Contextualization as an Independent Abstraction Mechanism for Conceptual Modeling

Anastasia Analyti[1], Manos Theodorakis[2,*], Nicolas Spyratos[3], Panos Constantopoulos[1,4]

[1] *Institute of Computer Science, FORTH-ICS, Crete, Greece*

[2] *KOMVOS, Avenue Adolphe Lacomble 29, 1030 Brussels, Belgium*

[3] *Laboratoire de Recherche en Informatique, Universite de Paris-Sud, France*

[4] *Department of Informatics, Athens University of Economics and Business, Greece*

E-mail: *analyti@ics.forth.gr, manos@theodorakis.gr, spyratos@lri.fr, panosc@aueb.gr*

Abstract

The notion of context appears in computer science, as well as in several other disciplines, in various forms. In this paper, we present a general framework for representing the notion of context in information modeling. First, we define a context as a set of objects, within which each object has a set of names and possibly a reference: the reference of the object is another context which “hides” detailed information about the object. Then, we introduce the possibility of structuring the contents of a context through the traditional abstraction mechanisms, i.e. classification, generalization, and attribution. We show that, depending on the application, our notion of context can be used as an independent abstraction mechanism, either in an alternative or a complementary capacity with respect to the traditional abstraction mechanisms. We also study the interactions between contextualization and the traditional abstraction mechanisms, as well as the constraints that govern such interactions. Finally, we present a theory for contextualized information bases. The theory includes a set of validity constraints, a model theory, as well as a set of sound and complete inference rules. We show that our core theory can be easily extended to support embedding of particular information models in our contextualization framework.

Keywords: conceptual modeling, contextualization, viewpoints, abstraction mechanisms, model theory, inference rules.

1 Introduction

The notion of context is of fundamental importance in cognitive psychology, linguistics, and computer science. In computer science, a number of formal or informal definitions of some notion of context have appeared in several areas, such as artificial intelligence, software development, databases, data integration, machine learning, and

[*] Work was done during the doctoral studies of the author at FORTH-ICS.

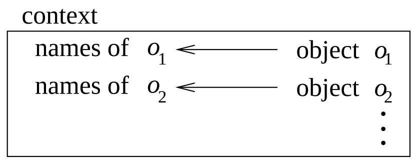
knowledge representation. All these notions of context are very diverse and serve different purposes, yet they all share the general feature of serving as a frame of reference for relativizing representations of reality.

In software development and databases, the notion of context appears in the form of views [3, 23, 49, 1, 52, 41, 9], aspects [46], roles [24, 50], or even workspaces which are used to support cooperative work [30]. In machine learning, context is treated as environmental information for concept classification [39, 37]. In the area of data integration, contexts are used to exchange and adapt value from local information sources to the global application domain [29, 45, 28]. In artificial intelligence, the notion of context appears as a means of partitioning knowledge into manageable sets, or as a logical construct that facilitates reasoning activities [53, 25, 38, 22, 4, 21, 6, 51]. Finally, in the area of knowledge representation, the notion of context appears as an abstraction mechanism for partitioning an information base into possibly overlapping parts [43, 44, 61, 60], or for dividing the global schema of a database into clusters in order to deal with schema complexity [70, 18, 58, 13, 8, 65].

Lately, there is growing interest in the application of context to the Semantic Web, pervasive computing, and context-aware services. In the Semantic Web, the notion of context is used to encapsulate different views and applications of objects [14, 5, 7], as well as user preferences and query context for effective web search [36]. In pervasive computing and context-aware services, explicit representation of context and contextual knowledge is considered critical to intelligent agents. In this framework, a context can be a distinguished collection of possible world features that has predictive worth to the agents. Once an agent knows that it is in a particular context, it immediately knows a great deal about the situation [63, 64]. In addition, a context can be a description of a situation (location, environmental attributes etc.) evaluated by an agent, or available to a service before and during execution [10, 11, 57, 56, 55].

Our objective in this paper is to establish a formal notion of context in conceptual modeling, that supports the development and effective use of large information bases in various applications. A *context* in an information base can be seen as a higher-order conceptual entity that groups together other conceptual entities on which we want to focus.

In [60], a context is defined as a set of objects within which each object is associated with a set of names as in the following diagram:



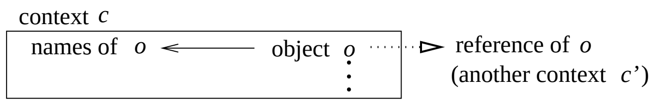
In a research institute comprising several groups, the set of all newly hired researchers might be seen as a context, the $juniors$ context. In that context, the objects are the junior researchers (independently, of the research group to which they belong) and each object is associated with a set of names: i.e., a social name (e.g. “John”) together with one or more nicknames (e.g., “the hacker”) that are used only among junior researchers.

A prominent feature of the approach in [60] is that each context is considered itself to be an object, i.e., there is no distinction between objects and contexts, thus allowing nesting and sharing of contexts.

In the present approach, we *distinguish* between objects and contexts, allowing both nesting and sharing of contexts to be handled in a more flexible way. More specifically, with respect to [60], the notion of context is enhanced in two ways:

1. We introduce references from objects to contexts.

   The contents of a context is still defined to be a set of objects, each of which is associated with a set of names (as before), with the following *additional* feature: We allow each object to be associated with another context that we call its *reference*. Thus, each object of a context is now associated with a set of names, and (possibly) with a reference, as in the following diagram:

   In this framework, each real-world object is represented in the information base by an object. Collections of real world objects that are of interest are represented by contexts. The reference of an object o in a context c (if defined) is another context c' that contains further information about o, as seen from context c. We consider that this information is critical for understanding the semantics of a context. Different references of the same object (from different contexts) signify different partial representations or points of view of the same object.

   As we shall see, this notion of context can be used as an independent means for modeling reality. Thus, it can serve as the foundation of an abstraction mechanism for information modeling, that we shall call *contextualization*.

2. We allow the set of objects of a context to be structured through the traditional abstraction mechanisms of classification, generalization, and attribution[1]. We study how these three abstraction mechanisms interact with contextualization, in particular how instance-of, ISA, and attribute links between objects affect the definition of their references.

The extended notion of context introduced here enriches the modeling capabilities of the traditional abstraction mechanisms in two significant ways:

1. *Expressive power*: By supporting relative semantics, i.e., relative naming and relative relationships, and by interacting with the traditional abstraction mechanisms, contextualization provides new modeling capabilities.

2. *Modularity*: By retaining the essential information and hiding inessential details (encapsulated in the form of references), context helps to increase comprehensibility and communicability in complex applications such as information retrieval over the web, cooperative work in distributed environments, large engineering databases, scientific catalogs, etc.

[1]By ”attribution” we mean the assignment of an intrinsic property to an object as well as the declaration of its (binary) relationships to other objects. The abstraction mechanism of *aggregation* is a limited form of attribution [26].

In this paper, we present a formal theory of context and the ways contexts interact with each other and with the traditional abstraction mechanisms of conceptual modeling. Our theory includes a set of validity constraints, a model theory, as well as a set of sound and complete inference rules. We also demonstrate that our core theory can be extended to support embedding of particular information models in our contextualization framework.

The remainder of the paper is organized as follows: In Section 2, we define the notion of context *without* structuring of its objects, and we discuss some of the modeling capabilities of contextualization. In Section 3, we *add* structure to the set of objects of a context through the traditional abstraction mechanisms, i.e., classification, generalization, and attribution. Additionally, we study the interaction between contextualization and the traditional abstraction mechanisms. In Section 4, we present a formal theory of contextualized information bases, including a set of validity constraints, a model theory, as well as a set of sound and complete inference rules. In Section 5, we show that our core theory can be combined with a set of particular information model validity constraints/inference rules to support embedding of particular information models into our contextualization framework. In Section 6, we compare our framework with those of related works. Finally, in Section 7, we make some concluding remarks and suggestions for further research. The proofs of all propositions are given in the Appendix.

2 The notion of context

In this section, we give the definition of a context as used in this paper. First, we motivate this definition through an example. Suppose that we want to talk about Greek islands by simply using their names without further description. Let us consider the island of Crete. We can represent this island by an *object identifier*, say o_1, and by associating this identifier with the name `Crete`. We write $names(o_1) = \{\texttt{Crete}\}$ and we denote this as follows[2]:

$$\texttt{Crete} : o_1$$

Next, let us consider the island of Santorini. Following a similar approach, we represent this island by an object identifier o_2 and by associating it with the name `Santorini`. However, the island of Santorini is also known under the name Thera. So this time, we associate o_2 with the set of names $\{\texttt{Santorini}, \texttt{Thera}\}$, i.e., this time we write $names(o_2) = \{\texttt{Santorini}, \texttt{Thera}\}$ and we denote this as follows:

$$\texttt{Santorini}, \texttt{Thera} : o_2$$

Finally, let us consider one of those tiny, uninhabited islands of Greece that happen to be nameless. We represent such an island by an object identifier o_3 and by associating it with no name, i.e., we write $names(o_3) = \{\}$ and we denote this as follows:

$$: o_3$$

Continuing in the same way, we can represent every Greek island in a similar manner. The set of all such representations is what we call a *context* and we represent it by a *context identifier*, say c_1, as shown in Figure 1.

Suppose next we want to talk about the Greek mainland by simply using the names of each region of Greece without further description. Proceeding in a similar way as in the case of Greek islands, we can create a second

[2]In this paper, the terms *object* and *object identifier* will be used interchangeably.

context, say c_2, as shown in Figure 1.

Suppose now that we want to talk about the geography of Greece seen as a division of Greece into islands and mainland. First, let us consider the islands. We can represent the islands by an object identifier, say o, and by associating it with the name `Islands`, i.e., $names(o) = \{\texttt{Islands}\}$. However, the object o is a higher level object that collectively represents all Greek islands, i.e., the object o collectively represents the contents of context c_1. In other words, if we want to see what o means at a finer level of detail, then we have to "look into" the contents of c_1. Thus we call context c_1 the *reference* of object o, and we write $ref(o) = c_1$. Summarizing our discussion on islands, we write $names(o) = \{\texttt{Islands}\}$ and $ref(o) = c_1$, and we denote this as follows:

$$\text{Islands : o} \cdots\!\rhd\ c_1$$

Following a similar reasoning, we can represent the mainland by an object identifier, say o', and by associating it with the name `Mainland` and the reference c_2. We can now group together the islands and the mainland to form a context c_3, as shown in Figure 1. Then, the geography of Greece can be represented by an object identifier o'' whose reference is context c_3.

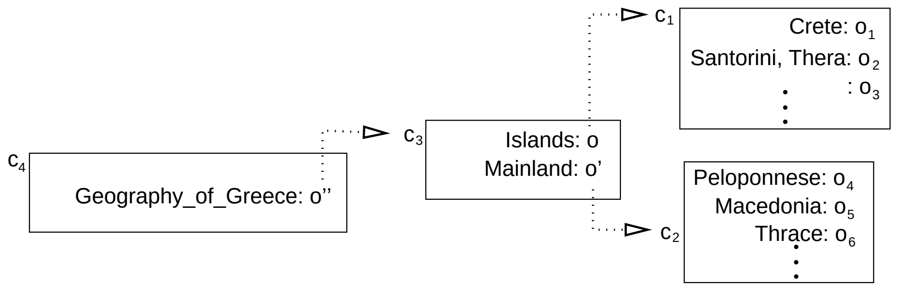


Figure 1: An example of context structure: Geography of Greece

The previous examples suggest the following definition of context.

Definition 2.1 Context. A context consists of a context identifier c and a set of object identifiers, denoted by $objs(c)$, such that each $o \in objs(c)$ is associated with:

1. a set of names, called *the names of o in c*, and denoted by $names(o, c)$, and
2. at most one context identifier, called *the reference of o in c*, and denoted by $ref(o, c)$.

If $o \in objs(c)$ is not associated with a context identifier then $ref(o, c)$ is considered to be undefined. $\diamond$

The reason why we use the symbols $names(o, c)$ and $ref(o, c)$, instead of $names(o)$ and $ref(o)$ used in the previous examples, is that an object can belong to different contexts and may have different names and/or a different reference in each context. That is, *names and references are context-dependent*.

Of course, starting from a context c, we should not be able to reach c again, by navigating through reference links (denoted in the figure by dotted arrows). To express this formally, we call $successor$ of a context c any context c' referenced by an object of c. Additionally, we call $descendant$ of a context c_1 any context c_n such that there is a sequence of contexts $c_1, c_2, ..., c_n$, where c_i is a successor of c_{i-1}, for $i = 2, ..., n$.

Definition 2.2 Scope of a context. Let c be a context. Consider the set of contexts consisting of c together with all descendants of c. For any two contexts c_1 and c_2 in this set, create an edge from c_1 to c_2 iff c_2 is a successor of c_1. We call the created graph, $scope$ of c, and denote it by $scope(c)$. $\diamond$

Clearly, the scope of a context c can be infinite and/or cyclic. However, for the purposes of this paper, we shall make the following basic assumption: *The scope of any context c is a finite, directed acyclic graph.* Obviously, the scope of c has c as its only root, and every leaf of the scope is a context whose objects have no references. For example, in Figure 1, $scope(c_4)$ has four nodes, contexts c_1, c_2, c_3, c_4, and contexts c_1, c_2 are the leaves of $scope(c_4)$.

In our previous examples, while explaining the construction of a context, we followed a bottom-up approach. That is, we started from simple objects and built up contexts which were later on referenced by higher level objects ("moving" from right to left in Figure 1). Clearly, we could have followed the opposite construction, i.e., a top-down approach ("moving" from left to right in Figure 1). Or, we could follow a mixed approach.

This flexibility is important in conceptual modeling and gives (among other things) the possibility of *modular design*, i.e., retaining at each level of abstraction the essential information and hiding inessential details (by "encapsulating" them in the form of a reference).

Let us see a top-down definition of a context. Suppose we are defining a context c_0 containing several types of guides to Greece. Let us model first a tourist guide as follows:

$$\text{Tourist\_Guide: } o_1 \cdots\!\triangleright\; c_1$$

The next stage is to define the context c_1 that contains the information concerning the tourist guide. The context c_1 is shown in Figure 2.

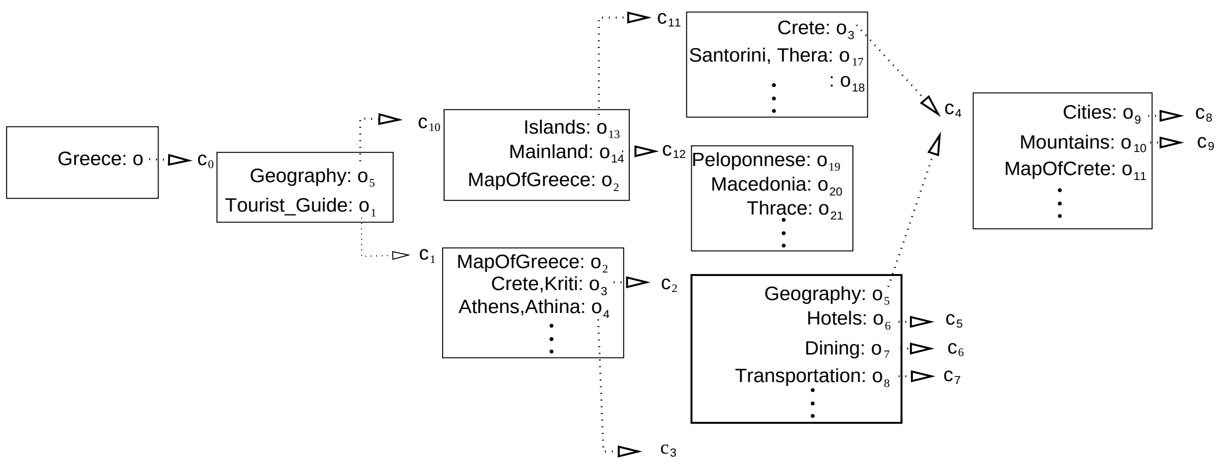


Figure 2: An example of context structure: tourist guide and geography of Greece

We then define the contexts c_2, c_3 to which the objects of c_1 refer. Context c_2 contains tourist information concerning Crete, such as geography, hotels, dining, transportation, etc, while context c_3 is not shown in the figure. Subsequently, we have to define the contexts c_4, c_5, c_6, and c_7 to which the objects of c_2 refer. Context c_4 contains geographical information about Crete, while contexts c_5, c_6, c_7 are not shown in the figure.

Note that object o_3 is shared by both contexts c_1 and c_{11}, but the references of o_3 in the two contexts are different. Additionally, the set of names of o_3 in c_1 are $\{Crete, Kriti\}$, whereas in c_{11} is just $\{Crete\}$. Further, note the sharing of objects o_2 and o_5. Object o_5 is shared by both contexts c_0 and c_2, though in context c_0, object o_5 refers to the geography of Greece, whereas in context c_2, object o_5 refers to the geography of Crete. Disambiguation of meaning is achieved through the contexts c_0 and c_2. Finally note that context c_4 is a reference to both objects o_3 in context c_{11}, and object o_5 in context c_2, as both objects refer to geography of Crete.

Obviously, the notion of context supports a simple and straightforward way of referencing objects at any level of detail. Consider, for example, the tourist guide of Greece in Figure 2. Suppose that, currently, we are in context c_0, and we want to look at Cretan hotels. To do so we can ”go” from object o_1 (`Tourist_Guide`) to object o_3 (`Crete`, `Kriti`), and then to object o_6 (`Hotels`). We indicate this as follows: $o_1.o_3.o_6$, i.e., by forming a path of object identifiers. The last object in the path has a reference to a context that contains the information of interest.

The previous examples demonstrate the features of context, supported directly by our context definition, namely:

1. **Object sharing or overlapping contexts.**
   An object can belong to one or more different contexts. When contexts share objects, we say that contexts overlap. This feature is useful when we want to view an object under different perspectives.

2. **Context-dependent object names.**
   The same object can have different names in different contexts. This is very convenient, because a name may be clearly understood in one context, while not understood in a different context.

3. **Context-dependent references.**
   The same object can have different references within different contexts. In other words, references are context-dependent, representing context-dependent views of an object.

4. **Context sharing.**
   Two different objects, whether they belong to the same context or not, can have the same reference. This is useful for representing the same point of view from two different starting points.

5. **Context-dependent reachability.**
   From within a given context, we can “reach” any object that belongs to the reference of an object within that context (and, recursively, any object that lies on a path).

6. **Synonyms, Homonyms, Anonyms.**
   The same object can have different names in the same context (synonyms). Two different objects can have the same name within a context (homonyms). An object may have no name within a context (anonyms).

Contextualization constitutes an abstraction mechanism in the sense that a context c “encapsulates” its contents and thus any object referencing c can be seen as the abstraction of the contents of c. For example, in Figure 2, the object o_3 (`Crete`, `Kriti`) in context c_1 can be seen as the abstraction of the contents of c_2.

Contextualization can be seen as either an alternative or a complementary abstraction mechanism to traditional abstraction mechanisms (depending on the application). For example, referring to Figure 2, the object o_{13} (`Islands`) in context c_{10} can also be modelled as a class, and the objects o_3, o_{17}, o_{18}, can be modelled as instances of o_{13}. However, it is less obvious how to model the object o_1 (`Tourist_Guide`) by means of the traditional abstraction mechanisms. Indeed, the objects o_2, o_3, o_4 can hardly be considered as instances of o_1. Thus, the use of contexts for describing the tourist guide of Greece seems to be more appropriate.

Roughly speaking, the modeling power of contexts stems from the fact that one can group together quite dissimilar things, regardless of any structural relationships they may have. In fact, no such relationships are required to hold the contents of a context together.

As we shall see shortly, the combination of contextualization with the traditional abstraction mechanisms provides even further modeling capabilities.

3 Structuring the contents of a context

In this section, we informally[3] structure the objects of a context through attribution, classification, and generalization. To achieve this we enhance our notion of context, as follows:

- each object of a context is either a simple object or a link object (attribute, instance-of, or ISA);
- each object can be related to other objects through attribute, instance-of, or ISA links;

In order to specify the objects (source and destination) that a link relates, each context is assumed to be equipped with:

- A predicate for defining the objects that are attribute links. This is the predicate

  $attr(att\_obj, from, to)$

  declaring that object att_obj is an attribute link with source object *from* and destination object to.
- A predicate for defining the objects that are instance-of links. This is the predicate

  $in(in\_obj, from, to)$

  declaring that the object in_obj is an instance-of link, and the object *from* is an instance of the class to.
- A predicate for defining the objects that are ISA links. This is the predicate

  $isa(isa\_obj, from, to)$

  declaring that the object isa_obj is an ISA link, and the class *from* is a subclass of class to.

Note that, as attribute, instance-of, and ISA links are objects, a link can have zero, one, or more names.

Consider, for example, modeling employees using a class whose instances have three attributes: name, salary and address. Using our definition of context, this modeling can be done as shown in Figure 3(a), where o is the employee class, and the three attribute declarations define the objects o_1, o_2 and o_6 as attributes from class o to classes o_4, o_5 and o_3, respectively. Figure 3(b) shows a more convenient representation of context c, where the attributes o_1, o_2 and o_6 are represented by arrows. For example, $attr(o_1, o, o_4)$ is represented by the arrow from

[3]The formal definition as given in Section 4.

o to o_4. Note that attribute o_6 has the same name as object o_3, something allowed by our definition of context. However, if referencing of objects is done through names, this may lead to ambiguities. We address this problem in [60][4].

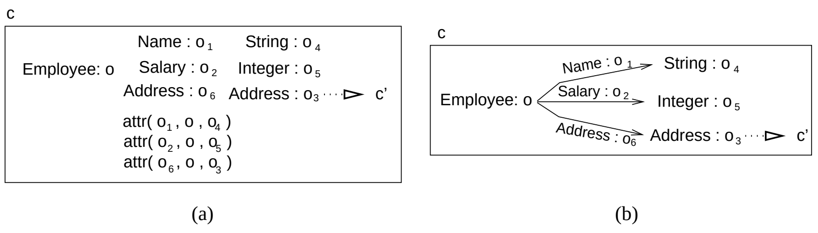


Figure 3: Modeling an employee using attributes

In the rest of the paper, in order to simplify the pictorial presentation of contexts, the object identifiers of instance-of, and ISA links will be omitted from the pictures.

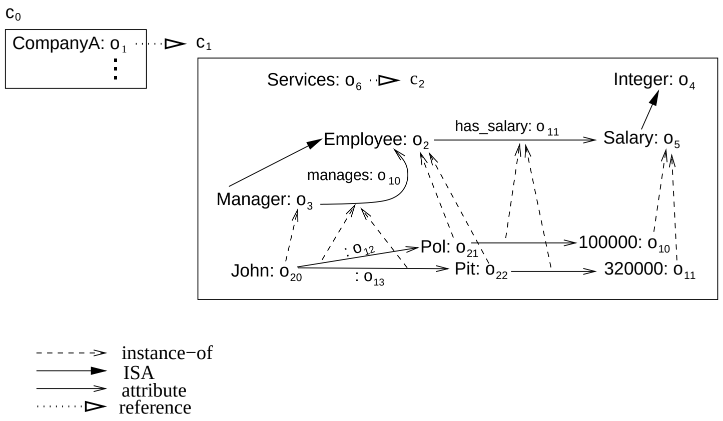


Figure 4: Modeling the employees of a company

Figure 4 shows another example of context with structured contents, this time using all three abstraction mechanisms, i.e., classification, generalization, and attribution. In this example, company `CompanyA` is represented by object o_1. We can find more information about the company in context c_1, which contains information about employees, managers and services provided by the company. In particular, objects `Employee` and `Manager` represent the class of employees and managers, respectively. The salary paid by the company is represented by object o_5 and is an Integer. The fact that employees have a salary is represented by the object o_{11}. Similarly, object o_{10} represents the fact that managers manage employees. Individuals employees are represented by objects `Pol`, `Pit`, and `John`, where `John` manages `Pol` and `Pit`. Thus, `Pol` and `Pit` are instances of `Employee`, whereas `John` is instance of `Manager`. Attributes o_{12} and o_{13} are instances of attribute class `manages` and represent that `John` manages `Pol` and `Pit`, respectively. Object o_6 represents collectively the services available by the company (more information

[4]A simple way to avoid this problem is to put the name of object o_6 in a verb form, e.g. `has_address`, or to use more than one names for object o_6, e.g. $\{$`Address`, `employee_address`$\}$.

is available in context c_2).

Figure 5 shows another example of context with structured contents representing recommendations for dining in a touristic place.

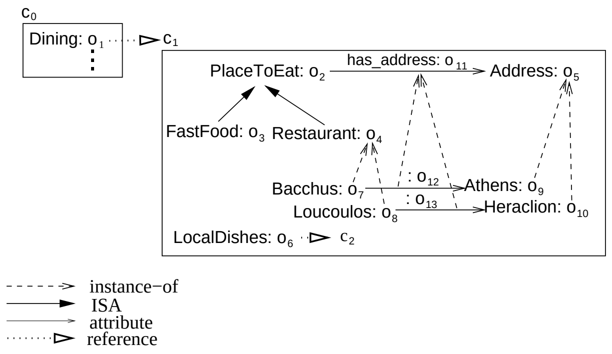


Figure 5: Structured contents of a context

It must be stressed that *all items in a context, including the predicate declarations, are defined relatively to that context.* For example, an object that belongs to two different contexts can have different attributes in each context. Similarly, an object o which is an instance of a class o' in a given context might not be an instance of o' in a different context (assuming that o, o' belong to both contexts), and so on. This is illustrated in Figure 6. Contexts c_2 and c_4 contain information concerning geographic data about Crete during the 15th and 20th century, respectively. The object o_6 represents a place in Crete which is classified under `Village` in the 15th century, whereas the same place is classified under `City` in the 20th century. Note that object o_5 was called Chandax in the 15th century, whereas the same object is called Heraklion in the 20th century. Moreover, the 15th century description includes information on the fortification of the city, while that of the 20th century includes information on the airport.

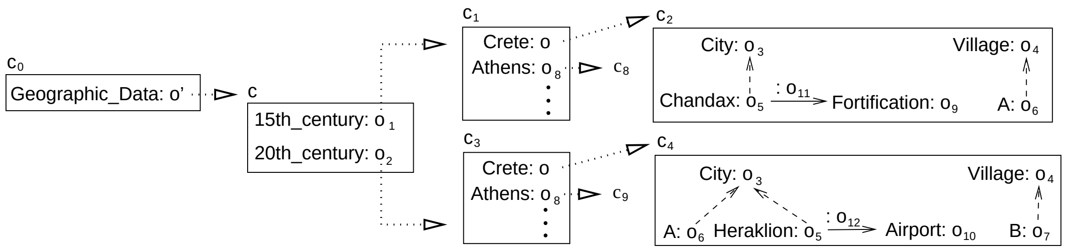


Figure 6: Context-dependent description

3.1 The interaction between abstraction mechanisms

In this section, we study the interaction between the traditional abstraction mechanisms and the mechanism of contextualization. The interaction between the traditional abstraction mechanisms has been extensively studied in the literature (see [26, 54] for a survey and [17] for a review of reasoning in Description Logics). For example, the interaction between classification and generalization is usually expressed through the following constraint: If an object o is instance of a class o', and o' is subclass of o'', then o is instance of o''.

To study the interaction between between the traditional abstraction mechanisms and the mechanism of contextualization, we need the notions of source reference and destination reference of a link object. By *source reference* of a link object, we refer to the reference of the source of a link object. Additionally, by *destination reference* of a link object, we refer to the reference of the destination of the link object.

3.1.1 Attribution and contextualization

In this subsection, we study the interaction between attribution and contextualization.

We start with a constraint imposed to the references of attribute objects. As an attribute cannot exist without a source and a destination, the source reference and the destination reference of the attribute should be included in the reference (if any) of the attribute object. This can be done as shown in Figure 7. Let o be an attribute from object o_1 to object o_2 with references c_1 and c_2, respectively. Then, the reference c of o should contain two special objects o_f and o_t. The object o_f, named `from`, has as reference the context c_1. The object o_t, named `to`, has as reference the context c_2. So from now on, we assume that the reference of every attribute is as shown in Figure 7.

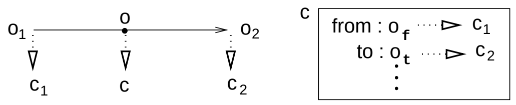


Figure 7: The reference of an attribute

Let us call *traversal path* any path from an object in the source reference of the attribute to an object in the destination reference such that every member of the path is an attribute, instance-of, or ISA link. We call *attribute path* any traversal path every member of which is an attribute. Intuitively, an attribute path defines an (abstract) attribute from an object in the source reference to an object in the destination reference.

Now, the constraint that we propose for the reference of an attribute can be stated informally as follows: the attribute must collectively represent all traversal paths from objects in its source reference to objects in its destination reference. Clearly, in order for this requirement to be satisfied, all traversal paths must be attribute paths. Hence, we have the following constraint on the information that the reference of an attribute can contain:

Constraint 3.1 Attribute Reference Constraint. Every traversal path in the reference of an attribute is an attribute path. $\diamond$

Figure 8 illustrates the interaction between attribution and contextualization in a top-down modeling of demographic data. The reference of attribute o_4 (`Related_To`) is context c_4. This reference contains two traversal paths that are both attribute paths. The first of these paths goes from object o_5 in context c_2 to object o_8 in context c_3, and consists of a single attribute: o_{11} (`born_in`). Within context c_4, this is defined as $attr(o_{11}, o_f.o_5, o_t.o_8)$

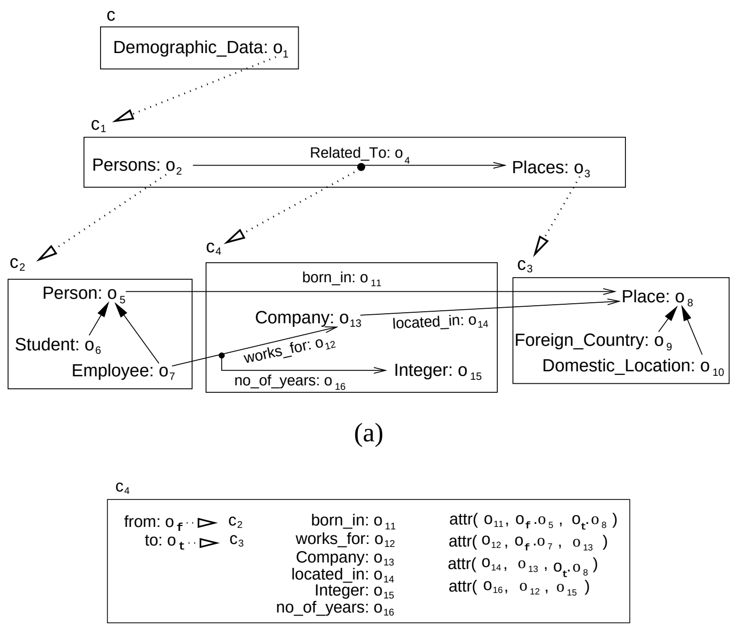


Figure 8: Interaction between attribution and contextualization

because objects o_f and o_t refer to the source and destination references of attribute o_4, respectively[5]. The second path goes from object o_7 in context c_2 to object o_8 in context c_3 and consists of two attributes: o_{12} (`works_for`) from o_7 to o_{13}, and o_{14} (`located_in`) from o_{13} to o_8. Within context c_4, this is defined as $attr(o_{12}, o_f.o_7, o_{13})$ and $attr(o_{14}, o_{13}, o_t.o_8)$, respectively. Note that in Figure 8(a), context c_4 is given in a pictorial way, where the special objects o_f and o_t are omitted and predicates are depicted through arrows. Its actual definition is given in Figure 8(b).

3.1.2 Classification and contextualization

The interaction between classification and contextualization is similar to that between attribution and contextualization. Thus the reference c of an instance-of link o should also contain the special objects o_f and o_t, as shown in Figure 7. In addition, c should contain only instance-of links from objects in the source reference of o to objects in the destination reference of o.

Constraint 3.2 Instance-of Reference Constraint. Every traversal path in the reference of an instance-of link consists of a single instance-of link. $\diamond$

If an object o is an instance of object o' then the reference of the instance-of link may classify objects in the reference of o into object classes in the reference of o'. Intuitively, we can say that the objects in the reference of o follow the ”schema” defined in the reference of o'.

[5]The object paths $o_f.o_5$ and $o_t.o_8$ in c_4 mean that objects o_5 and o_8 can be reached from objects o_f and o_t in c_4 through the references of o_f and o_t in c_4, that is contexts c_2 and c_3, respectively.

Probably the most relevant example of this interaction is the one relating a database schema with its instances. In Figure 9, object o_1 (`Instance_1`) is instance of object o'_1 (`Schema_1`). Note that the reference c_{in} of the instance-of link contains only instance-of links from objects of c_1 to objects of c'_1.

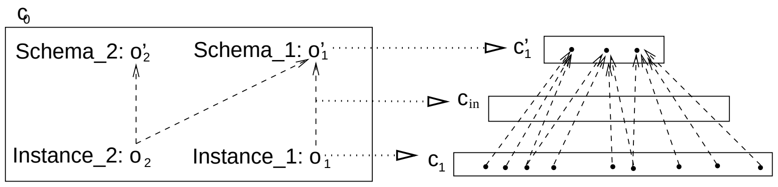


Figure 9: The reference of an instance-of link

Intuitively, within c_{in}, objects of c_1 are instances of objects of c'_1. For example, if c_1 contains a set of relational tuples and c'_1 contains a relational database schema then the instance-of links relate tuples in c_1 with tables in c'_1.

Note that the separation between instance and schema allows for several sets of objects to share the same schema, and the same set of objects to be classified under different schemas. For example, in Figure 9, schemas o'_1 (`Schema_1`) and o'_2 (`Schema_2`) share the same instance set, represented by object o_2 (`Instance_2`). On the other hand, instance sets o_1 and o_2 share the same schema o'_1 (`Schema_1`). A more realistic example for the second case is given in Figure 10, where object o_1 (`Company`) refers to the concept of Company, and objects o_2 (`CompA`) and o_3 (`CompB`) refer to two specific companies. Intuitively, the reference of o_1 corresponds to the schema of a company, and the references of o_2 and o_3 correspond to information about the particular companies. As objects o_2 and o_3 are instances of o_1, objects within the references of o_2 and o_3 are classified into classes in the reference of o_1. This classification takes place within the references c_4 and c_5 of the instance-of links from o_2 and o_3 to o_1, respectively.

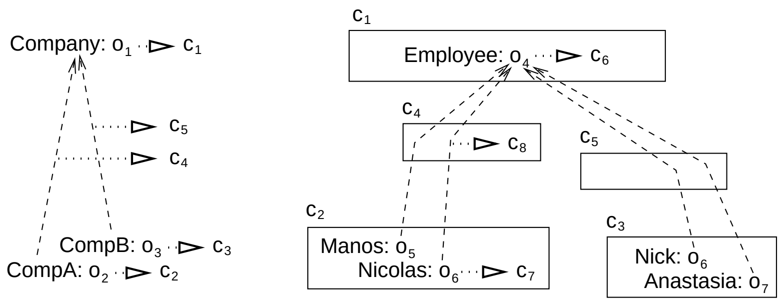


Figure 10: Interaction between classification and contextualization

Continuing with the example of Figure 10, consider the instance-of link from object o_2 (`CompA`) to object o_1 (`Company`). To satisfy the Instance-of Reference Constraint, the reference c_4 of this link may only contain instance-of links from objects in its source reference (context c_2) to objects in its destination reference (context c_1). In other words, within context c_4, objects of company `CompA` may only be classified to the classes in the schema of `Company`. Indeed, within context c_4, there is an instance-of link from object o_6 (`Nicolas`) to object o_4 (`Employee`). The reference c_7 of object o_6 contains information about the employee `Nicolas`, and the reference

c_6 of object o_4 contains schema information about the class `Employee`. To satisfy the Instance-of Reference Constraint, the reference c_8 of the instance-of link may only classify information about the employee `Nicolas` to the schema of `Employee`. Therefore, recursive application of the Instance-of Reference Constraint implies classification of objects according to their entire recursive structure.

3.1.3 Generalization and contextualization

Generalization establishes a subclass-superclass relation between classes and is used to emphasize the similarities among classes with common superclasses and to hide their differences. The interaction between generalization and attribution is expressed by the well known mechanism of *attribute inheritance*. In our framework, in addition to attribute inheritance, we support a new mechanism that we call *reference inheritance*. Roughly speaking, according to reference inheritance, the reference of the subclass *inherits* the contents of the reference of the superclass.

Formally, reference inheritance is defined through a partial order over contexts that we call *context refinement*.

Definition 3.1 Context refinement. We say that context c *refines* context c', or that context c is a $refinement$ of c', denoted by $c \precsim c'$ iff

1. every object of c' is also an object of c,
2. the names of every object o in c' are included in the names of o in c,
3. every instance-of, ISA, and attribute relationship in c' also holds in c, and
4. the reference of every object o of c' is either undefined or refined by the reference of o in c. $\diamond$

Note that the above definition of context refinement is recursive and that every context is a refinement of itself. We show in subsection 4.7 that refinement is a partial pre-ordering, i.e., reflexive and transitive. Moreover, we show that context refinement is a partial ordering up to context equivalence, where context equivalence is defined as follows: two contexts are *equivalent* iff they have (i) the same objects, (ii) the same names for each object, (iii) the same instance-of, ISA, and attribute relationships, and (iv) the references of each object in the two contexts are either both undefined or equivalent. Roughly speaking, two contexts are equivalent if they have the same contents, up to equivalence of the object references.[6].

The following constraint expresses the application of reference inheritance on ISA links.

Constraint 3.3 Reference Inheritance Constraint. The source reference of an ISA link refines the destination reference of the link. $\diamond$

For example, in Figure 11, object o_2 (`Hospital`) is a subclass of object o_1 (`Organization`). The source reference of this ISA link (context c_2) is a refinement of the destination of the link (context c_1), as c_2 contains all the contents of c_1. Therefore, the reference inheritance constraint is satisfied. Intuitively, we can say that the contents of c_1 have been inherited by c_2.

Context refinement can be achieved in stages through the repetitive application of the following operations on the contents of a context: (i) the addition of a new object, (ii) the addition of a new instance-of, ISA, and attribute

[6]Notice the similarity between context equivalence and deep object equality in object oriented databases [2].

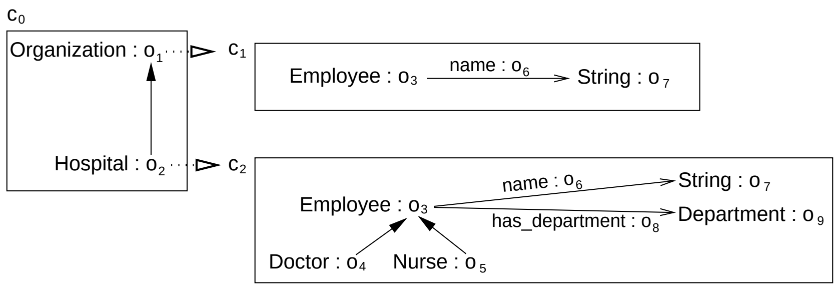


Figure 11: Interaction between generalization and contextualization

relationship, (iii) the addition of a name to an object, (iv) the addition of a reference to an object, and (v) the application of the previous operations to the contents of a reference. The resulting context is certainly a refinement of the original context, as it merely extends the information contained in that context (and no cancellation takes place). In this sense, refinement inheritance is a form of strict inheritance.

We now give a more involved example (see Figure 12). Let c_2 be a context describing medical services. Within c_2, the class o_4 (`Hospital`) and the class o_5 (`PrivateUnit`) are subclasses of the class o_3 (`Health_Care`). In accordance to the Reference Inheritance Constraint, the reference c_4 of `Hospital` and the reference c_5 of `PrivateUnit` are refinements of the reference c_3 of `Health_Care`. Specifically, contexts c_4 and c_5 inherit all the information contained in context c_3, including the object o_8 (`Agents`) that represents the concept of agent. Within context c_3, the reference c_8 of o_8 describes the `Agents` hierarchy in the general health care environment. Within context c_4, the reference c_9 of o_8 describes the `Agents` hierarchy in the hospital environment. Within context c_5, the reference c_{10} of o_8 describes the `Agents` hierarchy in the private unit environment. Note that although contexts c_9 and c_{10} refine context c_8, they describe different hierarchies. For example, c_9 indicates that the director of a hospital should be a doctor, whereas c_{10} indicates that the director of a private unit should be owner of the unit.

To keep the contexts concise, we could eliminate duplications in the contents of the source reference of the ISA link. In this case, the complete contexts are obtained after the application of reference inheritance on the ISA links. A mechanism for eliminating duplications is proposed in [59].

3.1.4 Classification, generalization and contextualization

In this section, we study the interaction between all three abstraction mechanisms, classification, generalization, and contextualization. This interaction is exemplified in Figure 13.

Let c_1 and c_2 be the references of the instance-of links from o to o' and from o to o'', respectively, as shown in Figure 13(a). The question is whether there is any relationship between the contents of c_1 and c_2, as there is an ISA link between o' and o''. Indeed, we show that c_1 refines c_2.

Let c, c' and c'' be the references of the objects o, o' and o'', respectively. As c' refines c'' (see the Reference Inheritance Constraint, Constraint 3.3), it should hold that if an object o_1 in c is classified in a class o_2 in c'' then, o_1 should also be classified in the inherited class o_2 in c'. That is, the instance-of links contained in c_2 should be

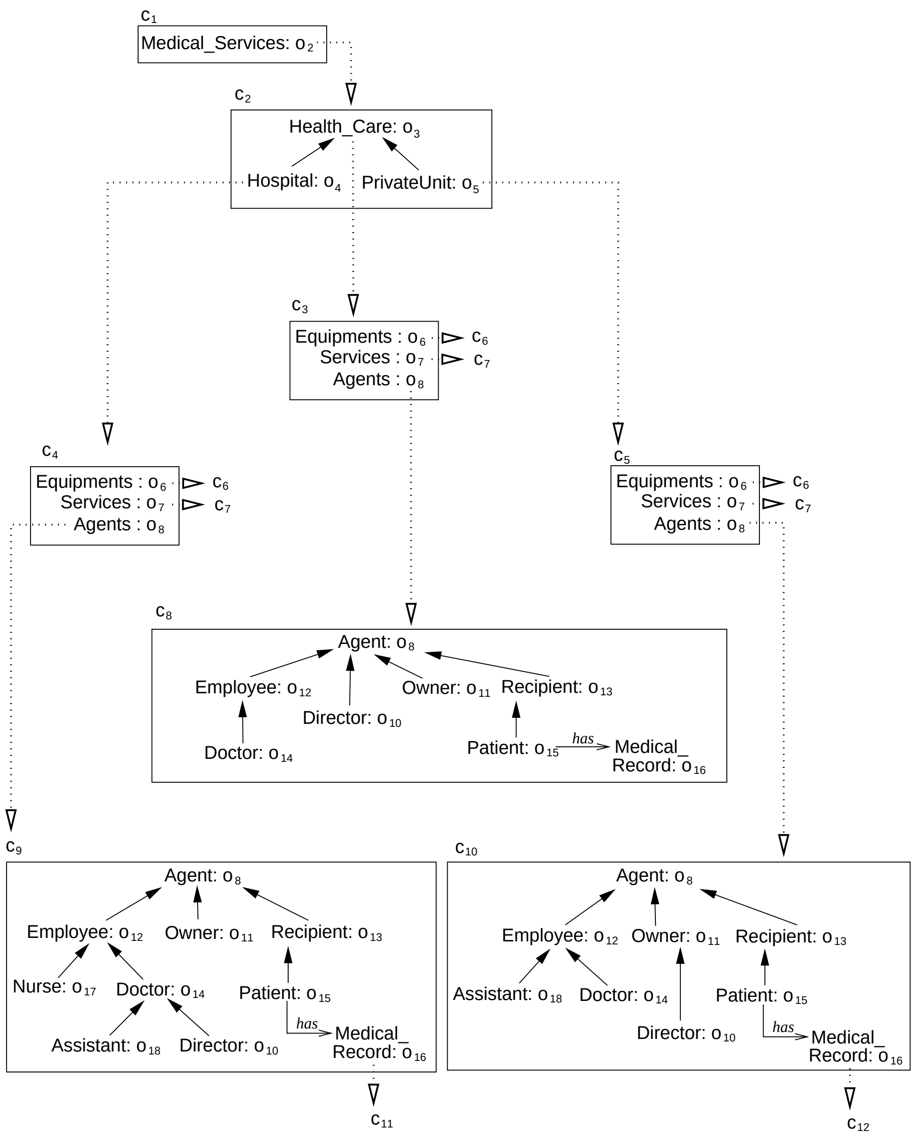


Figure 12: Interaction between generalization and contextualization: An example from a medical environment

inherited by c_1 (see Figure 13(b)), and the reference of an inherited link in c_1 should refine the reference of this link in c_2. This implies that c_1 should refine c_2. We refer to this constraint as *Instance-of Inheritance Constraint*.

Constraint 3.4 Instance-of Inheritance Constraint. The reference of an instance-of link from an object o to a class o' refines the reference of any instance-of link from o to a superclass of o'. $\diamond$

Duplications of instance-of links in context c_1 can be avoided through an inheritance mechanism similar to that applied in the interaction between generalization and contextualization, described in the previous subsection. One such mechanism is proposed in [59].

In the following section, we present a generic, formal theory for contextualized information bases. The theory includes a definition of contextualized information bases, a set of validity constraints, a model theory, as well as a

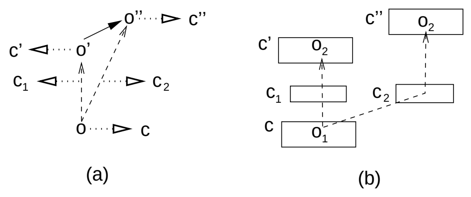


Figure 13: The interaction between classification, generalization, and contextualization

set of sound and complete inference rules.

4 A theory for contextualized information bases

Having introduced our ideas informally, we now formally define our contextualization abstraction mechanism.

We assume a domain consisting of the following three mutually disjoint sets:

- A set of *object identifiers*, or simply *objects*, denoted by $\mathcal{O}$.
- A set of *context identifiers*, or simply *contexts*, denoted by $\mathcal{CXT}$.
- A set of *atomic names*, denoted by $\mathcal{N}$.

4.1 Contextualized information base

In this subsection, we formally define the contents of a context and a contextualized information base.

Let us denote by $\mathcal{RP}$, the set of all possible object paths, that is:

$$\mathcal{RP} = \mathcal{O}^+ = \{o_1.o_2.\ \cdots\ .o_{n-1}.o_n \mid n \in I\!N^+\ \wedge\ \forall 1 \leq i \leq n,\ o_i \in \mathcal{O}\}$$

The contents of a context c are defined as follows:

Definition 4.1 Contents of a context. The *contents* of a context c, denoted by $cnts(c)$, is a tuple:

$$\langle\ objs(c), lex(c), rf(c), in(c), isa(c), attr(c)\ \rangle$$

where

1. $objs(c) \in \mathcal{P}(\mathcal{O})$ is a set of objects.
2. $lex(c) : objs(c) \longrightarrow \mathcal{P}(\mathcal{N})$ is a mapping which associates each object of c with a set of names. We call this function the *lexicon of* c, and we define $names(o, c) = lex(c)(o)$.
3. $rf(c) : objs(c) \longrightarrow \mathcal{CXT}$ is a partial function which associates an object of c with a context. We call this partial function the *reference association of* c, and we define $ref(o, c) = rf(c)(o)$.
4. $in(c) = \{\langle o, p_f, p_t\rangle \mid o \in objs(c)\ \wedge\ p_f, p_t \in \mathcal{RP}\}$ is a set of triplets of the form $\langle o, p_f, p_t\rangle$, expressing that o is an instance-of link in context c from object path p_f to object path p_t.
5. $isa(c) = \{\langle o, p_f, p_t\rangle \mid o \in objs(c)\ \wedge\ p_f, p_t \in \mathcal{RP}\}$ is a set of triplets of the form $\langle o, p_f, p_t\rangle$, expressing that o is an ISA link in context c from object path p_f to object path p_t.

6. $attr(c) = \{\langle o, p_f, p_t\rangle \mid o \in objs(c),\ p_f, p_t \in \mathcal{RP}\}$ is a set of triplets of the form $\langle o, p_f, p_t\rangle$, expressing that o is an attribute link in context c from object path p_f to object path p_t.

In order to formally define a contextualized information base, we need to define the notion of reference path. Accessing information in an information base often involves navigating from one object to another by following links [31]. As the reference of an object within a context is also a context, references provide a means to traverse from an object o of a context c to the objects of another context via the reference of o in c. The sequence of traversed objects constitutes a kind of path, which we call *reference path*.

Definition 4.2 Reference Path.

A *reference path* $o_1.o_2.\ \cdots\ .o_{n-1}.o_n$ in a context c_0 is a sequence of objects where each object o_{i+1} is contained in the reference of the previous object o_i. The reference of o_1 is taken in c_0 (call this reference c_1), the reference of o_2 is taken in c_1, and so on. The set of all reference paths in the context c_0, denoted by $\mathcal{RP}_{c_0}$, is defined as follows:

$$\mathcal{RP}_{c_0} = \{o_1.o_2.\ \cdots\ .o_{n-1}.o_n \mid n \in I\!N^+ \ \wedge\ \exists c_1, ..., c_{n-1} \in \mathcal{CXT} : \\ (\forall 1 \leq i < n,\ o_i \in objs(c_{i-1})\ \wedge\ c_i = ref(o_i, c_{i-1}))\ \wedge\ o_n \in objs(c_{n-1})\}. \diamond$$

In fact, a reference path $o_1.o_2.\ \cdots\ .o_{n-1}.o_n$ in c_0 is used to reach object o_n in context c_{n-1} starting from object o_1 in c_0. Note that, each object o of a context c is considered as a reference path in c of length 1.

We are now ready to define a contextualized information base.

Definition 4.3 Contextualized information base. A *contextualized information base* is a tuple:

$$\langle\ \mathcal{O}, \mathcal{CXT}, \mathcal{N}, cnts\ \rangle$$

where $cnts$ is a partial function that associates a context $c \in \mathcal{CXT}$ with its contents $cnts(c) = \langle objs(c), lex(c), rf(c), in(c), isa(c), attr(c)\rangle$, such that: if $\langle o, p_f, p_t\rangle\ \in\ in(c) \cup isa(c) \cup attr(c)$ then $p_f, p_t \in \mathcal{RP}_c$. $\diamond$

The fact that the source and the destination of an instance-of, ISA, or attribute link of a context c are reference paths in c implies that (i) from within c, we can access only objects that "lie" on a reference path in c, and (ii) the related objects (i.e., last objects in the paths) are contextualized, i.e., they are viewed from the last accessed context navigating through the path. For example, in Figure 8, o_{11} is an attribute link in c_4 from $o_f.o_5$ to $o_t.o_8$. As contexts c_2 and c_3 are the last accessed contexts navigating through the paths $o_f.o_5$ to $o_t.o_8$, respectively, the objects related by the attribute link o_{11} (i.e., objects o_5 and o_8) are contextualized within contexts c_2 and c_3, respectively.

A contextualized information base is built from a set of user declarations. However, not every contextualized information base is valid. A valid contextualized information base should guarantee that the scope of each context is a directed acyclic graph, as well as the appropriate interaction between the different abstraction mechanisms. Formally, a contextualized information base is valid if its structure satisfies a number of validity constraints, that will be presented in subsection 4.4. The following two subsections, provide the appropriate definitions and notions for defining validity.

4.2 Predicates and functions

In this subsection, we define a number of predicates and functions which will be used to express the validity constraints of a contextualized information base.

1. The predicate $IsLink_c(o)$ expresses that object o is a link in context c, and is defined as follows:

$$\forall c \in \mathcal{CXT},\ o \in \mathcal{O}:$$

$$IsLink_c(o) \Leftrightarrow\ \exists p, p' \in \mathcal{RP}_c:\ \langle o, p, p' \rangle \in isa(c) \cup in(c) \cup attr(c).$$

2. $From_c : \mathcal{O} \longrightarrow \mathcal{RP}_c$.

   This function is defined w.r.t. a context c, takes as input a link o, and returns the source of o in c. That is:

$$\forall c \in \mathcal{CXT},\ o \in \mathcal{O},\ p \in \mathcal{RP}_c:$$

$$From_c(o) = p \Leftrightarrow\ \exists p' \in \mathcal{RP}_c:\ \langle o, p, p' \rangle \in isa(c) \cup in(c) \cup attr(c).$$

   We call this function *the source of* o in context c.

   For example, in Figure 8, $From_{c_4}(o_{11}) = o_f.o_5$.

3. $To_c : \mathcal{O} \longrightarrow \mathcal{RP}_c$.

   This function is defined w.r.t. a context c, takes as input a link o, and returns the destination of o in c. That is:

$$\forall c \in \mathcal{CXT},\ o \in \mathcal{O},\ p \in \mathcal{RP}_c:$$

$$To_c(o) = p \Leftrightarrow\ \exists p' \in \mathcal{RP}_c:\ \langle o, p', p \rangle \in isa(c) \cup in(c) \cup attr(c).$$

   We call this function *the destination of* o in context c.

   For example, in Figure 8, $To_{c_4}(o_{11}) = o_t.o_8$.

4. $Ref_c : \mathcal{RP}_c \longrightarrow \mathcal{CXT}$.

   This function is defined w.r.t. a context c, takes as input a reference path p in c and returns the reference of the last object in p w.r.t. the last accessed context navigating through p. That is:

$$\forall c \in \mathcal{CXT}, p \in \mathcal{RP}_c:\ \ Ref_c(p) = \begin{cases} ref(p, c), & \text{if } length(p) = 1 \\ Ref_{ref(first(p),c)}(rest(p)), & \text{if } length(p) > 1 \end{cases}$$

   Note that the above definition is recursive and terminates when $length(p) = 1$. We call this function *the reference of* p in context c.

   For example, in Figure 6, $Ref_c(o_1) = c_1$, $Ref_c(o_1.o) = c_2$, and $Ref_c(o_2.o) = c_4$.

4.3 The notion of link path

In this subsection, we define the notion of link path and traversal path. As the source and the destination of a link (attribute, instance-of, and ISA) are objects (reached through reference paths), links provide a means to traverse from one object to another. For each such traversal, there is a corresponding sequence of traversed links. This sequence of traversed links (recall that links are objects themselves) constitutes a kind of path which we shall call *link path*. Link paths, similar to reference paths, are given in a context c.

Definition 4.4 Link path. The set of *link paths* in a context c, denoted by $\mathcal{LP}_c$, is defined as follows:

$$\mathcal{LP}_c = \{o_1.o_2.\ \cdots\ .o_{n-1}.o_n \mid n \in I\!N^+ \ \wedge\ \forall 1 \leq i < n :$$
$$IsLink_c(o_i)\ \wedge\ IsLink_c(o_{i+1})\ \wedge\ (From_c(o_{i+1}) = To_c(o_i)\ \vee\ From_c(o_{i+1}) = o_i\ \vee$$
$$To_c(o_i) = o_{i+1})\}. \diamond$$

Intuitively, a link path in c is a sequence of links of c, where the source of each link in the sequence coincides with the destination of the previous link, or the previous link itself. A link path in c, whose all members are either attributes, or instance-of, or ISA links in c, will be called *attribute*, *instance-of*, or *ISA path* in c, respectively. For example, in Figure 8, there are three link paths in context c_4: $p_1 = o_{12}.o_{14}$, $p_2 = o_{12}.o_{16}$, and $p_3 = o_{11}$.

As we have seen in subsection 3.1, a context is of type *link context* if it is the reference of an attribute or instance-of link. Every link context must contain the special objects o_f and o_t with names `from` and `to`, respectively. Additionally, if a link has as reference a link context c then the references of objects o_f and o_t in c should be the source and destination reference[7] of the link, respectively. We denote the set of link contexts by $\mathcal{LCXT}$. Clearly, it holds: $\mathcal{LCXT} \subset \mathcal{CXT}$.

Let c be a link context. We shall call *traversal path in* c, any link path in c that traverses from an object reached through a reference path starting from o_f, to an object reached through a reference path starting from o_t. Specifically, the traversal path should satisfy the following: (i) the source of its first link is a reference path starting from o_f, and (ii) the destination of its last link is a reference path starting from o_t. Intuitively, a traversal path in a link context traverses from an object in the source reference, to an object in the destination reference of the corresponding link.

Definition 4.5 Traversal Path. The set of all traversal paths in a link context c, denoted by $\mathcal{TP}_c$, is defined as follows[8]:

$$\mathcal{TP}_c = \{p \in \mathcal{LP}_c \mid \mathit{first}(From_c(\mathit{first}(p))) = o_f\ \wedge\ \mathit{first}(To_c(\mathit{last}(p))) = o_t\}. \diamond$$

For example, in Figure 8, there are two traversal paths within context c_4: $p_1 = o_{12}.o_{14}$ and $p_3 = o_{11}$.

The set of all attribute paths in c which are also traversal paths is denoted as $\mathcal{ATP}_c$. The set of all instance-of paths in c which are also traversal paths is denoted as $\mathcal{INTP}_c$, and the set of instance-of traversal paths in c of length one is denoted by $\mathcal{INTP}^1_c$, i.e., $\mathcal{INTP}^1_c = \{p \in \mathcal{INTP}_c \mid length(p) = 1\}$.

[7]Recall that we refer to the reference of the source (resp. destination) of a link as the *source* (resp. *destination*) reference of that link.

[8]The function $\mathit{first}(p)$ returns the first element of path p.

4.4 Validity constraints

In this subsection, we present the constraints that every valid contextualized information base should satisfy. These validity constraints guarantee (i) that the scope of every context is a directed acyclic graph, (ii) the appropriate interaction between the different abstraction mechanisms, as described in subsection 3.1, (iii) termination of the refinement inheritance process, and (iv) the existence of a special context that contains built-in information shared by all contexts.

Validity Constraint 4.1 Acyclicity. There is no reference path from a context c to c.

$$\forall c \in \mathcal{CXT} : \quad \nexists p \in \mathcal{RP} : \ Ref_c(p) = c. \diamond$$

As we have indicated in Section 2, the scope of every context is a directed acyclic graph.

Validity Constraint 4.2 Special objects of a link context. The objects of a link context must contain the special objects, o_f and o_t, with names `from` and `to`, respectively. That is:

$$\forall c \in \mathcal{LCXT} : \ o_f, o_t \in objs(c) \ \wedge \ \texttt{from} \in names(o_f, c) \ \wedge \ \texttt{to} \in names(o_t, c). \diamond$$

This validity constraint is justified and exemplified in subsection 3.1.1.

Validity Constraint 4.3 Interaction between attribution and contextualization. Let c be a context and p_f, p_t be reference paths in c. If o is an attribute link in c from p_f to p_t, and c_a is the reference of o in c, then the references of the special objects o_f and o_t in c_a are the references of p_f and p_t in c. Additionally, every traversal path in c is an attribute path in c.

$$\forall o \in \mathcal{O}, \ c \in \mathcal{CXT}, \ p_f, p_t \in \mathcal{RP}, \ c_a \in \mathcal{LCXT} :$$
$$\langle o, p_f, p_t \rangle \in attr(c) \ \wedge \ c_a = ref(o, c) \ \Rightarrow$$
$$ref(o_f, c_a) = Ref_c(p_f) \ \wedge \ ref(o_t, c_a) = Ref_c(p_t) \ \wedge \ \mathcal{TP}_c = \mathcal{ATP}_c. \diamond$$

This validity constraint is justified and exemplified in subsection 3.1.1.

Validity Constraint 4.4 Interaction between classification and contextualization. Let c be a context and p_f, p_t be reference paths in c. If o is an instance-of link in c from p_f to p_t, and c_{in} is the reference of o in c, then the references of the special objects o_f and o_t in c_{in} are the references of p_f and p_t in c. Additionally, every traversal path in c is an instance-of link in c.

$$\forall o \in \mathcal{O}, \ c \in \mathcal{CXT}, \ p_f, p_t \in \mathcal{RP}, \ c_{in} \in \mathcal{LCXT} :$$

$$\langle o, p_f, p_t \rangle \in in(c) \ \wedge \ c_{in} = ref(o, c) \ \Rightarrow$$

$$ref(o_f, c_{in}) = Ref_c(p_f) \ \wedge \ ref(o_t, c_{in}) = Ref_c(p_t) \ \wedge \ \mathcal{TP}_c = \mathcal{INTP}^1_c. \diamond$$

This validity constraint is justified and exemplified in subsection 3.1.2.

Validity Constraint 4.5 Interaction between generalization and contextualization. Let c be a context and o_1, o_2 be objects of c. If there is an ISA link in c from o_1 to o_2 then there should not be any reference path from the reference of o_2 to the reference of o_1.

$$\forall o, o_1, o_2 \in \mathcal{O},\ c, c_1, c_2 \in \mathcal{CXT} :$$

$$\langle o, o_1, o_2 \rangle \in isa(c) \ \wedge\ c_1 = ref(o_1, c) \ \wedge\ c_2 = ref(o_2, c) \ \Rightarrow\ \nexists p \in \mathcal{RP} :\ Ref_{c_2}(p) = c_1. \diamond$$

As we have described in subsection 3.1.3, if the left part of the above validity constraint is true then c_1 refines c_2 through an inheritance process. The constraint $\nexists p \in \mathcal{RP} :\ Ref_{c_2}(p) = c_1$ guarantees that the inheritance process terminates.

Validity Constraint 4.6 Interaction between classification, generalization, and contextualization. Let c be a context and p, p_1, p_2 be reference paths in c. If o_1 is an instance-of link in c from p to p_1, o is an ISA link in c from p_1 to p_2, and o_2 is an instance-of link in c from p to p_2, then there should not be any reference path from the reference of o_2 to the reference of o_1.

$$\forall o, o_1, o_2 \in \mathcal{O},\ c, c_1, c_2 \in \mathcal{CXT},\ p, p_1, p_2 \in \mathcal{RP} :$$

$$\langle o_1, p, p_1 \rangle \in in(c) \ \wedge\ \langle o, p_1, p_2 \rangle \in isa(c) \ \wedge\ \langle o_2, p, p_2 \rangle \in in(c) \ \wedge\ ref(o_1, c) = c_1 \ \wedge\ ref(o_2, c) = c_2$$

$$\Rightarrow\ \nexists p' \in \mathcal{RP} :\ Ref_{c_2}(p') = c_1. \diamond$$

As we have described in subsection 3.1.4, if the left part of the above validity constraint is true then c_1 refines c_2 through an inheritance process. The constraint $\nexists p' \in \mathcal{RP} :\ Ref_{c_2}(p') = c_1$ guarantees that the inheritance process terminates.

As a final validity constraint, we assume that in every contextualized information base, there is a special context C_{empty}, called *empty context*, which contains all built-in information[9] (objects, contexts, names) shared by all contexts. For example, in an object-oriented database, the empty context will contain all built-in classes and their relationships.

Validity Constraint 4.7 The empty context (C$_{empty}$). There is a special context C_{empty}, called *empty context*, which contains all built-in information. $\diamond$

We are now ready to define validity in contextualized information bases.

Definition 4.6 Valid Contextualized Information Base. Let $CIB = \langle\ \mathcal{O}, \mathcal{CXT}\ \mathcal{N},\ cnts\ \rangle$ be a contextualized information base. We say that CIB is $valid$ iff CIB satisfies the above validity constraints. $\diamond$

In the rest of the paper, we consider only valid contextualized information bases.

[9]Built-in information is not needed to be created explicitly by the users.

4.5 Model theory

Each contextualized information base is built from a set of user declarations. Based on these declarations, new information can be derived, enriching the contents of declared contexts and deriving new contexts. In this subsection, we present a model theory and define the logical implications of a (valid) contextualized information base. The logical implications of a contextualized information base allow to answer queries regarding the objects of a context c, the names of an object o in a context c, the reference of an object o in a context c, the instance-of, ISA, and attribute relationships in a context c, as well as context refinement.

Definition 4.7 Interpretation. An $interpretation\ I$ of a contextualized information base $CIB = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, \mathit{cnts} \rangle$ is a tuple $\langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, \mathit{cnts}^I, \precsim^I \rangle$, where cnts^I is a partial function that associates a context with its contents, and $\precsim^I$ is a binary relation between contexts. $\diamond$

From the above definition, it is obvious that an interpretation I of a contextualized information base CIB is also a contextualized information base, extended with the context refinement relation.

Definition 4.8 Model. Let $CIB = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, \mathit{cnts} \rangle$ be a contextualized information base. An interpretation $I = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, \mathit{cnts}^I, \precsim^I \rangle$ is a $model$ of CIB iff I satisfies the following model constraints. $\diamond$

Model Constraint 4.1 Satisfaction of user declarations. Every model I satisfies the CIB user declarations.

1. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT}: \quad o \in \mathit{objs}(c) \ \Rightarrow\ o \in \mathit{objs}^I(c) \ \wedge\ \mathit{names}(o,c) \subseteq \mathit{names}^I(o,c).$
2. $\forall c \in \mathcal{CXT}: \quad \mathit{in}(c) \subseteq \mathit{in}^I(c) \ \wedge\ \mathit{isa}(c) \subseteq \mathit{isa}^I(c) \ \wedge\ \mathit{attr}(c) \subseteq \mathit{attr}^I(c).$
3. $\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT}: \ \mathit{ref}(o,c) = c' \ \Rightarrow\ \mathit{ref}^I(o,c) = c'.$ $\diamond$

Model Constraint 4.2 Context Refinement. A context c *refines* a context c' ($c \precsim c'$) iff (i) every object of c' is also an object of c, (ii) the names of every object o in c' are included in the names of o in c, (iii) every instance-of, ISA, or attribute relationship in c' also holds in c, and (iv) the reference of every object o of c' is either undefined or refined by the reference of o in c.

$$\forall c, c' \in \mathcal{CXT}:$$

$$c \precsim^I c' \quad \Leftrightarrow \quad \begin{array}{lll} (\mathit{objs}^I(c') & \subseteq & \mathit{objs}^I(c) \quad \wedge \\ \mathit{attr}^I(c') & \subseteq & \mathit{attr}^I(c) \quad \wedge \\ \mathit{in}^I(c') & \subseteq & \mathit{in}^I(c) \quad \wedge \\ \mathit{isa}^I(c') & \subseteq & \mathit{isa}^I(c) \quad \wedge \\ \multicolumn{3}{l}{(\forall o \in \mathit{objs}(c'),\ c_1 \in \mathcal{CXT},\ \exists c_2 \in \mathcal{CXT}:} \\ \multicolumn{3}{l}{\qquad \mathit{names}^I(o,c') \subseteq \mathit{names}^I(o,c) \ \wedge} \\ \multicolumn{3}{l}{\qquad \mathit{ref}^I(o,c') = c_1 \ \Rightarrow\ \mathit{ref}^I(o,c) = c_2 \ \wedge\ c_2 \precsim^I c_1).\ \diamond} \end{array}$$

This model constraint is justified and exemplified in subsection 3.1.3.

Model Constraint 4.3 Contextualized binary instance-of and ISA. Let c be a context and p_1, p_2 be reference paths in c. There is an ISA link in c from p_1 to p_2 iff every instance of p_1 in c is also an instance of p_2 in c.

$$\forall c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}_c :$$

$$Isa_c(p_1, p_2) \ \Leftrightarrow\ \exists o \in \mathcal{O} :\ \ \langle o, p_1, p_2 \rangle \in isa^I(c).$$
$$In_c(p_1, p_2) \ \Leftrightarrow\ \exists o \in \mathcal{O} :\ \ \langle o, p_1, p_2 \rangle \in in^I(c).$$
$$Isa_c(p_1, p_2) \ \Leftrightarrow\ (\forall p \in \mathcal{RP} :\ \ In_c(p, p_1) \ \Rightarrow\ In_c(p, p_2)). \diamond$$

This is the contextualized version of the classical semantics of ISA.

Model Constraint 4.4 Interaction between generalization and contextualization. Let c be a context and o_1, o_2 be objects of c. If there is an ISA link in c from o_1 to o_2 then the reference of o_1 in c refines the reference of o_2 in c.

$$\forall o_1, o_2 \in \mathcal{O},\ c, c_1 \in \mathcal{CXT},\ \exists c_2 \in \mathcal{CXT} :$$

$$Isa_c(o_1, o_2)\ \wedge\ c_1 = ref^I(o_1, c)\ \ \Rightarrow\ \ c_2 = ref^I(o_2, c)\ \wedge\ c_1 \precsim^I c_2. \diamond$$

This model constraint is justified and exemplified in subsection 3.1.3.

Model Constraint 4.5 Interaction between classification, generalization and contextualization. Let c be a context and p, p_1, p_2 be reference paths in c. If o_1 is an instance-of link in c from p to p_1, there is an ISA link in c from p_1 to p_2, and o_2 is an instance-of link in c from p to p_2, then the reference of o_1 in c refines the reference of o_2 in c.

$$\forall o_1, o_2 \in \mathcal{O},\ c, c_1 \in \mathcal{CXT},\ p, p_1, p_2 \in \mathcal{RP},\ \exists c_2 \in \mathcal{CXT} :$$

$$\langle o_1, p, p_1 \rangle \in in^I(c)\ \wedge\ Isa_c(p_1, p_2)\ \wedge\ \langle o_2, p, p_2 \rangle \in in^I(c)\ \wedge\ c_2 = ref^I(o_2, c)\ \ \Rightarrow$$

$$c_1 = ref^I(o_1, c)\ \wedge\ c_1 \precsim^I c_2. \diamond$$

This model constraint is justified and exemplified in subsection 3.1.4.

Model Constraint 4.6 Inheritance of built-in information. Every context refines the empty context.

$$\forall c \in \mathcal{CXT} :\quad c \precsim^I C_{empty}. \diamond$$

This model constraint expresses that the contents of the empty context should be present in every context.

Having defined the models of a contextualized information base CIB, we are now ready to define the logical implications of CIB. Intuitively, an atom α is logically implied by CIB iff the information in α is reflected in the structure of every model of CIB.

Definition 4.9 Logical implications. Let CIB be a contextualized information base $\langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, cnts \rangle$. The *logical implications* of CIB are defined as follows:

$\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP},\ n \in \mathcal{N}$:

- $CIB \models Obj_c(o)$ iff $o \in objs^I(c)$, for every model I of CIB.
- $CIB \models Name_c(o,n)$ iff $n \in names^I(o,c)$, for every model I of CIB.
- $CIB \models ref(o,c) = c'$ iff $ref^I(o,c) = c'$, for every model I of CIB.
- $CIB \models defined(ref(o,c))$ iff $\exists c'' \in \mathcal{CXT} :\ ref^I(o,c) = c''$, for every model I of CIB.
- $CIB \models Isa_c(o,p_1,p_2)$ iff $\langle o,p_1,p_2 \rangle \in isa^I(c)$, for every model I of CIB.
- $CIB \models Isa_c(p_1,p_2)$ iff $\exists o' \in \mathcal{O} :\ \langle o',p_1,p_2 \rangle \in isa^I(c)$, for every model I of CIB.
- $CIB \models In_c(o,p_1,p_2)$ iff $\langle o,p_1,p_2 \rangle \in in^I(c)$, for every model I of CIB.
- $CIB \models In_c(p_1,p_2)$ iff $\exists o' \in \mathcal{O} :\ \langle o',p_1,p_2 \rangle \in in^I(c)$, for every model I of CIB.
- $CIB \models Attr_c(o,p_1,p_2)$ iff $\langle o,p_1,p_2 \rangle \in attr^I(c)$, for every model I of CIB.
- $CIB \models c \precsim c'$ iff $c \precsim^I c'$, is satisfied by every model I of CIB. $\diamond$

We refer to the atoms following $\models$, as *contextual atoms* and to the corresponding predicates as *contextual predicates*.

4.6 Inference rules

In this subsection, we present a set of sound and complete inference rules that allow us to derive the logical implications of a (valid) contextualized information base. Starting from user declarations, new information is derived by applying the inference rules the usual way. Intuitively, through the inference rules, new contexts are derived and the contents of user declared contexts are enriched.

Definition 4.10 Derivations. Let $CIB = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, cnts \rangle$ be a contextualized information base, and let α be a contextual atom. We say that α is $derived$ from CIB ($CIB \vdash \alpha$) iff α is derived from the following inference rules starting derivation from the CIB user declarations. $\diamond$

Inference Rule 4.1 Predicate initializations. Predicates are initialized based on the CIB user declarations.

1. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT} :\quad o \in objs(c) \ \Rightarrow\ Obj_c(o)$.
2. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT}, n \in \mathcal{N} :\quad n \in names(o,c) \ \Rightarrow\ Name_c(o,n)$.
3. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP} :\quad \langle o,p_1,p_2 \rangle \in in(c) \ \Rightarrow\ In_c(o,p_1,p_2)$.

4. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad \langle o, p_1, p_2 \rangle \in isa(c) \ \Rightarrow \ Isa_c(o, p_1, p_2).$
5. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad \langle o, p_1, p_2 \rangle \in attr(c) \ \Rightarrow \ Attr_c(o, p_1, p_2).$
6. $\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT}: \quad ref(o, c) = c' \ \Rightarrow \ defined(ref(o, c)).\ \diamond$

Inference Rule 4.2 Contextualized binary instance-of and ISA. Binary instance-of and ISA relationships are derived from ternary instance-of and ISA relationships.

1. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad In_c(o, p_1, p_2) \ \Rightarrow \ In_c(p_1, p_2).$
2. $\forall o \in \mathcal{O},\ c \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad Isa_c(o, p_1, p_2) \ \Rightarrow \ Isa_c(p_1, p_2).\ \diamond$

Inference Rule 4.3 Contextualized ISA reflexivity and transitivity.

1. ISA reflexivity.

   Let c be a context. For every reference path p in c, there is an ISA link in c from p to p.

   $$\forall c \in \mathcal{CXT},\ p \in \mathcal{RP}_c: \ Isa_c(p, p).$$

2. ISA transitivity.

   Let c be a context and p_1, p_2, p_3 be reference paths in c. If there are two ISA links in c from p_1 to p_2, and from p_2 to p_3, respectively, then there is an ISA link in c from p_1 to p_3.

   $$\forall\ c \in \mathcal{CXT},\ p_1, p_2, p_3 \in \mathcal{RP}:$$
   $$Isa_c(p_1, p_2) \ \wedge \ Isa_c(p_2, p_3) \ \Rightarrow \ Isa_c(p_1, p_3).\ \diamond$$

The above inference rules are contextualized versions of the ISA reflexivity and ISA transitivity properties of conventional object-oriented systems.

Inference Rule 4.4 Context Refinement. If a context c *refines* a context c' ($c \precsim c'$) then (i) every object of c' is also an object of c, (ii) the names of every object o in c' are included in the names of o in c, (iii) every instance-of, ISA, or attribute relationship in c' also holds in c, and (iv) the reference of every object o of c' is either undefined or refined by reference of o in c.

1. $\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT}: \quad c \precsim c' \ \wedge \ Obj_{c'}(o) \ \Rightarrow \ Obj_c(o).$
2. $\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT}: \quad c \precsim c' \ \wedge \ Name_{c'}(o, n) \ \Rightarrow \ Name_c(o, n).$
3. $\forall o \in \mathcal{O}, c, c' \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad c \precsim c' \ \wedge \ In_{c'}(o, p_1, p_2) \ \Rightarrow \ In_c(o, p_1, p_2).$
4. $\forall o \in \mathcal{O}, c, c' \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad c \precsim c' \ \wedge \ Isa_{c'}(o, p_1, p_2) \ \Rightarrow \ Isa_c(o, p_1, p_2).$
5. $\forall o \in \mathcal{O}, c, c' \in \mathcal{CXT},\ p_1, p_2 \in \mathcal{RP}: \quad c \precsim c' \ \wedge \ Attr_{c'}(o, p_1, p_2) \ \Rightarrow \ Attr_c(o, p_1, p_2).$
6. $\forall o \in \mathcal{O},\ c, c' \in \mathcal{CXT}:$

   $$c \precsim c' \ \wedge \ defined(ref(o, c')) \ \Rightarrow \ defined(ref(o, c)) \ \wedge \ ref(o, c) \precsim ref(o, c').\ \diamond$$

Inference Rule 4.5 Refinement is reflexive and transitive. The refinement relation between contexts is reflexive and transitive.

1. Reflexivity.

$$\forall c \in \mathcal{CXT}: \quad c \precsim c.$$

2. Transitivity.

$$\forall c_1, c_2, c_3 \in \mathcal{CXT}: \quad c_1 \precsim c_2 \ \wedge \ c_2 \precsim c_3 \quad \Rightarrow \quad c_1 \precsim c_3. \diamond$$

Inference Rule 4.6 Contextualized instance upward inheritance. Let c be a context and p, p_1, p_2 be reference paths in c. If there is an instance-of link in c from p to p_1, and an ISA link in c from p_1 to p_2, then there is an instance-of link in c from p to p_2.

$$\forall\ c \in \mathcal{CXT},\ p, p_1, p_2 \in \mathcal{RP}:$$

$$In_c(p, p_1)\ \wedge\ Isa_c(p_1, p_2) \quad \Rightarrow \quad In_c(p, p_2). \diamond$$

In conventional object-oriented systems, instances of classes are also instances of their superclasses. The above inference rule is the contextualized version of this property.

Inference Rule 4.7 Interaction between generalization and contextualization. Let c be a context and o_1, o_2 be objects of c. If there is an ISA link in c from o_1 to o_2 then the reference of o_1 in c refines the reference of o_2 in c.

$$\forall o_1, o_2 \in \mathcal{O},\ c \in \mathcal{CXT}:$$

$$Isa_c(o_1, o_2)\ \wedge\ defined(ref(o_2, c)) \quad \Rightarrow \quad defined(ref(o_1, c))\ \wedge\ ref(o_1, c) \precsim ref(o_2, c). \diamond$$

This inference rule is justified and exemplified in subsection 3.1.3.

Inference Rule 4.8 Interaction between classification, generalization, and contextualization. Let c be a context and p, p_1, p_2 be reference paths in c. If o_1 is an instance-of link in c from p to p_1, there is an ISA link in c from p_1 to p_2, and o_2 is an instance-of link in c from p to p_2, then the reference of o_1 in c refines the reference of o_2 in c.

$$\forall o_1, o_2 \in \mathcal{O},\ c \in \mathcal{CXT},\ p, p_1, p_2 \in \mathcal{RP}:$$

$$In_c(o_1, p, p_1)\ \wedge\ Isa_c(p_1, p_2)\ \wedge\ In_c(o_2, p, p_2)\ \wedge\ defined(ref(o_2, c)) \quad \Rightarrow$$

$$defined(ref(o_1, c))\ \wedge\ ref(o_1, c) \precsim ref(o_2, c). \diamond$$

This inference rule is justified and exemplified in subsection 3.1.4.

Inference Rule 4.9 Inheritance of built-in information. Every context refines the empty context.

$$\forall c \in \mathcal{CXT} : \quad c \precsim C_{empty}. \diamond$$

The following proposition expresses that the above inference rules are sound and complete. That is, every logical implication of a contextualized information base is derived from the inference rules, and vice versa.

Proposition 4.1 Soundness and Completeness. Let CIB be a contextualized information base, and let α be a contextual atom. It holds: $CIB \models \alpha \Leftrightarrow CIB \vdash \alpha$. $\diamond$

4.7 The equivalence and refinement relations

In this section, we give the formal definitions of the equivalence and refinement relations, described informally in subsection 3.1.3, and we prove some of their properties. First, we show how a context structure can be represented as a labelled directed acyclic graph. To simplify the presentation, when $ref(o, c)$ is undefined for an object o of a context c, we write $ref(o, c) = NIL$, where NIL is a special kind of context with no associated contents.

4.7.1 Context structure as labelled directed graphs

In this subsection, we recall briefly the notion of “labelled directed graph” and “graph isomorphism”. The notion of isomorphic graphs is used in the following subsection to define context equivalence. We also show how a context structure is represented as a labelled directed acyclic graph.

Definition 4.11 Labelled directed graph. A labelled directed graph is a triplet $G = \langle V, E, L\rangle$ where V is a finite set of vertices, $E \subseteq V \times V$ is the set of edges (which connect certain pair of vertices) and L is a function which assigns each vertex and each edge a label.

Definition 4.12 Labelled directed graph isomorphism. Two labelled directed graphs, $G_1 = \langle V_1, E_1, L_1\rangle$ and $G_2 = \langle V_2, E_2, L_2\rangle$ are isomorphic iff there is a bijective function (isomorphism) $\pi : V_1 \rightarrow V_2$ such that

1. $\langle v_1, v_2\rangle \in E_1$ iff $\langle \pi(v_1), \pi(v_2)\rangle \in E_2$,
2. $L_1(v) = L_2(\pi(v))$ for all $v \in V_1$, and
3. $L_1(\langle v_1, v_2\rangle) = L_2(\langle \pi(v_1), \pi(v_2)\rangle)$ for all $\langle v_1, v_2\rangle \in E_1$.

Let $Iso(G_1, G_2)$ denote the set of all isomorphisms between G_1 and G_2. $\diamond$

Lemma 4.1 Graph isomorphism transitivity. If graphs G_1, G_2 are isomorphic and graphs G_2, G_3 are also isomorphic, then graphs G_1, G_3 are isomorphic too. $\diamond$

Proof: If $\pi \in Iso(G_1, G_2)$ and $\phi \in Iso(G_2, G_3)$, it is easy to prove that $\pi \circ \phi$ is an isomorphism between graphs G_1 and G_3. $\square$

By context structure, we mean a structure of contexts and their nested subcontexts. In the structure, we don't take into account instance-of, ISA and attribute relationships, as well as the names of the objects, but we do take into account the objects of the contexts, which are represented as labels of the graph edges. Thus, a context structure can be seen as a labelled directed graph as follows:

- the contexts are represented as vertices of the graph,
- the subcontext c' of a context c is represented by the edge $\langle c, c' \rangle$,
- the navigation from a context c to its subcontext c' through the reference of an object o in c, i.e. $ref(o, c) = c'$, is represented by $o \in L(\langle c, c' \rangle)$, i.e. the label of edge $\langle c, c' \rangle$ is a set containing the object o. If there is another object o' such that $ref(o', c) = c'$ then $o' \in L(\langle c, c' \rangle)$ as well. In other words, the label of an edge $\langle c, c' \rangle$ is the set of all objects of c with reference c'.
- $L(v) = \{\}$, for all $v \in V$, as contexts are assigned no name.

Figure 14 shows how a context structure is represented as a labelled directed graph.

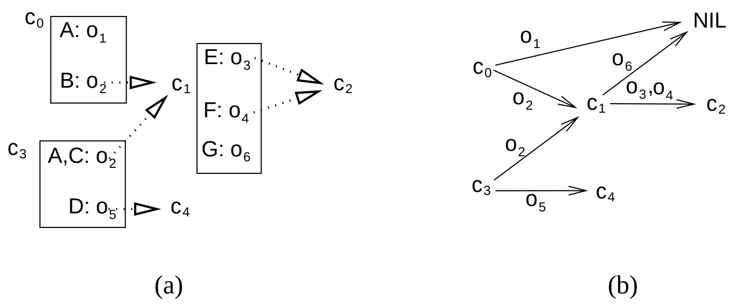


Figure 14: (a) A context structure, (b) its representation as a labelled directed graph.

Note that since each object within a context has either a NIL reference or a reference to another context and the objects of a context are represented as labels of edges originating from that context, it is easy to see that: if c is a context then

$$objs(c) = \bigcup_{\langle c, c' \rangle \in E} L(\langle c, c' \rangle)$$

Each context is associated with a context structure which contains the context itself and its nested subcontexts. The labelled directed graph induced by the context structure of a context is denoted by $GR(c)$ and is defined formally as follows:

Definition 4.13 Labelled directed graph induced by a context c, $GR(c)$. The labelled directed graph induced by the context c is defined as follows:

$$\forall c \in \mathcal{CXT} : \quad GR(c) = \langle V, E, L \rangle$$

where

$$V = \{c\} \cup \{c' \in \mathcal{CXT} \mid c' = Ref_c(p),\ p \in \mathcal{RP}_c\}$$

$$E = \{\langle c_1, c_2\rangle \mid c_1, c_2 \in V \ \wedge \ \exists o \in \mathit{objs}(c_1): \ \mathit{ref}(o, c_1) = c_2\}$$

$$\forall \langle c_1, c_2\rangle \in E: \ L(\langle c_1, c_2\rangle) = \bigcup_{\mathit{ref}(o,c_1)=c_2} \{o\}. \diamond$$

Note that every labelled directed graph induced by a context c is acyclic. Now, we are ready to define the equivalence relation between contexts.

4.7.2 Context equivalence

Definition 4.14 Context equivalence ($\sim$). Let c, c' be two contexts and $GR(c) = \langle V, E, L\rangle$, $GR(c')$ the graphs induced by c and c', respectively. *Context equivalence* is a relation between c and c', denoted by $c \sim c'$, and is defined as follows:

$$\begin{aligned} c \sim c' \quad \Leftrightarrow \quad & (c = c' = \mathit{NIL}) \\ & \vee \\ & (\ \exists \pi \in \mathit{Iso}(GR(c), GR(c')): \ \pi(c) = c' \wedge \\ & \qquad \forall v \in V: \\ & \qquad\quad \mathit{attr}(v) = \mathit{attr}(\pi(v)) \quad \wedge \\ & \qquad\quad \mathit{in}(v) = \mathit{in}(\pi(v)) \quad \wedge \\ & \qquad\quad \mathit{isa}(v) = \mathit{isa}(\pi(v)) \quad \wedge \\ & \qquad (\forall o \in \mathit{objs}(v): \ \mathit{names}(o, v) = \mathit{names}(o, \pi(v)))). \diamond \end{aligned}$$

Two contexts are equivalent ($c \sim c'$) iff (i) c and c' are identical (possibly NIL), or (ii) they have an isomorphic context structure and the contents (except references) of each context coincide. Intuitively, this means that two contexts are equivalent iff (i) they are identical, or (ii) their contents (except references) coincide and the references of their common objects are equivalent.

Graph Isomorphism Problem: The complexity of an algorithm which decides whether two given contexts are equivalent is the same to the complexity of an algorithm which decides whether the labelled directed acyclic graphs induced by given contexts are isomorphic. This problem is polynomially equivalent to the well-known *graph isomorphism problem* [33, 40, 19, 32]. The graph isomorphism problem is clearly in the class of NP, and it is not known whether it is in P. It is also unknown whether the problem is NP-complete.

An important property of the relation $\sim$ is that it is equivalence. Thus, in case context identifiers are not important, equivalent contexts can be used interchangeably for modeling information.

Proposition 4.2 Relation $\sim$ is an equivalence relation. The relation $\sim$ on the set of contexts $\mathcal{CXT}$ is an equivalence relation, i.e., reflexive, transitive, and symmetric. $\diamond$

4.7.3 Context refinement

We now define the refinement relation between two contexts.

Definition 4.15 Context refinement. The *refinement* relation between two contexts c, c', denoted by $c \precsim c'$, is recursively defined as follows:

$$\begin{array}{lll} c \precsim c' & \Leftrightarrow & (c' = NIL) \\ & & \vee \\ & & (objs(c') \subseteq objs(c) \wedge \\ & & attr(c') \subseteq attr(c) \wedge \\ & & in(c') \subseteq in(c) \wedge \\ & & isa(c') \subseteq isa(c) \wedge \\ & & (\forall o \in objs(c') : \; names(o,c') \subseteq names(o,c) \wedge \\ & & \qquad ref(o,c) \precsim ref(o,c'))). \diamond \end{array}$$

The recursion in the definition of $c \precsim c'$ stops when c' is NIL, or $c \precsim c'$ is called again. Note that due to Validity Constraint 4.1, the recursion always terminates. Intuitively, a context c refines a context c' ($c \precsim c'$), if the contents of c' except references are subset of the contents of c, and the reference of an object o in c' is either undefined or refined by the reference of o in c. For example, in Figure 12 contexts c_9 and c_{10} refine context c_8, and contexts c_4, c_5 refine context c_3.

If both refinements $c \precsim c'$ and $c' \precsim c$ hold, then the contents of c and c' are intuitively the same. Indeed the following theorem shows that the refinement relation is partial order up to the equivalence relation.

Proposition 4.3 . The refinement relation on the set of contexts $\mathcal{CXT}$ is a partial order up to the equivalence relation $\sim$, i.e., it is reflexive, transitive, and antisymmetric[10]. $\diamond$

A contextualized information base CIB is an incomplete model of the real world. This implies that the conditions of Definition 4.15 may hold in CIB but not in the real world, and conversely, they may hold in the real world but not in CIB. Thus, we cannot use Definition 4.15 directly to derive $c \precsim c'$, for contexts c and c'. Indeed, the refinement relation is derived only through our inference rules.

5 Contextualization in Telos

Our contextualization mechanism is generic and can be applied to any semantic data model that supports the traditional abstraction mechanisms of classification, generalization, and attribution. In this section, we show how a specific data model can accommodate our contextualization mechanism. In particular, we choose for this purpose the Telos data model.

Telos [42, 34] is a knowledge representation language originally designed for information system development applications. A Telos information base consists of two kinds of objects: *individuals* and *attributes*. Objects are organized along three dimensions, referred to as the *classification*, *generalization*, and *attribution* dimensions.

The classification dimension calls for each object to be an instance of one or more *classes*. Classes are themselves objects, therefore they can be instances of other, more abstract classes. Generally, objects are classified

[10]That is $c \precsim c'$ and $c' \precsim c$ implies $c \sim c'$.

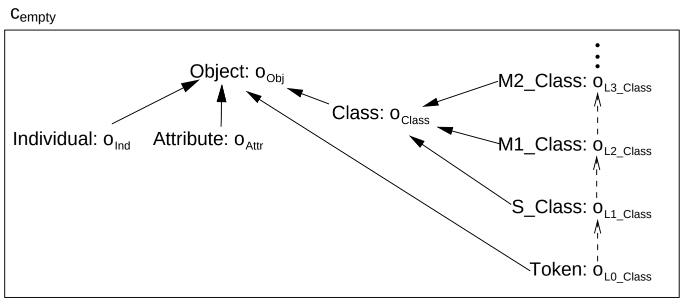


Figure 15: Context C_{empty}.

into:

tokens, i.e., objects having no instances and intended to represent atomic entities in the domain of discourse;

simple classes, i.e., objects having only tokens as instances;

metaclasses, i.e., objects having only simple classes as instances;

metametaclasses, and so on.

This classification defines an unbounded hierarchy of levels of ever more abstract objects. All tokens are classified under the class $\texttt{L}_0\texttt{\_Class}$, all simple classes under the class $\texttt{L}_1\texttt{\_Class}$, all metaclasses under the class $\texttt{L}_2\texttt{\_Class}$, and so on, and $\texttt{L}_0$, $\texttt{L}_1$, $\texttt{L}_2$, etc. are called *instantiation levels*. Classification is treated as a form of weak typing mechanism: if an attribute a is instance of an attribute class a' then the source of a should be instance of the source of a', and the destination of a should be instance of the destination of a'.

Classes at the same instantiation level can be specialized along *generalization* or *ISA hierarchies*.

We will see below how the Telos features are incorporated into our contextualized framework. For our discussion, we consider a variant of Telos, where instance-of and ISA links are also objects.[11]

To support Telos, the special context C_{empty} contains all built-in objects of Telos and their relationships. Specifically, the contents of C_{empty} include[12] (see Figure 15):

- Objects o_{Ind}, o_{Attr}, named `Individual` and `Attribute`, respectively. These objects represent the classes of individuals and attribute objects, respectively.
- Objects o_{L_0}, o_{L_1}, o_{L_2}, and so on, named `Token`, `S_Class`, `M1_Class`, and so on, respectively. These objects represent the instantiation levels of *Tokens*, *Simple Classes*, *Metaclasses*, and so on, respectively.
- Objects o_{Obj} and o_{Class}, named `Object` and `Class`, respectively. These objects represent the class of objects, and the class of classes, respectively.
- Instance-of links and ISA links that represent relationships between the objects of C_{empty}.

By definition the contents of the empty context are shared by any context. Therefore, the objects of C_{empty} are also objects of any context c. We say that an object o is *individual in context* c, iff there is an instance-of link in c

[11]O-Telos is such a variant of Telos, used in the deductive object base ConceptBase [27].

[12]In the figure, it is not shown that objects o_{Ind}, o_{Attr}, o_{Obj}, and o_{Class} are instances of the object o_{Class}.

from o to o_{Ind}. Similarly, an object o is *attribute in context* c, iff there is an instance-of link in c from o to o_{Attr}. A reference path p may be associated with an *instantiation level* i *in a context* c, iff there is an instance-of link in c from p to object o_{L_i}. The function $Level_c(p)$ returns the instantiation level of a reference path p in context c.

$$\forall c \in \mathcal{CXT}, p \in \mathcal{RP}_c : \ \ Level_c(p) = i \quad \Leftrightarrow \quad \exists o \in \mathcal{O} \ \langle o, p, o_{L_i} \rangle \in in(c).$$

5.1 Telos-dependent validity constraints

We now give a number of validity constraints, called Telos-dependent validity constraints, which together with our core validity constraints, presented in subsection 4.4, should be satisfied by any Telos contextualized information base. Indeed the following validity constraints support embedding of Telos into our contextualized framework, and are contextualized versions of the basic contraints of Telos [27].

Validity Constraint 5.1 Membership to the Attribute built-in object. If a is an attribute link in c from a reference path p_f to a reference path p_t, then there is an instance-of link in c from a to o_{Attr}.

$$\forall c \in \mathcal{CXT}, \ a \in \mathcal{O}, \ p_f, p_t \in \mathcal{RP} :$$

$$\langle a, p_f, p_t \rangle \in attr(c) \quad \Rightarrow \quad \exists o \in \mathcal{O} \ \langle o, a, o_{Attr} \rangle \in in(c). \diamond$$

Validity Constraint 5.2 Object kind uniqueness constraint. A reference path cannot be both individual and attribute within a context.

$$\forall c \in \mathcal{CXT} \ o'_1, o'_2 \in \mathcal{O}, \ p_1, p_2 \in \mathcal{RP} :$$

$$\langle o'_1, p_1, o_{Ind} \rangle \in in(c) \ \wedge \ \langle o'_2, p_2, o_{Attr} \rangle \in in(c) \quad \Rightarrow \quad p_1 \neq p_2. \diamond$$

Note that, it is possible for an object to be individual in one context and attribute in another.

Validity Constraint 5.3 Level uniqueness constraint. A reference path cannot have more than one instantiation levels within a context, that is, it can be classified within a context under only one of the classes `Li`, $i \geq 0$.

$$\forall c \in \mathcal{CXT}, \ p \in \mathcal{RP}, \ i_1, i_2 \in I\!N :$$

$$Level_c(p) = i_1 \ \wedge \ Level_c(p) = i_2 \quad \Rightarrow \quad i_1 = i_2. \diamond$$

Validity Constraint 5.4 Source and destination uniqueness constraints.

1. The source of a link in a context c is unique.

$$\forall c \in \mathcal{CXT}, o \in \mathcal{O}, p_1, p_2 \in \mathcal{RP} :$$

$$From_c(o) = p_1 \ \wedge \ From_c(o) = p_2 \quad \Rightarrow \quad p_1 = p_2.$$

2. The destination of a link in a context c is unique.

$$\forall c \in \mathcal{CXT}, o \in \mathcal{O}, p_1, p_2 \in \mathcal{RP} :$$

$$To_c(o) = p_1 \ \wedge \ To_c(o) = p_2 \quad \Rightarrow \quad p_1 = p_2. \diamond$$

Validity Constraint 5.5 Instance-of constraints.

1. Attribute instantiation constraint.

   Let a and a' be two attribute links in c from p_f to p_t, and p'_f to p'_t, respectively. If there is an instance-of link in c from a to a', then there is an instance-of link in c from p_f to p'_f, and an instance-of link in c from p_t to p'_t.

   $$\forall c \in \mathcal{CXT},\ o, a, a' \in \mathcal{O},\ p_f, p_t, p'_f, p'_t \in \mathcal{RP},\ \ \exists o', o'' \in \mathcal{O} :$$
   $$\langle o, a, a' \rangle \in in(c)\ \wedge\ \langle a, p_f, p_t \rangle \in attr(c)\ \wedge\ \langle a', p'_f, p'_t \rangle \in attr(c)\ \ \Rightarrow$$
   $$\langle o', p_f, p'_f \rangle \in in(c)\ \wedge\ \langle o'', p_t, p'_t \rangle \in in(c).$$

2. Instance-of level constraint.

   If there is an instance-of link in c from p_f to p_t, then the instantiation level of p_t in c equals the instantiation level of p_f in c plus one.

   $$\forall c \in \mathcal{CXT},\ o, \in \mathcal{O},\ p_f, p_t \in \mathcal{RP} :$$
   $$\langle o, p_f, p_t \rangle \in in(c)\ \ \Rightarrow\ \ Level_c(p_t) = Level_c(p_f) + 1.\ \diamond$$

Validity Constraint 5.6 ISA constraints.

1. Attribute generalization constraint.

   Let a and a' be two attribute links in c from p_f to p_t, and p'_f to p'_t, respectively. If there is an ISA link in c from a to a', then there is an ISA link in c from p_f to p'_f, and an ISA link in c from p_t to p'_t.

   $$\forall c \in \mathcal{CXT},\ o, a, a' \in \mathcal{O},\ p_f, p_t, p'_f, p'_t \in \mathcal{RP},\ \ \exists o', o'' \in \mathcal{O} :$$
   $$\langle o, a, a' \rangle \in isa(c)\ \wedge\ \langle a, p_f, p_t \rangle \in attr(c)\ \wedge\ \langle a', p'_f, p'_t \rangle \in attr(c)\ \ \Rightarrow$$
   $$\langle o', p_f, p'_f \rangle \in isa(c)\ \wedge\ \langle o'', p_t, p'_t \rangle \in isa(c).$$

2. ISA level constraint.

   If there is an ISA link in c from p_f to p_t, then the level of p_t in c equals the level of p_f in c.

   $$\forall c \in \mathcal{CXT},\ o, \in \mathcal{O},\ p_f, p_t \in \mathcal{RP} :$$
   $$\langle o, p_f, p_t \rangle \in isa(c)\ \ \Rightarrow\ \ Level_c(p_t) = Level_c(p_f).\ \diamond$$

6 Related works

Various forms of contextualization have appeared in the area of computer science. However, these forms are very diverse and serve different purposes. In this section, we compare our work with related approaches which we classify in six categories: general contextualization frameworks, semantic model clustering, nested associations, context in Artificial Intelligence, contextual ontologies for the Semantic Web, and context-aware computing. Moreover, we relate our contextual modeling structures to Description Logics.

6.1 General contextualization frameworks

In this subsection, we compare the present work with our own previous works [60], as well as with the work of Mylopoulos and Motschnig-Pitrik [43, 44]. These works attempt to introduce a general framework for contextualization in information bases.

In [43, 44, 60], a context is treated as a special object which is associated to a set of objects and a lexicon, i.e. a binding of names to these objects. These works support nesting of contexts, context overlapping, relative naming, and define operations on contexts, such as context union, intersection, and difference. In addition, [60] establishes properties of operations on contexts, such as commutativity, associativity, and distributivity. On the other hand, [43] considers issues, such as authorization and transaction execution.

The notion of context introduced in this work, still supports nesting of contexts, context overlapping, and relative naming, yet it advances with respect to [43, 44, 60] mainly along two lines:

- We distinguish between objects and contexts. Objects represent real world concepts, whereas contexts are collections of objects. Within each context, local names and semantics are assigned to objects, as well as references (which are also contexts) for describing objects in more detail. Thus, a real world concept (e.g. the geography of Greece) is represented by an object which can have different detailed descriptions (i.e. references) within different contexts (e.g. the 15th century and the 20th century). This is certainly a modeling capability not offered by the other approaches.
- We support the interaction of our contextualization mechanism with the traditional abstraction mechanisms. Works in [43, 44, 60] lack this interaction. In [44], the notion of context is introduced in the Telos data model, where each context is considered to be at the Token level, i.e., an atomic object, and contexts do not participate in classification or generalization hierarchies.

6.2 Semantic model clustering

In "real life" applications, it is often the case that semantic data models become large and complex, and thus difficult to understand. Several techniques cope with this problem by decomposing the global schema into smaller, more manageable partitions, called *entity clusters* [18, 58, 13, 62, 8, 12, 20, 58, 70, 47, 66, 53].

In [58], several kinds of clustering are defined, all of which are supported by our framework:

1. *Dominance grouping*: Here, a dominant object o is grouped together with its related, non-dominant objects into a cluster that represents the same real world entity as o, but at a different level of abstraction. In our framework, we support this kind of grouping by allowing the reference of an object to contain the object itself. For example, in Figure 12, the reference of the object o_8 (`Agents`) in context c_3, contains the object itself (under the name `Agent`).
2. *Abstraction grouping*: Here, objects participating in abstractions such as classification, generalization, and attribution are grouped in a cluster. In our framework, we support this kind of grouping by allowing objects related by instance-of, ISA, and attribute links to be grouped together with these links in a context.

3. *Relationship grouping*: Here, a relationship together with its participating entities are grouped into a cluster. In our framework, we support this kind of grouping, as relationships are represented by attributes, and a context may contain attributes and their source and destination objects.

In [13], an additional kind of clustering is defined, called *relationship abstraction*, which abstracts a number of relationships into a higher-level relationship. In our framework, we support this kind of clustering through the concept of attribute reference.

Our framework differs substantially, from all of the above approaches in the following:

- A global schema is not a requirement for modeling the real world. Rather, it is possible that information about an object can only be found scattered across contexts.
- We support relative naming and relative semantics w.r.t. a context. Within different contexts, information about the same object can even be conflicting. Thus, information is meaningful only within a context, and its validity outside of it cannot be assumed directly (unless explicitly declared).
- We distinguish between objects and contexts, with the advantages described in the previous subsection.
- We support the interaction of our contextualization mechanism with the traditional abstraction mechanisms.

Although in semantic model clustering the schema is decomposed, instances of both low-level objects and high-level objects (i.e., clusters) are globally defined. A solution to this problem is given in [13] by providing an algorithm to extract an instance graph for a higher-level object, i.e., to decompose the instances according to the cluster. In our framework, instance-of links are context-dependent and the user can explicitly declare the instances of high-level objects. In particular, in our framework, the instance graph of a high-level object corresponds to the instance-of links directed towards the object, along with the references of these links, and the instance-of links within these references, recursively. In [13], a higher-level object can be defined as a view derived by a query expression over a semantic model. We think that view support is an important issue and we intend to address it for our framework in future work.

6.3 Nested associations

Work in [35] deals with the problem of abstracting complex associations between objects of a conceptual model in order to make large data schemas more comprehensive. Towards this goal, the authors define an *enclosing class* as an abstraction which encapsulates a set of local classes. Additionally, they define an *enclosing association class* as an abstraction which associates a source enclosing class with a destination enclosing class, and encapsulates a set of local classes, as well as local associations.

Intuitively, our concepts of object reference and attribute reference cover the concepts of enclosing class and enclosing attribute class. However, in [35], the main emphasis is placed on nested associations, and issues such as relative naming and relative semantics, as well as the interaction between the proposed abstraction and the abstractions of classification and generalization are not considered.

In [20], a levelled entity-relationship model is proposed, where higher-level entities encapsulate lower-level entities, similarly to our concept of object reference. However, the authors do not support the notion of attribute reference. Moreover, naming, semantics, and instances of objects are globally defined. In [20], the authors argue that a relationship to a subentity from a higher-level entity breaks the encapsulation of the entity containing the subentity. To solve this encapsulation problem, they propose the notion of *aspect* that works as a window that makes a lower-level object to appear at a higher-level object. Though encapsulation is an important issue, we do not examine it in this work. We consider this as an authorization issue that can be handled on top of our general framework mechanism.

6.4 Context in Artificial Intelligence

Motivated by the observation that one can never represent an object in complete generality, McCarthy [38] introduced the notation $ist(c, p)$ (pronounced as “is true”) meaning that a formula p holds in the context c, where c is meant to capture all that is not explicit in p and that is required to make p a true statement. One of the first attempts at formalizing context under McCarthy’s supervision was presented by Guha [25]. Motivated largely by his work in the Cyc project, an attempt to build an extremely large knowledge base to support common-sense reasoning. Guha introduced a formal semantics for the formula $ist(c, p)$. The consequences are: (i) contexts are first class objects, (ii) propositions and contexts are always relative to another context, (iii) contexts can be nested in any depth, (iv) lifting axioms can be used to relate the truth of formulas in different contexts.

Giunchiglia and Serafini proposed MultiContext Systems (MCS) [22] as a proof-theoretic framework for contextual reasoning. This paper introduces the notion of bridge rule, namely a special kind of inference rule whose premises and conclusion hold in different contexts. Moreover, Ghidini and Giunchiglia proposed Local Models Semantics (LMS) [21] as a model-theoretic framework for contextual reasoning. For an overview and comparison of the work on formalizing context in Artificial Intelligence, see [4, 6, 51].

In this paper, we consider the notion of context in information bases, and its interaction with the traditional abstraction mechanisms. The contents of a context can be thought of as formulas that are declared to hold within a context, and our logical implications to be formulas derived based on our model theory, and inference rules. Obviously, our reasoning framework is limited in the sense that we consider only specific formulas (that we call *contextual atoms*), and only one kind of bridge rule (that we call *context refinement*). However, elaborating on specific aspects of context, we were able to impose a formal structure for contextualized information bases, and define a number of validity constraints that every contextualized information base should satisfy. Moreover, our inference rules allow the derivation of new information based on this formal structure and the interaction between the different abstraction mechanisms.

6.5 Contextualized ontologies for the Semantic Web

In this subsection, we review ontology languages for the Semantic Web supporting the notion of context.

In [14], the authors extend the web ontology schema language RDF(S) [69] with the notion of context, such that conceptual graphs [53] can be represented in the extended schema language. Though conceptual graphs support (nested) contexts, their notion of context is mostly to encapsulate a set of (object-attribute-object) triples as a value to an attribute of a higher level context. In our framework, a context encapsulates not only attributes but also ISA and instance-of relationships. In addition, in our framework, an attribute a can have a reference to a context, supporting attribute abstraction and hiding of details for the exact linkage between the source and destination of a. Moreover, we support the refinement relation between contexts.

In [5], the authors present a contextualized version of Description Logic [17] for application to the web ontology languages OWL [68] and DAML+OIL [67]. In their framework, contextual concepts are defined based on a set of classical and/or contextual concept constructors. Contextual concepts are given context-dependent semantics based on their definition. Though this extension is interesting, it does not support nesting of contexts, attribute abstraction, and context refinement.

C-OWL [7] is an extension of the web ontology language OWL [68] with a notion of context. A context in C-OWL is an OWL ontology whose local concepts are defined based on local and/or foreign concepts and roles. Foreign concepts and roles are defined in a different context. Contexts are connected via bridge rules, stating that two contextualized concepts are equivalent, disjoint, intersect, or one subsumes the other. This notion of context does not support several of the features of our model, e.g. nesting of contexts, attribute abstraction, and context refinement.

6.6 Context-aware computing

In [48], context-aware computing is defined as a software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time. A more general definition is given in [16, 15], where context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application. A system is context aware if it uses context to provide relevant information and/or services to the user.

In pervasive computing and context-aware services, explicit representation of context and contextual knowledge is considered critical to intelligent agents. In this framework, a context can be a distinguished collection of possible world features that has predictive worth to the agents. Once an agent knows that it is in a particular context, it immediately knows a great deal about the situation [63, 64]. In addition, a context can be a description of a situation (location, environmental attributes etc.) evaluated by an agent, or available to a service before and during execution [10, 11, 57, 56, 55].

The field of context-aware computing can benefit from our work as different types of situations can be modeled in our framework as different contexts, called *situation-schema contexts*, that contain schema information for the particular type of situation. The semantics of the situation-schema contexts can be indicated by *situation-schema objects* referring to these contexts. Moreover, a refinement hierarchy of situation-schema contexts can be defined, through ISA links between the corresponding situation-schema objects. In our framework, the current situation of relevant entities can be modeled by (possibly short-lived) contexts, called *current-situation contexts*, based on information acquired by the context acquisition module. Our contextual instantiation mechanism can be used to relate objects in the current-situation contexts to classes in the situation-schema contexts, through instance-of links. Obviously, a formal organization of contexts can help agents to discover, reason about, and communicate contextual information.

6.7 Relationship with Description Logic

Description Logics (DLs) [17] are knowledge representation languages tailored for expressing knowledge about concepts and concept hierarchies. In particular, any DL is a decidable subset of the function-free FOL using at most *three* variable names. In DL, a knowledge base, also referred as a DL theory, denoted by Σ, is formed by two components: the *intensional* one, called TBox, (denoted by T), and the *extensional* one, called ABox (denoted by A), i.e. $\Sigma = (T, A)$. The first is a general schema concerning the classes of individuals to be represented (concepts), their general properties and mutual relationships. The latter is a (partial) instantiation of this schema, containing assertions relating either individuals to classes, or individuals to each other. Specifically, the language used is composed by symbols denoting *concepts*, *individuals*, and *roles* (binary relations). Besides the above symbols, the alphabet includes a number of *constructors* that permit the formation of *concept expressions*. Description Logics allow inferences, such as the derivation of new subsumption relationships between concepts and concept memberships for individuals.

Though it seems that the contents of a context c can be easily represented through a DL knowledge base Σ indexed by c, that is $\langle c, \Sigma \rangle$, the correspondences are not straightforward due to several differences between the two frameworks. Below we attempt a mapping between the two frameworks and state their differences.

DL concepts, individuals, and roles appearing in $\langle c, \Sigma \rangle$ correspond in our framework to objects of c. In particular, a DL role appearing in $\langle c, \Sigma \rangle$ corresponds to an attribute link in c. An ABox statement $C(o)$ in $\langle c, \Sigma \rangle$ corresponds to an instance-of link in c from individual o to concept C. An ABox statement $R(s, o)$ in $\langle c, \Sigma \rangle$ corresponds to (i) an attribute link a in c from individual s to individual o, and (ii) an ISA link in c from attribute link a to role R. A TBox statement $C_1 \sqsubseteq C_2$ in $\langle c, \Sigma \rangle$ corresponds to an ISA link in c from concept C_1 to concept C_2. Note that in all these statements of $\langle c, \Sigma \rangle$, the associated objects are objects of c. However, in our framework, the source and destination objects of instance-of, ISA, and attribute links, contextualized within a context c, are in general objects reached through a specified object path originating from an object in c by following reference links. Thus, the source and destination objects of the links are not necessarily objects of c.

Additionally, as in our framework, instance-of, ISA, and attribute links are objects contextualized within a context, they may be associated with a nested context (reference). In particular, the reference of an instance-of link i contains the particular instance-of links from the objects of the source reference of i to the objects of the destination reference of i (instance-of abstraction). The reference of an attribute link a contains the particular attribute paths from the objects of the source reference of a to the objects of the destination reference of i (attribute abstraction). These features cannot be easily represented in DL.

However, we want to mention that indeed the contents of a context c can be easily represented through a DL knowledge base, indexed by c and extended with references (to allow nesting of contexts), assuming however that several important features of our model are eliminated. In particular, we should impose the constraint that link objects in a context c relate only objects of c, and that references of link objects are not allowed.

7 Conclusions

In this paper, we are concerned with a notion of context in the area of conceptual modeling. In our approach, a context is seen as a structured set of objects, in which each object is associated with a set of names and (possibly) a reference: the reference of the object is another context which "hides" information related to the object. Within a context, objects can be structured through the traditional abstraction mechanisms of classification, generalization, and attribution. One of the contributions of this paper is that we study how the contextualization mechanism interacts with the traditional abstraction mechanisms.

Adding contextualization to an information base provides several modeling capabilities, including:

- *Context-dependent semantics*: A given object may be represented and interpreted differently in different context-delimited parts of the information base, representing different points of view.
- *Modular representation*: At each level of abstraction, an overview of the available information can be presented in the form of objects that provide access to relevant detail.
- *Focused information access*: A context delimits the parts of an information base that are accessible in a given way. Thus, context can act as a focusing mechanism when searching for information.
- *Ability to handle inconsistent information*: Contradictory information can be represented in the same information base as long as it is treated in different contexts.
- *Encapsulation of local information sources*: Contexts can be used to encapsulate and model (possibly conflicting) local information sources in data integration.

Future work concerns the extension of Semantic Web ontology languages, such as RDF(S) [69], DAML+OIL [67], OWL [68], with our notion of context. Such an extension will enhance Semantic Web ontologies with all of the above mentioned features. We believe that these features are necessary for modeling and integrating a distributed, complex, and many times contradictory environment, such as the web. Additionally, we are concerned with the development of a general framework for querying contextualized information bases. This framework includes the definition of useful fundamental query operations on contexts such as selection, projection, and join, as well as composition operations such as union, intersection, and difference. We are also concerned with methodological issues such as criteria for context formation.

APPENDIX: Proofs of Propositions

In this Appendix, we prove the propositions appearing in the paper.

Proposition 4.1 Soundness and Completeness. Let CIB be a contextualized information base, and let α be a contextual atom. It holds: $CIB \models \alpha \Leftrightarrow CIB \vdash \alpha$.

Proof: Let $CIB = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, cnts\rangle$. Soundness of the inference rules can be easily verified. That is, if α is a contextual atom such that $CIB \vdash \alpha$, it holds $CIB \models \alpha$. Concerning the proof of completeness, we first construct a model I of CIB.

Starting from the CIB user declarations and Inference Rule 4.1, we apply the inference rules until we reach a fixpoint F. Such a fixpoint is reached in a finite number of steps. This is because recursive derivations of context refinement (due to Inference Rule 4.4(6)) always terminate (due to Validity Constraints 4.1, 4.5, and 4.6). Then, we construct a model I of CIB by following the steps below:

Step 1: We extend F with additional binary ISA relationships by applying the following rule:

$\forall p_1, p_2 \in \mathcal{RP}$:

$$Isa_c(p_1, p_2) \;\; \Leftarrow \;\; (\forall p \in \mathcal{RP} : \;\; In_c(p, p_1) \;\; \Rightarrow \;\; In_c(p, p_2)).$$

We denote the extended F by F''.

Step 2: Based on F'', we construct an interpretation $I = \langle \mathcal{O}, \mathcal{CXT}, \mathcal{N}, cnts^I, \precsim^I\rangle$ of CIB as follows:

$\forall o \in \mathcal{O},\; c, c' \in \mathcal{CXT},\; p_1, p_2 \in \mathcal{RP},\; n \in \mathcal{N}$:

1. If $Obj_c(o) \in F''$ then $o \in objs^I(c)$.
2. If $Name_c(o, n) \in F''$ then $n \in names^I(o, c)$.
3. If $(ref(o, c) = c') \in F''$ then $ref^I(o, c) = c'$.
4. If $defined(ref(o, c)) \in F''$ and $\nexists\, c'' \in \mathcal{CXT}$: $(ref(o, c) = c'') \in F''$ then $ref^I(o, c) = c_{new}$, where c_{new} is an unused context identifier.
5. If $Isa_c(o, p_1, p_2) \in F''$ then $\langle o, p_1, p_2\rangle \in isa^I(c)$.
6. If $Isa_c(p_1, p_2)$ and $\nexists\, o \in \mathcal{O}$: $Isa_c(o, p_1, p_2) \in F''$ then $\langle o_{new}, p_1, p_2\rangle \in isa^I(c)$, where o_{new} is an unused context identifier.
7. If $In_c(o, p_1, p_2) \in F''$ then $\langle o, p_1, p_2\rangle \in in^I(c)$.
8. If $In_c(p_1, p_2) \in F''$ and $\nexists\, o \in \mathcal{O}$: $In_c(o, p_1, p_2) \in F''$ then $\langle o_{new}, p_1, p_2\rangle \in in^I(c)$, where o_{new} is an unused context identifier.
9. If $Attr_c(o, p_1, p_2) \in F''$ then $\langle o, p_1, p_2\rangle \in attr^I(c)$.
10. If $(c \precsim c') \in F''$ then $c \precsim^I c'$.

Step 3: We extend I with additional $\precsim^I$ relationships by applying the following rule:

$\forall c, c' \in \mathcal{CXT}$:

$$c \precsim^I c' \quad \Leftarrow \quad \begin{array}{l} (objs^I(c') \;\subseteq\; objs^I(c) \;\; \wedge \\ attr^I(c') \;\subseteq\; attr^I(c) \;\; \wedge \\ in^I(c') \;\subseteq\; in^I(c) \;\; \wedge \\ isa^I(c') \;\subseteq\; isa^I(c) \;\; \wedge \\ (\forall o \in objs(c'),\; c_1 \in \mathcal{CXT},\; \exists c_2 \in \mathcal{CXT} : \\ \qquad names^I(o, c') \subseteq names^I(o, c) \;\wedge \\ \qquad ref^I(o, c') = c_1 \;\; \Rightarrow \;\; ref^I(o, c) = c_2 \;\wedge\; c_2 \precsim^I c_1). \end{array}$$

We denote the extended I, also by I. It is easy to see that the interpretation I satisfies all model constraints. Therefore, I is a model of CIB.

Let α be one of the contextual atoms: $Obj_c(o),\ Name_c(o, n),\ ref(o, c) = c',\ defined(ref(o, c)),$ $Isa_c(o, p_1, p_2),\ In_c(o, p_1, p_2)$. Assume that $CIB \models \alpha$. From the construction of model I, it follows that $\alpha \in F$. Thus, $CIB \vdash \alpha$.

Let now $\alpha = In_c(p_1, p_2)$, and assume that $CIB \models \alpha$. We will show that $CIB \vdash \alpha$. As I is a model of CIB, there is an object o such that $\langle o, p_1, p_2\rangle \in in^I(c)$. This implies that $In_c(o, p_1, p_2) \in F$ or $In_c(p_1, p_2) \in F$. Therefore, from Inference Rule 4.2(1), it follows that $In_c(p_1, p_2) \in F$. Thus, $CIB \vdash \alpha$.

Let now $\alpha = Isa_c(p_1, p_2)$, and assume that $CIB \models \alpha$. We will show that $CIB \vdash \alpha$. First, we will create a model of CIB as follows: We create two new objects o, o_1, and extend the contents of c such that $o, o_1 \in objs(c)$ and $\langle o_1, o, p_1\rangle \in in(c)$. Then, we apply the inference rules, starting from the extended CIB and Inference Rule 4.1, until we reach a fixpoint F'. From F', we construct an interpretation I' of CIB, following the same steps that we followed to construct I from F. It is easy to see that I' is also a model of CIB. As I' is a model of CIB and $\langle o_1, o, p_1\rangle \in in^{I'}(c)$, there should be an object o_2 such that $\langle o_2, o, p_2\rangle \in in^{I'}(c)$. This implies that $In_c(o, p_1) \in F$ and $In_c(o, p_2) \in F$. Due to Inference Rule 4.6, this implies that (i) $p_1 = p_2$, or (ii) $Isa_c(p_1, p_2) \in F$, or (iii) $\exists p'_1, ..., p'_n \in \mathcal{RP} :\ \ Isa_c(p_1, p'_1) \in F,\ Isa_c(p'_1, p'_2) \in F,\ ...,\ Isa_c(p'_{n-1}, p'_n) \in F,\ Isa_c(p'_n, p_2) \in F$, for $n \geq 1$. But then $Isa_c(p_1, p_2) \in F$, due to Inference Rule 4.3. Thus, $CIB \vdash \alpha$.

Let now $\alpha = (c_1 \precsim c_2)$, and assume that $CIB \models \alpha$. We will show that $CIB \vdash \alpha$. First, we will create a model of CIB as follows: We create a new object o, and extend the contents of c_1 such that $o \in objs(c_1)$. Then, we apply the inference rules until we reach a fixpoint F'. From F', we construct an interpretation I' of CIB, following the same steps that we followed to construct I from F. It is easy to see that I' is also a model of CIB. As I' is a model of CIB and $o \in objs^{I'}(c_1)$, it should hold that $o \in objs^{I'}(c_2)$. This implies that $Obj_{c_1}(o) \in F$ and $Obj_{c_2}(o) \in F$. Due to Inference Rule 4.4, this implies that (i) $c_1 = c_2$, or (ii) $(c_1 \precsim c_2) \in F$, or (iii) $\exists c'_1, ..., c'_n \in \mathcal{CXT} :\ \ (c_1 \precsim c'_1) \in F,\ (c'_1 \precsim c'_2) \in F,\ ...,\ (c'_{n-1} \precsim c'_n) \in F,\ (c'_n \precsim c_2) \in F$, for $n \geq 1$. But then $(c_1 \precsim c_2) \in F$, due to Inference Rule 4.5. Thus, $CIB \vdash \alpha$.

Therefore, we have showed that if α is a contextual atom such that $CIB \models \alpha$, it holds $CIB \vdash \alpha$. $\Box$

Proposition 4.2 The relation $\sim$ on the set of contexts $\mathcal{CXT}$ is an equivalence relation, i.e., reflexive, transitive, and symmetric.

Proof: From Definition 4.14, it is easy to prove recursively that $\sim$ is reflexive and symmetric.

We will now prove that $\sim$ is also transitive, that is:

$$\forall c_1, c_2, c_3 \in \mathcal{CXT} :\ \ (c_1 \sim c_2)\ \wedge\ (c_2 \sim c_3) \Rightarrow c_1 \sim c_3$$

We distinguish the following cases:

1. $c_1 = c_2 = c_3 = NIL$.

   Then, from Definition 4.14, it easily follows that $c_1 \sim c_3$.

2. $c_1 \neq NIL$ and $c_2 \neq NIL$ and $c_3 \neq NIL$.

   (a) $c_1 = c_2 = c_3$ or $c_1 = c_2 \neq c_3$ or $c_1 \neq c_2 = c_3$ or $c_1 = c_3 \neq c_2$.

   Then, from Definition 4.14, it easily follows that $c_1 \sim c_3$.

   (b) $c_1 \neq c_2,\ c_2 \neq c_3$, and $c_1 \neq c_3$.

   From Definition 4.14 we have

$$c_1 \sim c_2 \Rightarrow \left\{ \begin{array}{rcl} \exists \pi \in Iso(GR(c_1), GR(c_2)) & : & \pi(c_1) = c_2 \wedge \\ \forall v_1 \in V_1 : & & \\ attr(v_1) & = & attr(\pi(v_1)) \wedge \\ in(v_1) & = & in(\pi(v_1)) \wedge \\ isa(v_1) & = & isa(\pi(v_1)) \wedge \\ \forall o \in objs(v_1) : \ names(o, v_1) & = & names(o, \pi(v_1)) \end{array} \right. \tag{1}$$

$$c_2 \sim c_3 \Rightarrow \left\{ \begin{array}{rcl} \exists \phi \in Iso(GR(c_2), GR(c_3)) & : & \phi(c_2) = c_3 \wedge \\ \forall v_2 \in V_2 : & & \\ attr(v_2) & = & attr(\phi(v_2)) \wedge \\ in(v_2) & = & in(\phi(v_2)) \wedge \\ isa(v_2) & = & isa(\phi(v_2)) \wedge \\ \forall o \in objs(v_2) : \ names(o, v_2) & = & names(o, \phi(v_2)) \end{array} \right. \tag{2}$$

$$\begin{array}{r} \text{Eq. } (1) \\ \text{Eq. } (2) \\ \text{Lemma } 4.1 \end{array} \Rightarrow \left\{ \begin{array}{rcl} \exists \rho = \pi \circ \phi \in Iso(GR(c_1), GR(c_3)) & : & \rho(c_1) = c_3 \wedge \\ \forall v_1 \in V_1 : & & \\ attr(v_1) & = & attr(\rho(v_1)) \wedge \\ in(v_1) & = & in(\rho(v_1)) \wedge \\ isa(v_1) & = & isa(\rho(v_1)) \wedge \\ \forall o \in objs(v_1) : \ names(o, v_1) & = & names(o, \rho(v_1)) \end{array} \right. \tag{3}$$

   From Definition 4.14 and Equation (3) we have: $c_1 \sim c_3$. $\square$

Proposition 4.3: The refinement relation on the set of contexts $\mathcal{CXT}$ is a partial order up to the equivalence relation $\sim$, i.e., it is reflexive, transitive, and antisymmetric.

Proof:

From Definition 4.15, it is easy to prove recursively that $\precsim$ is reflexive and transitive.

We will prove that $\precsim$ is antisymmetric, that is:

$$\forall c, c' \in \mathcal{CXT} : \ (c \precsim c') \ \wedge \ (c' \precsim c) \Rightarrow c \sim c')$$

We distinguish the following cases:

1. $c = c' = NIL$ then $c \sim c'$.

2. $(c' \neq NIL)$ and $(c = NIL)$

   Then, from our assumptions, we have:

   $NIL \neq c \precsim c' = NIL,$ which is true according to Definition 4.15, and

   $NIL = c' \precsim c \neq NIL,$ which is false according to Definition 4.15.

   Thus, this case is impossible.

3. $(c \neq NIL)$ and $(c' = NIL)$

   It is proved similar to the previous case, that this case is impossible.

4.

$$
\begin{array}{l}
(c \neq NIL \;\wedge\; c' \neq NIL) \\
\;\wedge \\
(\;\nexists \pi \in Iso(GR(c), GR(c')) :\;\; \pi(c) = c' \wedge \\
\qquad\qquad \forall v \in V : \\
\qquad\qquad \begin{array}{lcll}
attr(v) & = & attr(\pi(v)) & \wedge \\
in(v) & = & in(\pi(v)) & \wedge \\
isa(v) & = & isa(\pi(v)) & \wedge
\end{array} \\
\qquad\qquad (\forall o \in objs(v) :\;\; names(o,v) = names(o,\pi(v))))
\end{array}
$$

   We will show that there exists an isomorphism like this. From our assumption we have:

$$
c \precsim c' \Rightarrow \left\{
\begin{array}{rcl}
objs(c') & \subseteq & objs(c) \wedge \\
attr(c') & \subseteq & attr(c) \wedge \\
in(c') & \subseteq & in(c) \wedge \\
isa(c') & \subseteq & isa(c) \wedge \\
(\forall o \in objs(c') & : & \\
names(o,c') & \subseteq & names(o,c) \wedge \\
ref(o,c) & \precsim & ref(o,c'))
\end{array}
\right. \tag{4}
$$

$$
c' \precsim c \Rightarrow \left\{
\begin{array}{rcl}
objs(c) & \subseteq & objs(c') \wedge \\
attr(c) & \subseteq & attr(c') \wedge \\
in(c) & \subseteq & in(c') \wedge \\
isa(c) & \subseteq & isa(c') \wedge \\
(\forall o \in objs(c) & : & \\
names(o,c) & \subseteq & names(o,c') \wedge \\
ref(o,c') & \precsim & ref(o,c))
\end{array}
\right. \tag{5}
$$

$$\begin{array}{l} \text{Eq. } (4) \\ \text{Eq. } (5) \end{array} \Rightarrow \left\{ \begin{array}{rcl} objs(c) & = & objs(c') \wedge \\ attr(c) & = & attr(c') \wedge \\ in(c) & = & in(c') \wedge \\ isa(c) & = & isa(c') \wedge \\ (\forall o \in objs(c) & : & \\ names(o,c) & = & names(o,c') \wedge \\ ref(o,c) & \precsim & ref(o,c') \wedge \\ ref(o,c') & \precsim & ref(o,c)) \end{array} \right. \tag{6}$$

This applied recursively to all subcontexts of c and c'. Let $GR(c) = \langle V, E, L \rangle$ and $GR(c') = \langle V', E', L' \rangle$. Consider the function $\phi : V \rightarrow V'$ which is defined recursively as follows:

$$\begin{array}{rcl} \phi(c) & = & c' \\ \phi(ref(o,v)) & = & ref(o,v'), \quad \forall v \in V, v' \in V' : \ \phi(v) = v' \ \wedge \ o \in objs(v) \end{array}$$

It is easy to see that ϕ is a bijection, and that from Equation (6) it follows:

$$\begin{array}{l} \forall v \in V : \\ \begin{array}{rcl} objs(v) & = & objs(\phi(v)) \wedge \\ attr(v) & = & attr(\phi(v)) \wedge \\ in(v) & = & in(\phi(v)) \wedge \\ isa(v) & = & isa(\phi(v)) \wedge \\ (\forall o \in objs(v) & : & \\ names(o,v) & = & names(o, \phi(v))) \end{array} \end{array}$$

Thus, $c \sim c'$. $\square$

Acknowledgements: The authors would like to thank Aran Lunzer and the anonymous reviewers for their valuable comments.

References

[1] S. Abiteboul and A. Bonner. Objects and Views. In *Proc. of the ACM-SIGMOD Conference*, pages 238–247, Feb. 1991.

[2] S. Abiteboul and J. Van Den Bussche. Deep Equality Revisited. In T. Ling, A. Mendelzon, and L. Vieille, editors, *Proceedings of the Fourth International Conference on Deductive and Object-Oriented Databases (DOOD'95)*, volume 1013 of *Lecture Notes in Computer Science*, pages 213–228, Singapore, Dec. 1995. Springer.

[3] F. Bancilhon and N. Spyratos. Update Semantics of Relational Views. *ACM Transactions on Database Systems*, 6(4):557–575, Dec. 1981.

[4] M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Journal of Theoretical and Experimental Artificial Intelligence*, 12(3):279–305, 2000.

[5] D. Benslimane, C. Vangenot, C. Roussey, and A. Arara. Multi-representation in ontologies. In *Proc. of the 7th East-European Conference on Advances in Databases and Informations Systems (ADBIS'2003)*, September 2003.

[6] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri. Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics*, 35(3):455–484, 2003.

[7] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In K. Sekara and J. Mylopoulos, editors, *Proc. of the 2nd International Semantic Web Conference*, pages 164–179. Springer Verlag, October 2003.

[8] L. Campbell, T. Halping, and H. Proper. Conceptual Schemas with Abstractions: Making Flat Conceptual Schemas More Comprehensible. *Data and Knowledge Engineering*, 20(1):39–85, June 1996.

[9] O. Caron, B. Carre, and L. Debrauwer. Contextualization of OODB Schemas in CROME. In *Proc. of the 11th International Conference Database and Expert Systems Applications (DEXA 2000)*, pages 135–149, 2000.

[10] H. Chen, T. Finin, and A. Joshi. Semantic Web in a Pervasive Context-Aware Architecture. In *Artificial Intelligence in Mobile System 2003 (AIMS 2003), in conjuction with Ubicomp 2003*, October 2003.

[11] H. Chen, T. Finin, and A. Joshi. Using OWL in a Pervasive Computing Broker. In *Proc. of the AAMAS-2003 Workshop on Ontologies in Agent Systems*, July 2003.

[12] P. Creasy and H. Proper. A Generic Model for 3-Dimensional Conceptual Modelling. *Data and Knowledge Engineering*, 20(2):119–161, Oct. 1996.

[13] B. Czejdo and D. Embley. View Specification and Manipulation for a Semantic Data Model. *Information Systems*, 16(6):585–612, 1991.

[14] A. Delteil and C. Faron-Zucker. Extension of RDF(S) with Contextual and Definitional Knowledge. In *Proc. of WebNet 2001 - World Conference on the WWW and Internet (WebNet 2001)*, pages 273–278, October 2001.

[15] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[16] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, GVU Center, Georgia Institute of Technology, 1999.

[17] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In G. Brewka, editor, *Principles of Knowledge Representation*, chapter 1, pages 191–236. CSLI Publications, 1996.

[18] P. Feldman and D. Miller. Entity Model Clustering: Structuring a Data Model by Abstraction. *The Computer Journal*, 29(4):348–360, Apr. 1986.

[19] S. Fortin. The Graph Isomorphism Problem. Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada, 1996.

[20] M. Gandhi, E. Robertson, and D. Gucht. Levelled Entity Relationship Model. In *Proceedings of 13th International Conference on the Entity-Relationship Approach - ER'94*, pages 420–436, Manchester, U.K., Dec. 1994. Springer-Verlag.

[21] F. Giunchiglia and C. Ghidini. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[22] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics or: how we can do without modal logics. *Artificial intelligence*, 65(1):29–70, 1994.

[23] G. Gottlob, P. Paolini, and R. Zicari. Properties and update semantics of consistent views. *ACM Transactions on Database Systems*, 13(4):486–524, Dec. 1988.

[24] G. Gottlob, M. Schrefl, and B. Röck. Extending Object - Oriented Systems with Roles. *ACM Transactions on Information Systems*, 14(3):268–296, July 1996.

[25] R. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford University, 1991. Also published as Technical Report STAN-CS-91-1399-Thesis, and MCC Technical Report Number ACT-CYC-423-91.

[26] R. Hull and R. King. Semantic Database Modeling. *ACM Computing Surveys*, 19(3):202–260, Sept. 1987.

[27] M. Jarke, S. Eherer, R. Gallersdorfer, M. Jeusfeld, and M. Staudt. ConceptBase - A Deductive Object Base Manager. *Journal of Intelligent Information Systems (JIIS)*, 4(2):167–192, 1995.

[28] F. Jouanot, N. Cullot, and K. Yetongnon. Context Comparison for Object Fusion. In *Proc. of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003)*, pages 536–551, June 2003.

[29] V. Kashyap and A. Sheth. Semantic and Schematic Similarities between Database Objects: A Context-Based Approach. *The VLDB Journal*, 5(4):276–304, Dec. 1996.

[30] R. Katz. Towards a Unified Framework for Version Modeling in Engineering Databases. *ACM Computing Surveys*, 22(4):375–408, Dec. 1990.

[31] M. Kifer, W. Kim, and Y. Sagiv. Querying Object-Oriented Databases. In M. Stonebraker, editor, *Proc. of the ACM-SIGMOD Conference*, pages 393–402, San Diego, Clifornia, June 1992.

[32] J. Köbler, U. Schöning, and J. Torán. Graph Isomorphism is Low for PP. *Journal of Computational Complexity*, 2:301–330, 1992.

[33] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser Boston Inc., Boston, MA,, 1993.

[34] 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.

[35] B. Kristensen. Complex Associations: Abstractions in Object-Oriented Modeling. In *Proc. of the Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, volume 29:10 of *ACM Sigplan Notices*, pages 272–283, 1994.

[36] S. Lawrence. Context in Web Search. *IEEE Data Engineering Bulletin*, 23(3):25–32, 2000.

[37] S. Matwin and M. Kubat. The role of Context in Concept Learning. In *Proceedings of the ICML-96 Workshop on Learning in Context-Sensitive Domains*, pages 1–5, Bari, Italy, July 1996.

[38] J. McCarthy. Notes on Formalizing Context. In *Proc. IJCAI-93*, pages 555–560, Chambery, France, 1993.

[39] R. Michalski. How to Learn Impressive Concepts: A Method Employing a Two-Tiered Knowledge Representation for Learning. In *Proceedings of the 4th International Workshop in Machine Learning*, pages 50–58, Irvine, CA, 1987.

[40] G. L. Miller. Graph Isomorphism, General Remarks. *Journal of Computer and System Sciences*, 18(2):128–142, Apr. 1979.

[41] R. Motschnig-Pitrik. Contexts and Views in Object-Oriented Languages. In *Proc. of the 2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99)*, pages 256–269, September 1999.

[42] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing Knowledge about Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, Oct. 1990.

[43] J. Mylopoulos and R. Motschnig-Pitrik. Partitioning Information Bases with Contexts. In *Proc. of CoopIS'95*, pages 44–55, Vienna, Austria, 1995.

[44] J. Mylopoulos and R. Motschnig-Pitrik. Semantics, Features, and Applications of the Viewpoint Abstraction. In *Proc. of CAiSE'96*, pages 514–539, Heraklion, Greece, May 1996.

[45] A. Ouksel and C. Naiman. Coordinating Context Building in Heterogeneous Information Systems. *Journal of Intelligent Information Systems*, 3(2):151–183, 1994.

[46] J. Richardson and P. Schwarz. Aspects: Extending Objects to Support Multiple, Independent Roles. In *Proc. of the ACM-SIGMOD Conference*, pages 298–307, Denver, Colorado, May 1991.

[47] U. Schiel, A. Furtado, E. Neuhold, and M. Casanova. Towards Multi-Level and Modular Conceptual Schema Specifications. *Information Systems*, 9(1):43–57, 1984.

[48] B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.

[49] M. Scholl, C. Laasch, and M. Tresch. Updatable Views in Object-Oriented Databases. In *Proc. of DOOD'91*, pages 189–207, Munich, Dec. 1991.

[50] E. Sciore. Object Specialization. *ACM Transactions on Information Systems*, 7(2):103–122, Apr. 1989.

[51] L. Serafini and P. Bouquet. Comparing formal theories of context in AI. *Artificial Intelligence*, 155(1-2):41–67, 2004.

[52] J. Shilling and P. Sweeney. Three Steps to Views: Extending the Object-Oriented Paradigm. In *Proc. of the Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 353–361, Oct. 1989.

[53] J. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, 1984.

[54] V. Storey. Understanding Semantic Relationships. *VLDB Journal*, 2(4):455–488, Oct. 1993.

[55] T. Strang and C. Linnhoff-Popien. Service Interoperability on Context Level in Ubiquitous Computing Environments. In *Proc. of the International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR 2003)*, January 2003.

[56] T. Strang, C. Linnhoff-Popien, and K. Frank. Applications of a Context Ontology Language. In *Proc. of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2003)*, October 2003.

[57] T. Strang, C. Linnhoff-Popien, and K. Frank. CoOL:A Context Ontology Language to enable Contextual Interoperability. In *Proc. of 4th IFIP International Conferece on Distrinuted Applications and Interoperable Systems (DAIS 2003)*, November 2003.

[58] T. Teorey, G. Wei, D. Bolton, and J. Koenig. ER Model Clustering as an Aid for User Communication and Documentation in Database Design. *Communications of the ACM*, 32(8):975–987, Aug. 1989.

[59] M. Theodorakis. *Contextualization: An Abstraction Mechanism for Information Modeling*. PhD thesis, Department of Computer Science, University of Crete, June 2001.

[60] M. Theodorakis, A. Analyti, P. Constantopoulos, and N. Spyratos. A Theory of Context in Information Bases. *Information Systems*, 27(3):151–191, 2002.

[61] M. Theodorakis and P. Constantopoulos. Context-Based Naming in Information Bases. *International Journal of Cooperative Information Systems*, 6(3 & 4):269–292, 1997.

[62] O. Troyer and R. Janssen. On Modularity for Conceptual Data Models and the Consequences for Subtyping, Inheritance & Overriding. In *Proc. of the Ninth International Conference on Data Engineering*, pages 678–685, Vienna, Austria, Apr. 1993. IEEE Computer Society.

[63] R. Turner, E. Turner, T. Wagner, T. Wheeler, and N. Ogle. Using Explicit, A Priori Contextual Knowledge in an Intelligent Web Search Agent. In *Proc. of the 3rd International and Interdisciplinary Conference on Modeling and Using Context, CONTEXT'01*, pages 343–352, July 2001.

[64] R. M. Turner. A Model of Explicit Context Representation and Use for Intelligent Agents. In *Proc. of the 2nd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT'99)*, pages 375–388, September 1999.

[65] C. Vangenot, C. Parent, and S. Spaccapietra. Modeling and manipulating multiple representations of spatial data. In *Proc. of the International Symposium on Spatial Data Handling (SDH 2002)*, July 2002.

[66] D. Vermeir. Semantic Hierarchies and Abstractions in Conceptual Schemata. *Information Systems*, 8(2):117–124, 1983.

[67] W3C. DAML+OIL Reference Description. In *Technical Report, WWW Consortium (W3C), http://www.w3.org/TR/daml+oil-reference*, December 2001.

[68] W3C. OWL web ontology language overview. In *Techical Report, WWW Consortium (W3C), http://www.w3.org/TR/owl-features/*, August 2003.

[69] W3C. RDF Vocabulary Description Language 1.0: RDF Schema. In *Technical Report, WWW Consortium (W3C), http://www.w3.org/TR/rdf-schema/*, October 2003.

[70] H. Weber. Modularity in Data Base System Design: A Software Engineering View of Data Base Systems. *VLDB Surveys*, pages 65–91, 1978.