Ontology and Context

Isabel Cafezeiro
Departamento de Ciência da Computação
Universidade Federal Fluminense
Niterói - RJ, Brazil
isabel@dcc.ic.uff.br

Edward Hermann Haeusler,
Alexandre Rademaker
Departamento de Informática
Pontifícia Universidade Católica
Rio de Janeiro, Brazil
hermann,rademaker@inf.puc-rio.br

Abstract

This paper considers a formal framework to contextualize ontologies and presents a formal algebra to manipulate these entities providing several ways of composing ontologies, contexts or both. The algebra gives the flexibility that is required to model applications where the meaning of an entity depends on environment constraints or where dynamic changes in the environment must be considered.

Ontologies are used in computer science to describe real world things by hierarchically organizing concepts and enriching this hierarchy with relationships among concepts. A real word entity to be represented is always related to a context: a semantically consistent body of information in which the entity makes sense. The need of contexts is growing with the adoption of new paradigms of computation where information concerning either physical or computational environment is required, as in Computer-Aware Mobile Applications, where the environment suffer dynamic re-configurations [8].

In this paper we propose an algebra for manipulate *Contextualized Ontologies*. We adopt an homogeneous description of entities and contexts and define maps that consistently link entities and contexts. This framework gives the required flexibility, making possible the combination of entities or contexts in several ways, the changing and inheritance of context by an entity, and other useful operations. We consider a few basic formal concepts what makes it accessible even for those that are not familiar with formal methods. The formal approach for the proposed algebra not only gives a rigorous definition of *Contextualized Ontologies*, but also gives insights to sound implementation of environments. In addition, it is abstract enough to make possible the replacement of ontologies by other kind of technique for representing knowledge allowing the use of the same algebra in a large class of domains.

This paper is organized as follows: Section 1 gives an overview of the approach. Section 2 presents the complete algebra. Section 3 presents the formal basis and justifies the adopted formal method. Section 4 concludes the paper.

1 Contextualized Entities

Entities are described by three parts: the entity itself, a context, and a link between the entity and its context. As both entity and context are ontologies, an entity can be the context of other entity. The context, gives general information about the entity or about the environment wherein the entity operates. Any context is linked to a (meta)context. The link between the entity and its context, plays the role of ensuring that the entity and its context are coherent, that is, the context respects (preserves) information concerning the nature of the entity. For example, suppose that an entity E has parts e_1, e_2 related by f. Then a link F from E to its context C must respects the parts and internal relationships of E. Thus, $F : E \rightarrow C$ is such that $F(f(e_1, e_2)) = F(f)[F(e_1), F(e_2)]$.

In order to avoid violating internal constitution of entities, few constraints are stated about links: *(i)* any entity must have an identity link, that maps the entity to itself, and thus the entity may be viewed as a (non-informative) context of itself; *(ii)* an entity is called *domain* of a link, while a context is called *codomain* of a link; *(iii)* links can be composed in an associative way if the codomain of the first is the domain of the second. A triple (entity, link, context), also represented by $e \rightarrow c$, will be named *contextualized entity*. If the entity, or context, is represented by ontologies, we use the name *contextualized ontology*. We will use the symbol "$\circ$" to denote composition of contextualized entities.

In the sequel we present modular constructs that can be applied to contextualized entities, in order to coherently combine entities, contexts or both. We divide the operations in three classes: Entity Integration, Context Integration and Combined Integration.

2 The Algebra of Contextualized Entities

2.1 Entity Integration

Operations in this class have the purpose of integrating entities that share the same context. As entities are coherent with respect to their context, an operation of this class is guided by the context and results a new entity that is also attached to that context. Figure 1, part (a) shows entities E_1 and E_2 that share the context C. The right diagram of part (a) shows the produced entity, named E.

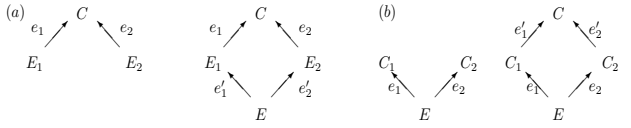

Figure 1. (a) **Integration of entities sharing the same context.** (b) **Integration of contexts of a same entity.**

The original entities play the role of context to the produced entity. By transitivity, the original context is also a context for the produced entity.

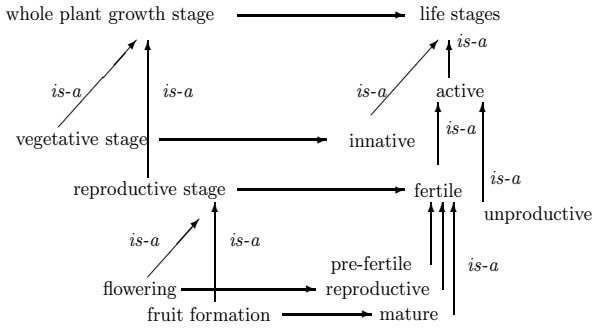

Figure 2. Whole Plant Growth Stage contextualized by Life Stages.

In order to correctly define E, we consider that right diagram of part (a) of Figure 1, has the following two properties: (i) The diagram is commutative, that is, from each component of E, the same component in C is reached by following the E_1 path or the E_2 path in the diagram. This property ensures the coherence of E_1, E_2 and E with respect of the context. (ii) The bottom entity E is the more complete entity that makes the diagram commute. By *more complete* we mean that all components of E_1 and E_2 that are linked to the same element in C have a corresponding in E, and nothing more than this is present in E. There is a unique entity constructed in this way that makes the diagram commutes (see Section 3). This unique entity is the semantic intersection of E_1 and E_2 with respect to the context. Thus, if E_1 and E_2 give different approaches about a subject C, then E express their agreement with respect to C.

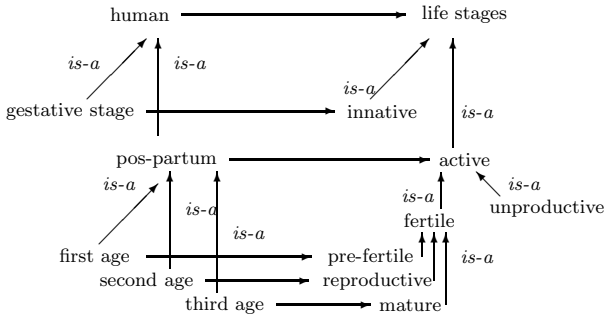

Figure 3. Human contextualized by Life Stages.

Example: Figures 2, 3 and 4 show contextualized entities: at left, the entity, at right, the context. Figure 4 shows the semantic intersection of contextualized entities of Figures 2 (about plant [7]) and 3 (about human being). Both are contextualized by an ontology that describes life stages. The result of the integration (Figure 4) embodies the semantic intersection of "plant" and "human". Both plant and human ontologies could also be viewed as context to the resulting entity. As stated by (i), components of the entity in Figure 4 correspond to those of "plant" and "human" that are linked to the same component of "life stage". As stated by (ii), *all* the components that satisfies (i) are present in the entity of Figure 4. In this way, hierarchy is preserved by the resulting links. Note also that it is the net formed by the links that gives the meaning of each entity, as it establishes the general context of entities.

Definition 1 (Entity Integration) *Given two contextualized entities sharing the same context $e_1 : E_1 \to C$ and $e_2 : E_2 \to C$, the integration of E_1 and E_2 with respect to C is the contextualized entity $E \to C$, such that, (i) There exists $e'_1 : E \to E_1$ and $e'_2 : E \to E_2$ such that $e_1 \circ e'_1 = e_2 \circ e'_2$, and, (ii) For any other other entity E'', with links $e''_1 : E'' \to E_1$ and $e''_2 : E'' \to E_2$ there exists a unique link $! : E'' \to E$ with $e'_1 \circ ! = e''_1$ and $e'_2 \circ ! = e''_2$*

The existence of links $e'_1 : E \to E_1$ and $e'_2 : E \to E_2$ ensures the semantic relationship of all components of E with those of E_1 and E_2. The commutative condition is (i). Condition *(ii)* expresses that E embody the complete intersection, and not part of it: if there exists another entity (E'') that could also be put in the place of E, then it could be semantically mapped in E.

Following, we present an algorithm that implements the operation. The first loop constructs the set of concepts C_E of the resulting ontology E, respecting the partial order (H^C) of E_1, E_2 and C. The second loop constructs the set

of relations R_E, respecting the relationships rel of E_1, E_2 and C. For concepts, the algorithm considers concepts x_1 in C_{E_1} that are linked to the same concept in C_C as a concept x_2 in C_{E_2}, that is: $f_{e_1}(x_1) = f_{e_2}(x_2)$. The image of $f_{e_1}(x_1)$ (or $f_{e_2}(x_2)$) will compose the set of concepts C_E of the new entity E. The components $f_{e'_1}$ and $f_{e'_2}$ of e'_1 and e'_2 are defined for each concept added to C_E linking the added concept to the inverse image of it by f_{e_1} in C_{E_1}, or by f_{e_2} in C_{E_2}. For relations, the algorithm acts in a similar way. The algorithm returns the composite $e_1 \circ e'_1$ (could be $e_2 \circ e'_2$), which is the link $E \rightarrow C$.

Algorithm. (Entity Integration)
Input: $e_1 : E_1 \rightarrow C$ and $e_2 : E_2 \rightarrow C$ Output: $E \rightarrow C$
Notation: x_i are variables for concepts of entities and y_i are variables for relations of entities. (C_E, R_E, H^C_E, rel_E) identify the components of an entity E. f_e is component f of a link e and g_e is component g of a link e. The symbol $\mapsto$ denotes the association by a function of the element at the left to the element at the right of the symbol $\mapsto$.
Initial conditions: C_E, R_E are empty sets and $f_{e'_1}, f_{e'_2}, g_{e'_1}, g_{e'_2}$ are empty functions.
For all $x_1 \in C_{E_1}$
If there is $x_2 \in C_{E_2}$ with $f_{e_1}(x_1) = f_{e_2}(x_2)$
$C_E := C_E \cup f_{e_1}(x_1)$
$f_{e'_1} := f_{e'_1} \cup (f_{e_1}(x_1) \in C_E) \mapsto x_1$
$f_{e'_2} := f_{e'_2} \cup (f_{e_2}(x_2) \in C_E) \mapsto x_2$
For all $y_1 \in R_{E_1}$
If there is $y_2 \in R_{E_2}$ with $g_{e_1}(y_1) = g_{e_2}(y_2)$
$R_E := R_E \cup g_{e_1}(y_1)$
$g_{e'_1} := g_{e'_1} \cup (g_{e_1}(y_1) \in C_E) \mapsto y_1$
$g_{e'_2} := g_{e'_2} \cup (g_{e_2}(y_2) \in C_E) \mapsto y_2$
return $(f_{e_1}, g_{e_1}) \circ (f_{e'_1}, g_{e'_1})$

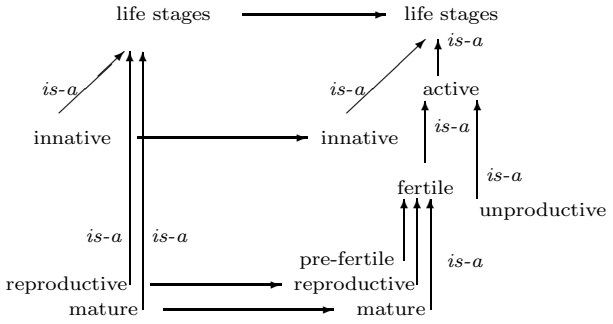

Figure 4. The semantic intersection between contextualized entities of Figures 2 and 3.

2.2 Context Integration

In Section 2.1 we showed an operation that results in an entity with more than one context. This situation happens not only as a consequence of that operation, but also in modeling many real world aspects, where a single entity can be viewed in different ways. This motivates the definition of a class of operations that act in the contexts of a single entity. A new context is produced as a result of combining and integrating given contexts. The resulting context must be coherent with respect to the corresponding entity.

Part (b) of Figure 1 shows entity E with two contexts C_1 and C_2, meaning that all components of that entity can be understood both as concepts of C_1 or as concepts of C_2. The right diagram of part (b) of Figure 1 shows the produced context C. All components of the original contexts will be represent in the new context, as the resulting links have the original contexts as domain. Thus this operation is a summation (amalgamation) of contexts. The commutativity of the diagram ensures that it is a coherent sum with respect to the entity: a component of the entity is mapped into (different) concepts in C_1 and C_2, and then, mapped to the same concept in C. The top context C is the less informative context that makes the diagram commute. By *less informative* we mean that all elements of C_1 and C_2 are represent in C, and nothing more than this. It can be proved that there is a unique context constructed in this way that makes the diagram commute.

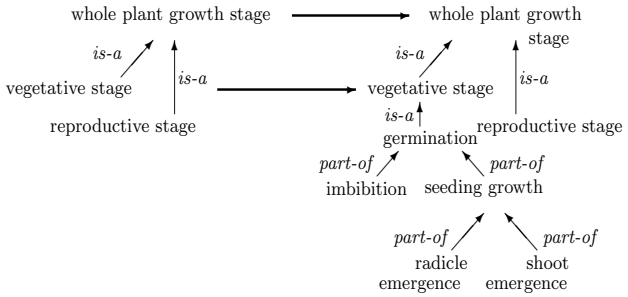

Figure 5. Part of *Whole Plant Growth Stage* ontology.

Example: A plant growth stage ontology is composed by two separate parts: vegetative stage (Figure 5) and reproductive stage (Figure 6). These parts can be developed in separate and glued later to form the complete plant growth stage. In the glue process part of the ontology (the glue points) must be contextualized by the ontologies containing the new parts. The integration of these contexts results the whole plant growth stage ontology.

Definition 2 (Context Integration) *Given two contextualizations of the same entity $e_1 : E \rightarrow C_1$ and $e_2 : E \rightarrow C_2$, the context integration of C_1 and C_2 with respect to E is the contextualized entity $E \rightarrow C$, such that, (i) There exists $e'_1 : C_1 \rightarrow C$ and $e'_2 : C_2 \rightarrow C$ such that $e'_1 \circ e_1 = e'_2 \circ e_2$, and, (ii) For any other other context C'', with maps $e''_1 : C_1 \rightarrow C''$ and $e''_2 : C_2 \rightarrow C''$ there exists a unique map $! : C \rightarrow C''$ with $! \circ e'_1 = e''_1$ and $! \circ e'_2 = e''_2$.*

The existence of maps $e'_1 : C_1 \rightarrow C$ and $e'_2 : C_2 \rightarrow C$ ensures that all components of contexts C_1 and C_2 are semantically mapped to the new context C, and thus, C is a kind of semantic union of C_1 and C_2. The commutative property is (i). Condition *(ii)* expresses that C is the less

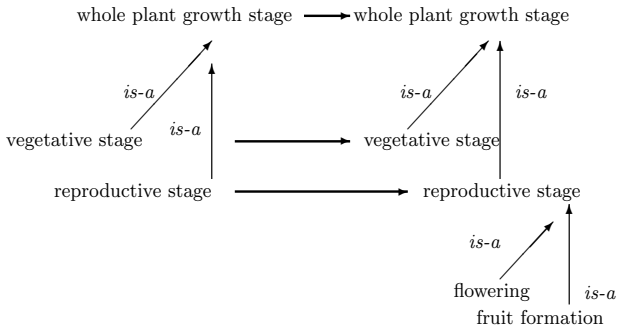

Figure 6. Another part of *Whole Plant Growth Stage* **ontology.**

informative context that can represent the union of C_1 or C_2.

Algorithm. (Context Integration)
Input: $e_1 : E \to C_1$ and $e_2 : E \to C_2$ Output: $E \to C$
Notation: similar of Entity Integration.
Initial conditions: C_E is the empty set and $f_{e'_1}, f_{e'_2}$ are empty functions.
(i) For all $x \in C_E$
$C_C := C_C \cup x$
$f_{e'_1} := f_{e'_1} \cup (f_{e_1}(x) \in C_{C_1}) \mapsto (x \in C_C)$
$f_{e'_2} := f_{e'_2} \cup (f_{e_2}(x) \in C_{C_2}) \mapsto (x \in C_C)$
(ii) For all $x \in C_{E_1}$ that is not in the image of f_{e_1}
$C_C := C_C \cup x$
$f_{e'_1} := f_{e'_1} \cup (x \in C_{C_1}) \mapsto (x \in C_C)$
(iii) For all $x \in C_{E_2}$ that is not in the image of f_{e_2}
$C_C := C_C \cup x$
$f_{e'_2} := f_{e'_2} \cup (x \in C_{C_2}) \mapsto (x \in C_C)$
return $f_{e'_1} \circ f_{e_1}$

The algorithm shows how to construct the set of concepts and the partial order of concepts (H^C) of the resulting ontology. The set relations R and the function rel is constructed in the same way and is not shown. The first step (loop *(i)*) adds concepts from E in C. The image of these concepts by f_{e_i} is collapsed in C. The loops *(ii)* and *(iii)* add C_C the concepts of C_{E_1} and C_{E_2} that are not in the image of $f_{e'_1}$ or $f_{e'_2}$. Then f'_{e_1} or f'_{e_2} are defined for these concepts. The compositions $(f_{e'_1}, g_{e'_1}) \circ (f_{e_1}, g_{e_1})$ ensure that each concept of C is either a collapsed element from C_1 and C_2 or an element of C_1 and C_2 that does not come from E.

2.3 Combined Integration

We defined ways of operating contextualized entities by combining entities (2.1) or contexts (2.2). In this section we show how to operate contextualized entities as a whole, considering both entity and context. We present the motivation for the operations and the formal definitions. We do not present the algorithms due to limitation of space.

The commutative square $C_2 \xleftarrow{c} C_1 \xleftarrow{e_1} E_1 \xrightarrow{e'} E_2 \xrightarrow{c} C_2$ defines a map between contextualized entities ensuring the coherence of entities and contexts. Following the left way ($c \circ e_1$), we ensure that C_2 is coherent with C_1, and thus, is also a context for E_1. Following the right way ($e_2 \circ e'$), we ensure that C_2 is a context for E_2, and as E_2 is also a context for E_1, C_2 is a context for E_1.

Definition 3 (Map Between Contextualized Entities) *Given two contextualized entities $E_1 \xrightarrow{e_1} C_1$ and $E_2 \xrightarrow{e_2} C_2$, a pair contextualized entities $(C_1 \xrightarrow{c} C_2, E_1 \xrightarrow{e'} E_2)$ is a map from e_1 to e_2 if $E_1 \xrightarrow{c \circ e_1} C_2 = E_1 \xrightarrow{e_2 \circ e'} C_2$ is a contextualized entity.*

2.3.1 The Relative Intersection

The relative intersection gives commonalities among entities with different contexts. It is the coherent intersection of two given contextualized entities with respect to a third contextualized entity, and is performed in three contextualized entities, as in the diagram $CE_1 \to CE \leftarrow CE_2$. The result is a contextualized entity defined by the commutative diagram of Figure 7 (a): a contextualized entity (CE') mapped into contextualized entities (CE_1, CE_2) which are mapped into the upper contextualized entity (CE). Thus, CE' is the more informative contextualized entity that is coherent with CE_1 and CE_2 with respect to CE.

The relative intersection is a parallel operation acting both on contexts and entities, as shows Figure 7 (b). The vertical arrows are contextualized entities. The pairs of links between entities (down square) and between contexts (upper square) are maps between contextualized entities. By definition of maps between contextualized entities, all the lateral squares of the cube in Figure 7(b) commute. The relative intersection will be defined in such a way that ensures that the bottom and top squares of the cube also commute.

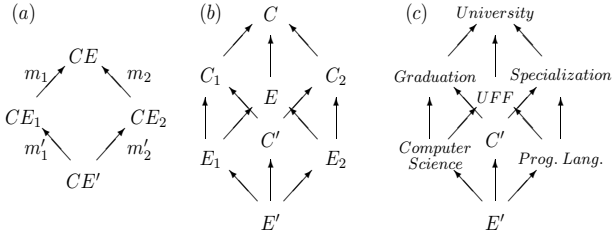

Figure 7. (a, b) **Relative Intersection.** (c) **An example.**

Definition 4 (Relative Intersection) *Given two maps between contextualized entities $m_1 : CE_1 \to CE$ and $m_2 : CE_2 \to CE$, the relative intersection of CE_1 and CE_2 with respect to CE is*

the contextualized entity CE', with maps $m'_1 : CE' \rightarrow CE_1$ and $m'_2 : CE' \rightarrow CE_2$ such that, (i) $m_1 \circ m'_1 = m_2 \circ m'_2$, and, (ii) For any other other contextualized entity CE'', with maps $m''_1 : CE'' \rightarrow CE_1$ and $m''_2 : CE'' \rightarrow CE_2$ there exists a unique map $! : CE'' \rightarrow CE'$ with $m'_1 \circ ! = m''_1$ and $m'_2 \circ ! = m''_2$.

2.3.2 The Collapsing Union

The collapsing union of contextualized entities acts both in context and entity of contextualized entities and results the union of them, possibly collapsing some components. In a dual way of subsection 2.3.1, it is performed in three contextualized entities forming a diagram as $CE_1 \leftarrow CE \rightarrow CE_1$. The result is a contextualized entity defined by the commutative diagram of Figure 8 (b): a contextualized entity CE' that is the target of links of both contextualized entities CE_1 and CE_2, thus any concept in CE_1 or CE_2 is mapped a component in CE'. Moreover, the concepts of CE are mapped to the same concept of CE' via links through CE_1 or CE_2. Thus, CE' is the less informative contextualized entity that is coherent with CE_1 and CE_2. Collapsed components of CE' are defined by CE. As well as the relative intersection, the collapsing union is a parallel operation acting both on contexts and entities (Figure 8 (b).

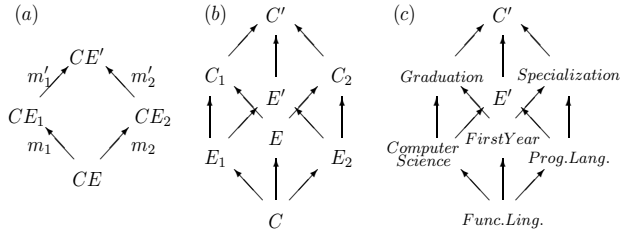

Figure 8. (a, b) The Collapsing Union. (c) An example.

Definition 5 (Collapsing Union) *Given two maps between contextualized entities $m_1 : CE \rightarrow CE_1$ and $m_2 : CE \rightarrow CE_2$, the collapsing union of CE_1 and CE_2 with respect to CE is the contextualized entity CE', with maps $m'_1 : CE_1 \rightarrow CE'$ and $m'_2 : CE_2 \rightarrow CE'$ such that, (i). $m'_1 \circ m_1 = m'_2 \circ m_2$, and, (ii). for any other other contextualized entity CE'', with maps $m''_1 : CE_1 \rightarrow CE''$ and $m''_2 : CE_2 \rightarrow CE''$ there exists a unique map $! : CE' \rightarrow CE''$ with $! \circ m'_1 = m''_1$ and $! \circ m'_2 = m''_2$.*

Examples: Consider that CE, CE_1 and CE_2 model Universities and Courses. CE has as context an ontology modeling Universities and as entity, the instance of university named "UFF". CE_1 has as context an ontology modeling Graduation Courses and as entity, the instance of Graduation Course "Computer Science". Finally, CE_2 has as context an ontology modeling Specialization Courses and, as entity, the instance "Programming Language". Figure 7 (c) shows the diagram that sketches the relative intersection of CE_1 and CE_2 with respect to CE. In CE', the resulting contextualized entity, C' expresses what the concepts "Graduation" and "Specialization" have in common, under the scope of "University". E' expresses the commonalities of "Computer Science" and "Programming Language" while courses of the University "UFF". The commutativity of the whole diagram ensures that "Computer Science", "Programming Language", "UFF" and their intersection are coherent with Graduations, Specializations and Universities. In Figure 8 (c), CE identifies "Functional Languages" as a first year discipline, and the mappings $CE \rightarrow CE_1$ and $CE \rightarrow CE_2$ indicate that it occurs in both Graduation in Computer Science and in Specialization in Programming Languages. The result of the collapsing union on this diagram produces a new course of graduation with emphasis in Programming Languages, where equivalent disciplines of both courses (as functional languages) are not duplicated.

3 The Formal Approach: Category Theory

The algebra of contextualized ontologies is an application of Category Theory. We avoided making explicit use of categorical terms to make the paper more readable for those who do not know this theory. In this section, we justify the use of Category Theory and show the correspondence between the concepts presented here and categorical concepts. The reader can find in the appropriated literature (for example, [6]) the proofs that the operations presented here are well defined. To complete the formal approach, in [5] the reader can see in that the adopted mechanism to structurally compose and decompose ontologies is accompanied by a model-theoretic and syntactic mechanism.

Category Theory provides a great power of integration and interoperability of heterogeneous entities because of the focus that is put in relationship (categorical morphisms) and not in the "things" being represented (categorical objects). "Things" are described abstractly, accordingly to their interactions. It is possible to define several categories (according to the kinds of "things" to be described) that can be related by relationships between categories (categorical functors). Functors that preserve properties of categories making possible the co-existence of heterogeneous "things". Category Theory offers a set of ways of combining entities, some of which (as colimits) are traditionally used to integration. It has been successfully used in situations where interoperability is a crucial point, as in formal specification of systems [11], software architecture [3, 4], and, more recently, in ontology integration [2, 10, 1]. In Context Aware Applications, Category Theory contributes not only formalizing the object of study, but also giving good directions to operationalize the concept and facilitating implementations.

Let $\mathcal{C}$ be a category with objects o and morphisms m. We cal $\mathcal{C}$ a *domain of knowledge*. The objects of $\mathcal{C}$ are en-

tities over which the domain of knowledge is constructed, and the morphisms of $\mathcal{C}$ are the mappings between these entities. When modeling an application, o is the concept over which the application is about and m express ways of relating this concept. A contextualized entity is an object of the category $\mathcal{C}^{\rightarrow}$, that is, the category that has as objects morphisms of $\mathcal{C}$ and as morphisms pairs of $\mathcal{C}$ morphisms (m, m') that defines commutative squares on $\mathcal{C}$, as in Definition 3. A map between contextualized entities is a morphism in $\mathcal{C}^{\rightarrow}$. The operation *entity integration* (Definition 1) is a pullback in $\mathcal{C}$. In order to ensure that any diagram of the form $o_1 \rightarrow o \leftarrow o_2$ has a pullback it is proved that $\mathcal{C}$ is *finitely complete*[5]. The operation *context integration* (Definition 2) is a pushout in $\mathcal{C}$. In order to ensure that any diagram of the form $o_1 \leftarrow o \rightarrow o_2$ has a pushout it is proved that $\mathcal{C}$ is *finitely cocomplete*[5]. The combined operations are performed in $\mathcal{C}^{\rightarrow}$, which is also finitely complete and cocomplete, as a consequence of completeness and cocompleteness of $\mathcal{C}$ [6]. The relative intersection (Definition 4) and collapsing union (Definition 5) are respectively the pullback and pushout in $\mathcal{C}^{\rightarrow}$.

4 Conclusion

Considering that contexts are essential to clarify the meaning of entities, and that applications that consider dynamic changes of the environment require new forms of representing the context, we present in this paper an algebra to support the representation of contexts and to emphasize its relationships with entities. Abstraction, modularity and reuse are important properties that guide this approach, which are achieved by the uniform representation of entities and contexts and by the compositional definitions of operations. The role of an object (as entity or context) is given by the net of links *from* or *to* this object. Thus, the meaning of an entity is dependent of its relationships. This abstract framework gives a sound basis for implementing systems where contexts play a central role. It also reaches the 5 properties raised in [9] for a formal model of context-awareness. It is *expansive* as the transitive composition of links makes possible to express how a distant entity can affect an agent's behavior. *Specificity* is ensured by the flexibility in the combination of context. The notion of context is *explicit* and *separated* from the entity what ensures control over its context. Maintenance of context is *transparent* to the entity, that is free of knowing internal details of the context. We add to these properties *interoperability* of heterogeneous descriptions. This framework support the coexistence of different kinds of categories defined over different techniques of knowledge representation, as long as each category is finitely complete and co-complete.

References

[1] T. Bench-Capon and G. Malcolm, Formalizing Ontologies and their Relations, in: K. V. Andersen, J. K. Debenham, R. Wagner, eds., *P*roceedings of 16th International Conference on Database and Expert Systems Applications (1999).

[2] I. Cafezeiro and E. H. Haeusler, Semantic Interoperability via Category Theory, Conferences in Research and Practice in Information Technology, Vol. 83. J. Grundy, S. Hartmann, A. H. F. Laender, L. Maciaszek and J. F. Roddick, Eds. 26th International Conference on Conceptual Modeling, ER 2007, Auckland, New Zealand.

[3] I. Cafezeiro and E. H. Haeusler, Categorical Limits and Reuse of Algebraic Specifications, in Advances in Logic, Artificial Intelligence and Robotics, Eds J. MihoroAbe and J. Silva, IOS Press, Amsterdan, (2002) 216–233.

[4] I. Cafezeiro and E. H. Haeusler, Algebraic Framework for Reverse Engineering on Specifications, to appear in Proceedings of the The 6th Congress of Logic Applied to Technology, LAPTEC2007, SP, Brazil.

[5] I. Cafezeiro and E. H. Haeusler, Description Logics as Institutions, Universidade Federal Fluminense, Niteroi, Brasil. (2007).

[6] S. MacLane, *C*ategories for the Working Matematician, Berlin: Springer Verlag(1997).

[7] A. Pujar and others, Whole-Plant Growth Stage Ontology for Angiosperms and Its Application in Plant Biology, *Plant Physiology - American Society of Plant Biologists*, Avaiable at http://www.plantphysiol.org/cgi/content/full/142/2/414 (2006)

[8] A. Ranganathan, R. E. McGrath, R. H. Campbell and M. D. Mikunas, The Use of Ontologies in a Pervasive Computing Environment, The knowledge Engeneering Review, **18:3** (2004) 209-220.

[9] G. C. Roman, C. Julien, J. Payton, A Formal Treatment of Context Awareness, *P*roceedings of FASE'04 Vol. 928-1, LNSC. (2004).

[10] M. Schorlemmer, and Y. Kalfoglou, Progressive Ontology Alignment for Meaning Coordination: An Information-Theoretic Foundation, *in* ' Proceedings of the 4th AAMAS'05', Utrecht, Holland.

[11] A. Tarlecki and D. Sannella, Algebraic preliminaries, in: E. Artesiano, Ed., *Algebraic Foundations of Systems Specification*, (Berlin, 1999) 13-30.