fig. 6 not only the atomic names of the nodes are much shorter than in Fig. 1 but also the same atomic name is used more than once (homonyms).

A straightforward question is "how could we externally refer to nodes with the same atomic name?". This can be achieved using the *global* and *relative* name of a node, which are path name expressions through context hierarchy. These paths are referred to as *scope paths*, and denote navigations within contexts and are formally defined as follows:

Definition 5.11 Scope paths. *For every $\mathcal{CNM}$ and node x, we define the set of paths ending at x as follows:*

$$\begin{aligned}\forall x \in \mathcal{ND} : \mathcal{SP}(x) = \{p \in \mathcal{P}(x) \mid\ & \exists n \in I\!N : \\ & p = nd\,(c_n)\,.[lc\,(c_n)].\ \cdots\ .nd\,(c_1)\,.[lc\,(c_1)].x, \\ & \forall 1 \leq i \leq n, c_i = S^i(x)\}.\end{aligned}$$

For example, in Fig. 6 we have $\mathcal{SP}(x_9) = \{x_9,\ x_{50}.[instof].x_9\}$ and $\mathcal{SP}(x_3) = \{x_3,\ x_2.[r_2].x_3,\ x_1.[r_1].x_2.[r_2].x_3\}$.

Intuitively, by combining axiom 5.4 with definition 5.11 it is easy to see that there is a unique way to navigate from one node to another within contexts (following scope paths). For example, in Fig. 6, to navigate from x_1 or x_2 towards x_3 within contexts, there is a unique way (the scope path $x_1.[r_1].x_2.[r_2].x_3$ or $x_2.[r_2].x_3$, respectively). It can be proved that the scope paths of a node are acyclic. This ensures that contexts are disjoint and organized in a tree hierarchy. The proofs of all theorems that follow are given in the Appendix B.

Theorem 5.1 *Scope paths do not include cycles. That is:*

$$\forall x \in \mathcal{ND},\ \nexists i \in I\!N :\ nd\left(S^i(x)\right) = x.$$

Every scope path is assigned a specific name by the path name function, called *scope name*. The set of all scope names of a specific node x ($\mathcal{SN}(x)$) are defined as follows: $\mathcal{SN}(x) = \{N_p(sp) \mid sp \in \mathcal{SP}(x)\}$. The scope names identify the terminal nodes of the respective scope paths uniquely, either over the whole information base (global names) or with respect to a context (relative names).

Definition 5.12 Relative name. *The relative name of a node with respect to a context is the name of the scope path which ends at that node and its root node belongs to the neighbors of that context. It is defined as follows:*

$\forall x \in \mathcal{ND}, c \in Cxt, rn \in \mathcal{N}_{\mathcal{P}},\ \exists rp \in \mathcal{SP}(x)$:

$$RelN(x,c) = rn \Leftrightarrow rn = N_p(rp) \wedge Root(rp) \in NDneighbors\,(c)\,.$$

For example, in Fig. 6 the relative name of the node x_3 with respect to the context c_2 is '1', and denotes the fact that object x_3 represents the first section in the body of article 'BCN'.

Definition 5.13 Global name. *The global name of a node is the relative name of the node with respect to the information base context.*

$$\forall x \in \mathcal{ND} :\ GlobN(x) = RelN(x, IBcontext).$$

For example, in Fig. 6 the global name of the node x_1 is its atomic name 'CBN', of x_2 is '$CBN.[has\_body].$', which represents the body of the article named CBN and of x_3 is '$CBN.[has\_body]..[has\_section].1$', which represents the first section of that article. The global name of node x_9, which belongs to the context $(x_{50}, instof)$, is '$Figure.[classof].paths$'.

Theorem 5.2 *The relative name of a node with respect to a context is unique within that context. That is:*

$$\forall x, x' \in \mathcal{ND}, c \in Cxt :\ RelN(x,c) = RelN(x',c) \Leftrightarrow x = x'.$$

This theorem is very important, because it ensures that each node of the information base has a unique/unambiguous external identification with respect to a specific context. Therefore, naming ambiguities can always be resolved by either extending the name or restricting the context.

A specific context can be externally referred to by its name defined as follows:

$$\forall c \in Cxt : CxtN(c) = \left(GlobN(nd\,(c)), N_a(lc\,(c))\right).$$

For example, the name of context (x_1, r_1) in Fig. 6 is $(CBN, Article.has\_body)$.

6. Using Context-based Naming

6.1. Name consistency

Node names (relative or global) are constructed taking into account the node context and therefore the environment which is considered more characteristic for it. Consequently, if the environment changes (by updating the information base) the name will change accordingly. For example, if the atomic name of node x_1 in Fig. 6 changes from 'CBN' to 'new_CBN', the global names of nodes representing the body, sections and paragraphs of that article will adapt to that change (e.g. the global name of node x_2 will change to '$new\_CBN.[has\_body]$.'). Therefore, node names follow the updates of the information base and remain consistent with the real world concept being represented by that node.

6.2. Queries

6.2.1. Flexibility

A query may be ambiguous, because names used in formulating the query may refer to more than one contexts of the information base. Take, for example, a query for 'clubs'. This may refer to different contexts and return the union of very different queries such as: society of people (youth, tennis, golf clubs) or playing-cards, or even the headed sticks used to hit the ball in golf (see Fig. 7).

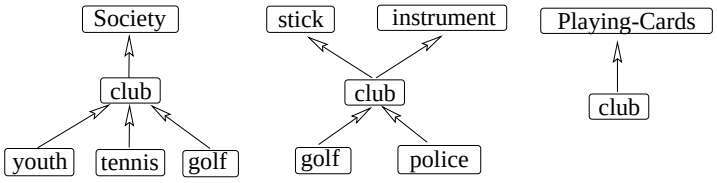

Fig. 7. Ambiguous names.

It seems more likely that the user had a single context in mind, but did not specify that context well enough. One way for the user to resolve the ambiguous query is to either extend the ambiguous names or to restrict the context. So, for example, the user can extend the name '$club$' to '$Society.[superclass\_of].club$' or restrict the context from $IBcontext$ to $(Society.[superclass\_of].club, superclass\_of)$. Our mechanism provides this flexibility.

6.2.2. Expressiveness

Context-based naming enhances the query expressiveness since not only the

queries can be posed with respect to a specific context but also the query results can be adapted to the user context by naming the returned object with respect to that context. For example, in Fig. 7, the user can specify a context, e.g. '*Society*', and make queries with respect to that context, e.g. query for '*clubs*'. The results, ('*youth*', '*tennis*', '*golf*') have been adapted to the user context.

Moreover, if the context of the query was not specified well enough, the system would able to guide the user (in a interactive mode) by suggesting the relevant context referred by the query. Systems employed global naming schemes can not support queries involving contextual information. For example, a query for clubs would fail in these systems, whereas, the same query in our system would result either in all possible clubs and their context or to inform the user about all the contexts that refer to clubs (Society, Instrument, Playing-card) in order to help the user to specify its context in mind.

6.3. Context types

Contexts are distinguished into different types according to the pivot link class. In our model there are three different context types each of which carries different semantics. These are:

(i) *classification context* (pivot link class is the instance of relation).
(ii) *generalization context* (pivot link class is the ISA relation).
(iii) *attribution context* (pivot link class is the attribute relation).

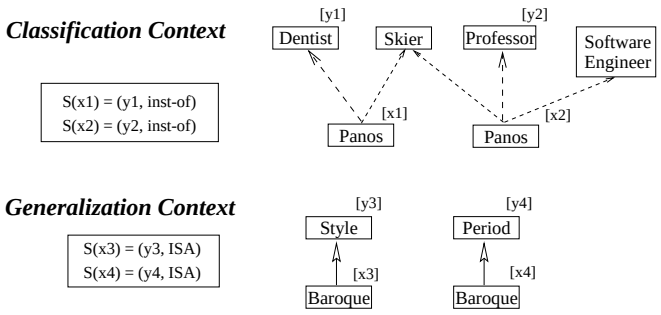

Fig. 8. Context types: classification and generalization.

In classification context, node classes characterize uniquely node instances. An example is illustrated in Fig. 8, where classes represent different roles and object x_1 represents person Panos better characterized as being dentist (object x_1 belongs to the context of class $Dentist$), whereas object x_2 represents another person Panos better characterized as being professor (object x_2 belongs to the context of class $Professor$).

On the other hand, in generalization context, node superclasses characterize

uniquely node subclasses. In Fig. 8, the homonymous terms Baroque are disambiguated by the context of their superclasses. Thus the term Baroque can refer to the style or to the period.

Attribution context uses attribute relations to define context (an example is illustrated in Fig. 6).

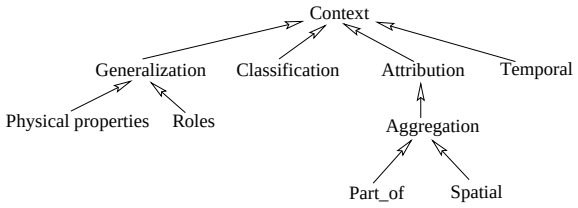

Fig. 9. Hierarchy of context types.

All the context types can be further divided into context subtypes. For example, attribute context can be divided into *part_of* contexts, useful for representing composite objects, *spatial* or *temporal* contexts. These contexts use pivot link classes which carry relevant semantics. An indicative context type/subtype hierarchy is shown in Fig. 9. This hierarchy can be enriched by systematically analyzing the different types of context appearing in the real world, but this is not in the scope of this paper.

Classification, generalization and part_of context can be useful in handling homonyms in thesauri systems.[21,42]

Different context types may employ different naming generation/formulation conventions in order to compose relative or global names for objects. Even the same context type may use different naming methods in different languages in order to generate names more close to natural language.

7. Implementation Issues

Our mechanism can be easily incorporated into a structurally object-oriented information base system. The only extra requirements are to save the scope function information and to check the necessary integrity constraints. This can be easily achieved by extending the structure which saves the atomic name function (L) so as to capture contextual information. Suppose that the atomic name function is implemented by a table, called *symbol table*, comprising two columns: the first for the internal identification of an object and the second for its atomic name (see Fig. 10). For each object a row is created in the symbol table and saves the internal identifier and the atomic name of that object (each row represents a specific relationship of the atomic name function). For the purposes of the context-based naming mechanism this structure can be extended to the structure shown in Fig. 11.

OID	atomic name
x_1	CBN
x_2	
x_3	1

Fig. 10. Implementation: atomic name function.

OID	atomic name	Context nd element	Context lc-element
x_1	CBN	individual	instof
x_2		x_1	r_1
x_3	1	x_2	r_2

Fig. 11. Implementation: extended atomic name function including contextual information.

Another implementation approach for context-based naming is to create a different symbol table for each context consisting of the contents of that context and their atomic names.

The first approach is easier to implement but requires some extra time to compute the name of an object or to access an object identified by a path name, because it has to look up the whole symbol table. On the other hand the second approach needs some extra structures to be implemented and requires extra memory and address space to save the different local symbol tables but it localizes and thus facilitates consistency checking, query processing and update operations.

To validate our mechanism, we have started a prototype implementation within the Semantic Index System (SIS) based on the first approach. The SIS[8,9] is a system for the management of very large collections of highly interrelated information objects with evolving structures. The SIS data model follows the structural part of the Telos knowledge representation language.[30]

8. Discussion and Conclusion

The objective of this paper was to deal with the question of naming in information bases, where several problems appear: ambiguous logical names, excessively long and difficult to handle, unrelated to the environment of the named object, or unable to follow changes of that environment. As a solution to this problem, we proposed a context-based naming mechanism.

Our approach in a sense simulates naming in natural language: a notion of context is supported and objects can be identified externally by using logical names with respect to a specific context (relative names). This results in the use of polysemous and homonymous names, which is common in natural language. Names are generated dynamically by taking into account the context of the corresponding object. Moreover, the generated names are meaningful, which contributes to the

quality of a limited natural language generation, useful for developing hypermedia applications over information bases[10], and data entry facilitation. Object identification techniques like extending a name in a specific context or restricting the context of a specific name are also supported. All these contribute important quality factors to modeling, like *simplicity*, *flexibility* and *expressiveness*. The capability of using relative names ensures the frugality of atomic names thus saving a lot of redundant name substrings. Our experience from large modeling applications indicates that as many as 60% of the names in some applications (such as the CLIO cultural documentation system[7]) can be redundant. In addition, the unnamed objects can be exploited by higher level update operations which facilitate the interactive creation of composite objects[11,45].

The names, on the other hand, are composite and can be extended or restricted dynamically in order to identify the desired object uniquely. Moreover, they are consistent with the environment of the named object. This minimizes the *maintainance cost* of names, because names follow the updates (renaming, deletion, context switching, etc.) of the information base, which is very important for evolutionary applications. It also enhances the *visibility* and *readability* of the names making them more understandable to the users. This is important for the development of interactive applications. In addition, the use of atomic, relative or global names enhances query expressiveness, since both the formulation and the results of a query can include atomic, relative or global names.

An important feature of this mechanism is that it can be easily incorporated into an information bases system. The only extra requirements are to save the scope function information and to check the necessary integrity constraints. Moreover, the absence of redundant name substrings facilitates string matching algorithms to run more efficiently. The price is some extra time to compute the name of an object or to access an object identified by a path name. We can cope with this problem by using a local symbol table (interpretation of atomic name function) for the contents of each context. This localizes and facilitates *consistency checking*, *query processing* and *update operations*.

For modeling purposes the context-based naming mechanism is suitable for representing composite objects and managing complex conceptual structures found in many advanced applications such as scientific catalogs, CAD, manufacturing and software development. It is also useful for terminological bases, which are usually large and exhibit a lot of naming ambiguities, and for evolutionary applications.

Future research includes dealing with *conjunctive contexts* (a node can be characterized in conjunction by more than one contexts) and *disjunctive contexts* (a node can be characterized by one of several contexts) relaxing the assumption that a node belongs to only one context. Disjunctive contexts will generate names which are synonyms for a given node. In heterogeneous environments, separate databases can be thought as disjunctive contexts; sharing of information in these environments is the common objects which can be maintained (e.g. named) in different ways within different disjunctive contexts. Also, overlapping contexts and intercontext

communication need attention.

Acknowledgments: We would like to thank Anastasia Analyti for carefully reading the paper and for her valuable comments, Martin Dörr for many stimulating conversations on the subject, and Yannis Tzitzikas and Polivios Klimathianakis for their contribution to this work. We would like also to thank the anonymous referees for their helpful comments.

Appendix A: Detailed specifications of path expressions

Definition A.1 Path composition.

$$\forall p_1, p_2 \in \mathcal{P}, r \in \mathcal{R}: \ p_1 \odot_r p_2 \equiv \begin{cases} p_1, & \text{if } p_2 = \texttt{Id} \\ p_2, & \text{if } p_1 = \texttt{Id} \\ p_1.[r].p_2, & \text{otherwise.} \end{cases}$$

Definition A.2 Path equality.

$$\forall p, s, p', s' \in \mathcal{P}, \ x \in \mathcal{ND}, \ r \in \mathcal{R}: \ p = s \Leftrightarrow \begin{cases} p = s = \texttt{Id}, \\ p = x \odot_r \ p' \ \wedge \ s = x \odot_r s' \ \wedge \ p' = s' \end{cases}$$

Definition A.3 Length of a path. The *length* of a path is the number of nodes occurring in the path ($len : \mathcal{P} \longrightarrow \mathbb{N}_0$).

$$\forall p, p' \in \mathcal{P}, \ x \in \mathcal{ND}: \ len(p) \equiv \begin{cases} 0, & p = \texttt{Id}, \\ 1 + len(p') & p = x \ \odot p' \end{cases}$$

The *composition*, *length* and *equality* of names are defined like those of paths. The composition operator for names is the symbol $\oplus$ and the following identity is true: $N_p(p_1 \odot_r p_2) = N_p(p_1) \oplus_r N_p(p_2)$. Clearly, $\mathcal{P}$ and $\mathcal{N}_A$ is closed under the composition operator, and the operator itself is associative, that is: $p_1 \odot_{r_1} (p_2 \odot_{r_2} p_3) = (p_1 \odot_{r_1} p_2) \odot_{r_2} p_3$. Associativity allows us to omit an indication of precedence in expressions with more than one instance of the operator. The following identities on *len* is also a straightforward consequence of our definitions: $\forall p_1, p_2 \in \mathcal{P}, r \in \mathcal{R}: \ len(p_1 \odot_r p_2) = len(p_1) + len(p_2)$ and $\forall n_1, n_2 \in \mathcal{N}_A, r \in \mathcal{R}: len(n_1 \oplus_r n_2) = len(n_1) + len(n_2)$.

We give the following definitions: $\forall x, y \in \mathcal{ND}, \ \mathfrak{a} \in \mathcal{A}:$

$conn\,(a, x, y) \equiv (from\,(\mathfrak{a}) = x \ \wedge \ to\,(\mathfrak{a}) = y) \ \vee \ (from\,(\mathfrak{a}) = y \ \wedge \ to\,(\mathfrak{a}) = x),$

$\forall x, y \in \mathcal{ND}, \ r \in \mathcal{R}: \ link\,(x, y, r) \equiv \ \exists \mathfrak{a} \in \mathcal{A} : \mathfrak{a} \in inst\,(r) \ \wedge \ conn\,(\mathfrak{a}, x, y)\,,$

$\forall x, y \in \mathcal{ND}, \ r \in \mathcal{R}: !link\,(x, y, r) \equiv \ \exists! \mathfrak{a} \in \mathcal{A} : \mathfrak{a} \in inst\,(r) \ \wedge \ conn\,(\mathfrak{a}, x, y)\,.$

Appendix B: Proofs of Theorems

Proof of Theorem 5.1

Assume, to the contrary, that $\exists x \in \mathcal{ND} \ \ \exists i \in \mathbb{N}: \ nd\left(S^i(x)\right) = x.$

Then we have $S^{i+1}(x) = S\left(nd\left(S^i(x)\right)\right) = S\,(x).$

Similarly, we prove that $S^{i+2}(x) = S^2(x)$ and so on.

That is, $\forall j \in \mathbb{N}, \ 1 \leq j \leq i: \ S^{i+j}\,(x) = S^j\,(x).$

By the previous equation it is clear that

$$\forall n, j \in I\!N,\ 1 \le j \le i:\quad S^{n*i+j}(x) = S^{j}(x)$$

This is in contrast with axiom (5.2), which implies that all the nodes in a finite number of recursive steps belongs to information base context.□

Proof of Theorem 5.2

($\Rightarrow$) Assume, to the contrary, that $\exists y \in \mathcal{ND}:\ RelN(x,c) = RelN(y,c) \wedge x \neq y$. Since $RelN(x,c) = RelN(y,c)$, we have

$$len(RelN(x,c)) = len(RelN(y,c)) = k.$$

By the definition 5.12, we know that $RelN(x,c) \in \mathcal{SN}(x)$ and $RelN(y,c) \in \mathcal{SN}(y)$. Then we have

$RelN(x,c) = N_p(nd\left(S^{k-1}(x)\right) \odot_{lc(S^{k-1}(x))} \cdots \odot_{lc(S(x))} x)$
$\quad = L\left(nd\left(S^{k-1}(x)\right)\right) \oplus_{lc(S^{k-1}(x))} \cdots \oplus_{lc(S(x))} L(x)\,.$
$RelN(y,c) = N_p(nd\left(S^{k-1}(y)\right) \odot_{lc(S^{k-1}(y))} \cdots \odot_{lc(S(y))} y)$
$\quad = L\left(nd\left(S^{k-1}(y)\right)\right) \oplus_{lc(S^{k-1}(y))} \cdots \oplus_{lc(S(y))} L(y)\,.$

Since $RelN(x,c) = RelN(y,c)$ the previous equations imply that

$$\forall i \in I\!N, 1 \le i < k: L\left(nd\left(S^{i}(x)\right)\right) = L\left(nd\left(S^{i}(y)\right)\right) \wedge L(x) = L(y) \wedge lc\left(S^{i}(x)\right) = lc\left(S^{i}(y)\right). \quad \text{(B.1)}$$

By the definition 5.12 we have

$$nd\left(S^{k-1}(x)\right) \in NDneighbors(c) \wedge nd\left(S^{k-1}(y)\right) \in NDneighbors(c)\,.$$

This and the equation (B.1) imply that:

$$\left\{\begin{array}{rcl} S^{k}(x) & = & c \\ S^{k}(y) & = & c \\ L\left(nd\left(S^{k-1}(x)\right)\right) & = & L\left(nd\left(S^{k-1}(y)\right)\right) \end{array}\right\} \Rightarrow$$

$$\left\{\begin{array}{rcl} S\left(nd\left(S^{k-1}(x)\right)\right) & = & c \\ S\left(nd\left(S^{k-1}(y)\right)\right) & = & c \\ L\left(nd\left(S^{k-1}(x)\right)\right) & = & L\left(nd\left(S^{k-1}(y)\right)\right) \end{array}\right\} \overset{(ax.5.3)}{\Rightarrow}$$

$$nd\left(S^{k-1}(x)\right) = nd\left(S^{k-1}(y)\right) \overset{(B.1)\&(ax.5.4)}{\Rightarrow} S^{k-1}(x) = S^{k-1}(y)\,.$$

We can go on similarly to prove that

$$\forall i \in I\!N,\ 1 \le i \le k-1:\ S^{i}(x) = S^{i}(y)\,.$$

The previous equation is true for $i = 1$, that is, $S(x) = S(y)$. This, the equation (B.1) and the axiom (5.3) imply that $x = y$. Hence our original assumption that $x \neq y$ must be false.

($\Leftarrow$) Assume, to the contrary, that $\exists y \in \mathcal{ND}:\ x = y \wedge RelN(x,c) \neq RelN(y,c)$. The fact that the S is a function combining with the axiom (5.4) implies that

$x = y \Rightarrow S(x) = S(y) \Rightarrow \cdots \Rightarrow S^{k}(x) = S^{k}(y)$
$\Rightarrow nd\left(S^{k-1}(x)\right)\ \odot_{lc(S^{k-1}(x))}\ \cdots\ \odot_{lc(S(x))}\ x$
$\quad = nd\left(S^{k-1}(y)\right)\ \odot_{lc(S^{k-1}(y))}\ \cdots\ \odot_{lc(S(y))}\ y$
$\Rightarrow N_p(nd\left(S^{k-1}(x)\right)\ \odot_{lc(S^{k-1}(x))}\ \cdots\ \odot_{lc(S(x))}\ x)$
$\quad = N_p(nd\left(S^{k-1}(y)\right)\ \odot_{lc(S^{k-1}(y))}\ \cdots\ \odot_{lc(S(y))}\ y)$
$\Rightarrow RelN(x,c) = RelN(y,c).$

Hence, our original assumption that $RelN(x,c) \neq RelN(y,c)$ must be false.□

Appendix C: Sets, Functions, Expressions

Tables C.1, C.2, C.3 collects all sets, functions and expressions of this paper.

Table C.1. Table of sets.

Set	Description
$\mathcal{ND}$	nodes
$\mathcal{A}$	links
$\mathcal{O}$	objects ($\mathcal{O} = \mathcal{ND} \cup \mathcal{A}$)
$\mathcal{R}$	link classes ($\mathcal{R} \subseteq \mathcal{A}$)
$\mathcal{L}$	atomic logical names
$\mathcal{P}$	paths
$\mathcal{P}(x)$	paths reaching the node x

Set	Description
$\mathcal{N}_A$	link names
$\mathcal{N}_P$	path names
Cxt	contexts
$\mathcal{SP}(x)$	scope paths ending at the node x
$\mathcal{SN}(x)$	names of scope paths ending at the node x

Table C.2. Table of functions.

Function	Description
$from\,(\mathfrak{a})$	gives the object which the link $\mathfrak{a}$ emanates from
$to\,(\mathfrak{a})$	gives the object which the link $\mathfrak{a}$ points to
$inst\,(o)$	gives the set of instances of the object class o
$L\,(o)$	gives the atomic name of the object o
$Root(p)$	gives the first node of the path p
$Leaf(p)$	gives the last node of the path p
$len(p)$	gives the length of the path p
$N_a(a)$	link name function, Definition 4.3 (gives the name of the link a)
$N_p(p)$	path name function, Definition 4.5 (gives the name of the path p)
$nd\,(c)$	gives the pivot node of the context c
$lc\,(c)$	gives the pivot link class of the context c
$Cnts(c)$	gives the set of contents of the context c
$brCnts(c)$	gives the broad contents of the context c
$strCnts(c)$	gives the strict contents of the context c
S	scope function, Definition 5.8

Table C.3. Table of expressions.

Expression	Description
$conn\,(a, x, y)$	link a connects the nodes x and y
$link\,(x, y, r)$	there exists at least one link between the nodes x and y which is instance of link class r
$RelN(x, c)$	relative name of the node x w.r.t. the context c
$GlobN(x)$	global name of the node x
$CxtN(c)$	name of the context c

References

1. S. Abiteboul and A. Bonner, Objects and Views, *Proc. ACM-SIGMOD Conf.*, February 1991, 238–247.
2. A. Albano, R. Bergamini, G. Ghelli, and R. Orsini, An Object Data Model with Roles, *Proc. VLDB'93*, Dublin, Ireland, 1993, 39–51.
3. F. Bancilhon and N. Spyratos, Update Semantics of Relational Views, *ACM Trans. on Database Systems*, **6** (1981) 557–575.
4. C. Batini and M. Lenserini, A Methodology for Data Schema Integration in the Entity Relationship Model, *IEEE Trans. on Software Engineering*, **SE-10** (1984) 650–663.
5. P. Bernstein and U. Dayal, An Overview of Repository Technology, *Proc. 20th VLDB Conf.*, Santiago, Chile, 1994, 705–713.
6. H. Bhargava, S. Kimbrough, and R. Krishnan, Unique Names Violations, a Problem for Model Integration or You Say Tomato, I Say Tomahto, *ORSA Journal on Computing*, **3** (1991) 107–120.
7. P. Constantopoulos, Cultural Documentation: The CLIO System, Technical Report 115, Institute of Computer Science, Foundation for Research and Technology Hellas, January 1994.
8. P. Constantopoulos and M. Dörr, The Semantic Index System: A brief presentation, Institute of Computer Science, Foundation for Research and Technology - Hellas, 1994, (http://www.ics.forth.gr/proj/isst/Systems/SIS/).
9. P. Constantopoulos and M. Dörr, Component Classification in the Software Information Base, in *Object-Oriented Software Composition*, eds. O. Nierstrasz and D. Tsichritzis (Prentice-Hall, 1995) 177–200.
10. P. Constantopoulos, M. Theodorakis, and Y. Tzitzikas, Developing Hypermedia Over an Information Repository, Proc. 2nd Workshop on Open Hypermedia Systems, Hypertext'96, Washington,DC, USA, March, 1996.
11. P. Constantopoulos and Y. Tzitzikas, Context-Driven Information Base Update, *Proc. CAiSE'96*, Heraklion, Crete, Greece, May 1996, 319–344 (Published in Lecture Notes in Computer Science 1080, Springer).
12. G. Copeland and S. Khoshafian, Object Identity, *Proc. OOPSLA*. ACM, September 1986, 406–416.
13. B. Czejdo and D. Embley, View Specification and Manipulation for a Semantic Data Model, *Information Systems*, **16** (1991) 585–612.
14. N. Delisle and M. Schwartz, Contexts: A Partitioning Concept for Hypertext, *ACM Trans. on Office Information Systems*, **5** (1987) 168–186.
15. N. Findler, *Associative Networks*, New York: Academic Press, 1979.
16. G. Gottlob, P. Paolini, and R. Zicari, Properties and Update Semantics of Consistent Views, *ACM Trans. on Database Systems*, **13** (1988) 486–524.
17. G. Gottlob, M. Schrefl, and B. Röck, Extending Object-Oriented Systems with Roles, *ACM Trans. on Information Systems*, **14** (1996) 268–296.
18. R. Guha, Contexts: A Formalization and Some Applications, Technical Report ACT-CYC-423-91, MCC, 1991.
19. G. Hendrix, Encoding Knowledge in partitioned networks, 1979, In [15].
20. R. Hull and R. King, Semantic Database Modeling, *ACM Computing Surveys*, **19** (1987) 202–260.
21. ISO-2788, Documentation — Guidelines for the Development of Monolingual Thesauri, 1986.
22. V. Kashyap and A. Sheth, Semantic and Schematic Similarities between Database Objects: A Context-based approach, *VLDB Journal*, **5** (1996) 276–304.
23. B. Katzenberg and P. Piela, Work Language Analysis and the Naming Problem, *Com-*

munications of the ACM, **36** (1993) 86–95.
24. M. Kifer, W. Kim, and Y. Sagiv, Querying Object-Oriented Databases, in *Proc. ACM-SIGMOD Conf.*, ed. M. Stonebraker (San Diego, California, June 1992) 393–402.
25. M. Koubarakis, J. Mylopoulos, M. Stanley, and A. Borgida, Telos: Features and Formalization, Technical Report 18, Institute of Computer Science, Foundation for Research and Technology - Hellas, 1989.
26. R. Langacker, *Foundations of Cognitive Grammars*, Stanford U. Press, 1987.
27. J. McCarthy, Notes on Formalizing Context, *Proc. IJCAI-93*, Chambery, France, 1993, 555–560.
28. R. Motschnig-Pitrik, An Integrated View on the Viewing Abstraction: Contexts and Perspectives in Software Develepment, AI, and Databases, *J. of Systems Integration*, **5** (1995) 23–60.
29. R. Motschnig-Pitrik, Requirements And Comparison of View Mechanisms for Object-Oriented Databases, *Information Systems*, **21** (1996) 229–252.
30. J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis, Telos : Representing Knowledge about Information Systems, *ACM Trans. on Information Systems*, **8** (1990) .
31. J. Mylopoulos and R. Motschnig-Pitrik, Patritioning Information Bases with Contexts, *Proc. 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95)*, Vienna, Austria, 1995, 44–55.
32. J. Mylopoulos and R. Motschnig-Pitrik, Semantics, Features, and Applications of the Viewpoint Abstraction, *Proc. CAiSE'96*, Heraklion, Crete, Greece, May 1996, 514–539 (Published in Lecture Notes in Computer Science 1080, Springer).
33. A. Ouksel and C. Naiman, Coordinating Context Building in Heterogeneous Information Systems, *J. of Intelligent Inf. Systems*, **3** (1994) 151–183.
34. J. Peckham and F. Maryanski, Semantic Data Models, *ACM Computing Surveys*, **20** (1988) 153–189.
35. V. Prevelakis and D. Tsichritzis, Perspectives on Software Development Environments, *Proc. CAiSE'93*, Paris, France, June 1993, 586–600.
36. J. Pustejovsky and B. Pustejovsky, Lexical Knowledge Representation and Natural Language Processing, *Artificial Intelligence*, **63** (1993) 193–223.
37. J. Richardson and P. Schwarz, Aspects: Extending Objects to Support Multiple, Independent Roles, *Proc. ACM-SIGMOD Conf.*, Denver, Colorado, May 1991, 298–307.
38. E. Sciore, Object Specialization, *ACM Trans. on Information Systems*, **7** (1989) 103–122.
39. E. Sciore, M. Siegel, and A. Rosenthal, Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems, *ACM Trans. on Database Systems*, **19** (1994) 254–290.
40. J. Shilling and P. Sweeney, Three Steps to Views: Extending the Object-Oriented Paradigm, *Proc. OOPSLA*, October 1989, 353–361.
41. Y. Shoham, Varieties of Context, in *AI and Mathematical Theory of Computation — Papers in Honor of John McCarthy*, ed. V. Lifschitz (Academic Press, 1991).
42. E. Svenonius, Design of Controled Vocabularies, *Encyclopedia of Library ond Information Science* (Marcel Dekker, 1989) 83–109.
43. M. Theodorakis, Name Scope in Semantic Data Models, Master's thesis, Dept. of Computer Science, University of Crete, Greece, October 1995, Also, Technical Report, no. 141, Institute of Computer Science, Foundation for Research and Technology Hellas.
44. M. Theodorakis, Context-based Naming in Semantic Networks, *Proc. 3rd Doctoral Consortium on CAiSE'96*, May 1996, 53–56.
45. Y. Tzitzikas, View updates in knowledge bases, Master's thesis, Dept. of Computer Science, University of Crete, Greece, October 1995, Also, Technical Report, no. 142, Institute of Computer Science, Foundation for Research and Technology Hellas.