Mobile Netw Appl (2013) 18:391–397
DOI 10.1007/s11036-012-0404-0

Coalgebraic Aspects of Context-Awareness

Phan Cong Vinh · Nguyen Thanh Tung

Published online: 26 August 2012

Abstract This paper will be both to give an in-depth analysis as well as to present the new material on the notion of context-aware computing, an idea that computing can both sense and react accordantly based on its environment. The paper formalizes context-awareness process using coalgebraic language, including the coalgebraic definition of context-awareness, bisimulation between context-awarenesses, homomorphism between context-awarenesses and context-awarenesses as coalgebras. Discussions for further development based on this approach are also given.

Keywords context-awareness · context-aware computing

1 Introduction

In design of context-aware systems, one of the limitations of the current design approaches is that when increasing (fully or partially) the context-awareness of computing, the semantics and understanding of the context-awareness process become difficult to capture for the design [18]. As motivation, the context-awareness process should be carefully considered under a suitably rigorous mathematical structure to capture its semantics completely, and then support an automatic design process, in particular, and applications of context-aware computing, generally [16, 17, 19].

Both initial algebras and final coalgebras are mathematical tools that can supply abstract representations to aspects of the context-awareness process [17]. On the one hand, algebras can specify the operators and values. On the other hand, coalgebras, based on a collection of observers, are considered in this paper as a useful framework to model and reason about the context-awareness process. Both initiality and finality give rise to a basis for the development of context-awareness calculi directly based on and driven by the specifications. From a programming point of view, this paper provides coalgebraic structures to develop the applications in the area of context-aware computing.

A coalgebraic structure provides an expressive, powerful and uniform view of context-awareness, in which the observation of context-awareness processes plays a central role. The concepts of bisimulation and homomorphism of context-awareness are used to compute the context-awareness process.

2 Outline

The paper is a reference material for readers who already have a basic understanding of context-aware systems and are now ready to know the novel approach for formalizing context-awareness in context-aware systems using coalgebraic language.

Formalization is presented in a straightforward fashion by discussing in detail the necessary components

P. Cong Vinh (✉)
IT Department, NTT University, 300A Nguyen Tat Thanh St., Ward 13, District 4, HCM City, Vietnam
e-mail: pcvinh@ntt.edu.vn

N. Thanh Tung
International School, Vietnam National University in Hanoi, 144 Xuan Thuy St., Cau Giay District, Hanoi, Vietnam
e-mail: tungnt@isvnu.vn

and briefly touching on the more advanced components. Some significant coalgebraic aspects, including justifications needed in order to achieve the particular results, are presented.

The rest of this paper is organized as follows: Section 3 briefly describes related work and existing concepts. Coalgebraic definition of context-awareness is the subject of Section 4. Section 5 presents relation of bisimulation between context-awarenesses. Homomorphism between context-awarenesses is investigated in Section 6. We consider context-awarenesses as coalgebras in Section 7. In Section 8, we briefly discuss our further development. Finally, Section 9 is a brief summary.

3 Related work and existing concepts

Most notions and observations of this paper are instances of a theory called universal coalgebra [5, 12]. In [11, 14], some recent developments in coalgebra are presented.

The programming paradigm with functions called functional programming [1, 3, 4, 7, 9] treats computation as the evaluation of mathematical functions. Functional programming emphasizes the evaluation of functional expressions. The expressions are formed by using functions to combine basic values.

The notion of bisimulation is a categorical generalization that applies to many different instances of infinite data structures, various other types of automata, and dynamic systems [5, 11, 12]. In theoretical computer science, a bisimulation is an equivalence relation between abstract machines, also called the abstract computers or state transition systems (i.e., a theoretical model of a computer hardware or software system) used in the study of computation. Abstraction of computing is usually considered as discrete time processes. Two computing systems are bisimular if, regarding their behaviors, each of the systems "simulates" the other and vice-versa. In other words, each of the systems cannot be distinguished from the other by the observation.

Homomorphism is one of the fundamental concepts in abstract algebra [10], which scrutinizes the sets of algebraic objects, operations on those algebraic objects, and functions from one set of algebraic objects to another. A function that preserves the operations on the algebraic objects is known as a homomorphism. In other words, if an algebraic object includes several operations, then all its operations must be preserved for a function to be a homomorphism in that category [8, 15].

4 Coalgebraic definition of context-awareness

Definition 1 (Context-Awareness) Let T be a (finite or infinite) set of contexts. A context-awareness with set of contexts T is a pair $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle \rangle$ consisting of

- a set SYS of *states*,
- an *output function* $o_{\mathsf{SYS}} : \mathsf{SYS} \longrightarrow (T \longrightarrow \mathbf{2})$, and
- an *evolution function* $e_{\mathsf{SYS}} : \mathsf{SYS} \longrightarrow (T \longrightarrow \mathsf{SYS})$.

where

- $\mathbf{2} = \{0, 1\}$,
- o_{SYS} assigns, to a state c, a function $o_{\mathsf{SYS}}(c) : T \longrightarrow \mathbf{2}$, which specifies the value $o_{\mathsf{SYS}}(c)(t)$ that is reached after a context t has been developed. In other words,

$$o_{\mathsf{SYS}}(c)(t) = \begin{cases} 1 \text{ when } t \text{ becomes fully available, or} \\ 0 \text{ otherwise} \end{cases}$$

- Similarly, e_{SYS} assigns, to a state c, a function $e_{\mathsf{SYS}}(c) : T \longrightarrow \mathsf{SYS}$, which specifies the state $e_{\mathsf{SYS}}(c)(t)$ that is reached after a context t has been developed. Sometimes $c \xrightarrow{t} c'$ is used to denote $e_{\mathsf{SYS}}(c)(t) = c'$.

Generally, both the state space SYS and the set T of contexts may be infinite. If both SYS and T are finite, then we have a finite context-awareness, otherwise we have an infinite context-awareness. The function $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle$ can be explained as detailed from two different aspects of context-awareness, using an analysis and universal view, below.

4.1 Analysis view of context-awareness

The function $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle$ is a tool or technology for building the context-awareness process. The set of contexts T as input data flow join with the state information of a state-based structure, denoted by SYS, to built up the context-awareness process that will eventually modify the state information, thus influencing the change in more later states.

In the denoting way as same as functional programming, the set of functions $T \longrightarrow \mathbf{2}$ is denoted by $\mathbf{2}^T$ and the set of functions $T \longrightarrow \mathsf{SYS}$ by SYS^T. In other words, $\mathbf{2}^T = \{f \mid f : T \longrightarrow \mathbf{2}\}$ and $\mathsf{SYS}^T = \{g \mid g : T \longrightarrow \mathsf{SYS}\}$. Therefore, the signature of $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle$ becomes

$$\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle : \mathsf{SYS} \longrightarrow (\mathbf{2} \times \mathsf{SYS})^T \qquad (1)$$

We recognize that context-awareness process represented by function $e_{\mathsf{SYS}}:\ \mathsf{SYS} \longrightarrow \mathsf{SYS}^T$ does not apply in all cases. This is expressed by observing the output of e_{SYS} in a more refined context: SYS is replaced by $\mathbf{1}+\mathsf{SYS}$, where $\mathbf{1}=\{*\}$ containing an exception value, and $e_{\mathsf{SYS}}:\ \mathsf{SYS} \longrightarrow (\mathbf{1}+\mathsf{SYS})^T$ is a *partial* function returning either a valid output or an exception value, in which the outcome of e_{SYS} is deadlocked, in the sense that the evolution of the context-awareness observed in the state space can be undefined. In other words, if function $e_{\mathsf{SYS}}(c)$ in $(\mathbf{1}+\mathsf{SYS})^T$ and context t in T cause $e_{\mathsf{SYS}}(c)(t)=*$ then it means that $e_{\mathsf{SYS}}(c)$ is undefined in t, simply denoted by $e_{\mathsf{SYS}}(c)(t)^*$.

4.2 Universal view of context-awareness

The function $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle$ can suggest an alternative model for context-awareness. Instead of using this to build the context-awareness process, context-awareness processes and their evolution can be observed. These can give rise to different context-awareness forms and the context-awareness processes must thus be scrutinized.

The universal view will equip itself with the right "*glass*"—that is a tool with which to observe and which necessarily gives rise to a particular "*shape*" for observation. The relation be specified is between the input data flow as a set of contexts and the output states depending on the particular kind of observation we want to perform. In other words, our focus becomes the *universe* or, more pragmatically, the *state space*. The observed state being produced from the context-awareness is just one among other possible observations. The basic concepts required to support an observation of context-awareness processes consist of:

- A *glass*: $\odot$
- An *observation structure*: state space $\xrightarrow{\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle} \odot$ state space

Informally, the glass can be thought of as providing a shape which $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle$ is able to deal with in each of the above mentioned situations. From a technical point of view, as this paper aims to make clear, the pair $\langle$ state space, $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$, as above, constitutes a *coalgebra* with interface $\odot$. Coalgebras provide uniform models for context-awareness processes.

Differently from the algebraic approach to the operations, which are completely defined by a set of operators, context-awarenesses we scrutinize here are specified by determining their behavior. This is done not only by an external input specification, but also by the internal state in the state-based structure to which there is no direct access in general. Such context-awarenesses are inherently dynamic and have an observable behavior, but their internal states remain hidden and have therefore to be identified if not distinguishable by observation. Our approaches throughout the paper to model and calculate context-awarenesses are characterized by:

- A *state space*, which evolves and persists in time;
- *Interaction* with environment during the context-awareness process;

5 Bisimulation between context-awarenesses

Definition 2 A *bisimulation* between two context-awarenesses $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ and $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'}\rangle\rangle$ is a relation $R \subseteq \mathsf{SYS}\times\mathsf{SYS}'$ with, for all c in SYS, c' in SYS' and t in T

$$\text{if } c\ R\ c' \text{ then } \begin{cases} o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(c')(t) \text{ and} \\ e_{\mathsf{SYS}}(c)(t)\ R\ e_{\mathsf{SYS}'}(c')(t). \end{cases}$$

In other words,

$$\text{if } \langle c, c'\rangle \in R \text{ then } \begin{cases} o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(c')(t) \qquad \text{and} \\ \langle e_{\mathsf{SYS}}(c)(t), e_{\mathsf{SYS}'}(c')(t)\rangle \in R. \end{cases}$$

Two states that are related by a bisimulation relation are observationally indistinguishable in that:

- They give rise to the same observations, and
- Applying the same context on both states will lead to two new states that are indistinguishable again.

The only thing we can *observe* about state of a context-awareness is whether it is 1 or 0 in **2**. We can offer a context that leads to a new state. For this new state, we can of course observe again whether it is 1 or 0.

A bisimulation between $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ and itself is called a bisimulation *on* $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$. We write $c \sim c'$ whenever there exists a bisimulation R with $c\ R\ c'$.

Proposition 1 *Union (denoted by $\cup$) of bisimulations on $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ is a bisimulation.*

Proof In fact, if $R_1 \subseteq \mathsf{SYS} \times \mathsf{SYS}$ and $R_2 \subseteq \mathsf{SYS} \times \mathsf{SYS}$ are any two bisimulations then $R_1 \cup R_2$ is also bisimulation because every $\langle x, y \rangle \in R_1 \cup R_2$ is either $\langle x, y \rangle \in R_1$ or $\langle x, y \rangle \in R_2$. In other words, it is formally denoted by

$$R_1 \cup R_2 = \{\langle x, y \rangle \mid x\ R_1\ y \text{ or } x\ R_2\ y\}$$

□

Proposition 2 *Given bisimulations $R_1 \subseteq \mathsf{SYS} \times \mathsf{SYS}'$ and $R_2 \subseteq \mathsf{SYS}' \times \mathsf{SYS}''$, the relational composition of R_1 with R_2 denoted $R_1 \circ R_2$ is a bisimulation.*

Proof In fact, the relational composition of two bisimulations $R_1 \subseteq \mathsf{SYS} \times \mathsf{SYS}'$ and $R_2 \subseteq \mathsf{SYS}' \times \mathsf{SYS}''$ is the bisimulation obtained by

$$R_1 \circ R_2 = \{\langle x, y \rangle \mid x\ R_1\ z \text{ and } z\ R_1\ y \text{ for some } z \in \mathsf{SYS}'\}$$

□

Definition 3 The union of all bisimulations on $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle\rangle$ is the greatest bisimulation. The greatest bisimulation is called the *bisimulation equivalence* or *bisimilarity* [11], again denoted by the operation $\sim$.

Proposition 3 *The bisimilarity is an equivalence relation.*

Proof In fact, a bisimilarity on a set of states SYS is a binary relation on SYS that is reflexive, symmetric and transitive; i.e., it holds for all a, b and c in SYS such that

– (Reflexivity) $a \sim a$
– (Symmetry) if $a \sim b$ then $b \sim a$
– (Transitivity) if $a \sim b$ and $b \sim c$ then $a \sim c$ □

6 Homomorphism between context-awarenesses

Definition 4 A *homomorphism* between $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle\rangle$ and $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'} \rangle\rangle$ is any function $h : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$ with, for all c in SYS and t in T

– $o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(h(c))(t)$ and
– $h(e_{\mathsf{SYS}}(c)(t)) = e_{\mathsf{SYS}'}(h(c))(t)$

We can use a commutative diagram as a diagram of objects and homomorphisms such that, when picking two objects, we can follow any path through the diagram and obtain the same result by composition (see Fig. 1).

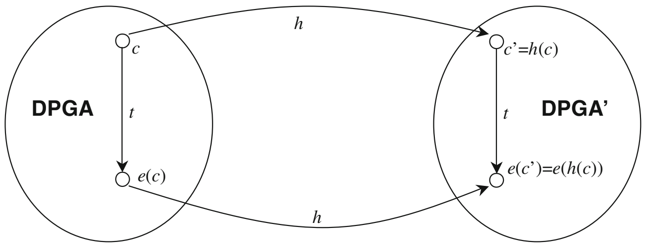

Fig. 1 Commutative diagram of the homomorphism h

Definition 5 A context-awareness $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle\rangle$ is a "*subcontext-awareness*" of $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'} \rangle\rangle$ if $\mathsf{SYS} \subseteq \mathsf{SYS}'$ and the inclusion function $i : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$ (i.e., $i(c) = c$) is a homomorphism.

For a state c' in SYS', let $\langle c' \rangle$ denote the subcontext-awareness *generated* by c'.

Proposition 4 *$\langle c' \rangle$ is the smallest subcontext-awareness of SYS' containing c'.*

Proof This can be obtained by including all states from SYS' that are reachable via a finite number of evolutions from c'. □

Proposition 5 *Given $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'} \rangle\rangle$ and $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle\rangle = \langle c' \rangle$, the functions o_{SYS} and e_{SYS} are uniquely determined.*

Proof This follows from Definitions 4 and 5. □

Proposition 6 *For a homomorphism $h : \mathsf{SYS}' \longrightarrow \Gamma$ and $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}} \rangle\rangle$ is a subcontext-awareness of $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'} \rangle\rangle$, $\langle h(\mathsf{SYS}), \langle o_{h(\mathsf{SYS})}, e_{h(\mathsf{SYS})} \rangle\rangle$ is a subcontext-awareness of $\langle \Gamma, \langle o_\Gamma, e_\Gamma \rangle\rangle$.*

Proof This is as a result of the Definitions 4 and 5. □

Proposition 7 *For a homomorphism $h : \mathsf{SYS}' \longrightarrow \Gamma$ and c' in SYS', $h(\langle c' \rangle) = \langle h(c') \rangle$.*

Proof This is a result of the Definitions 4 and 5. □

The notions of context-awareness, homomorphism and bisimulation are closely related, in fact

Proposition 8 *The function $h : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$ is a homomorphism iff its relation $R = \{\langle c, h(c) \rangle \mid c \in \mathsf{SYS}\}$ is a bisimulation.*

Proof Firstly, prove that *homomorphism* $h \Longrightarrow R = \{\langle c, h(c)\rangle\}$ *is a bisimulation*. In fact, the homomorphism of h is given by $h(e_{\mathsf{SYS}}(c)(t)) = e_{\mathsf{SYS}'}(h(c))(t)$ and $o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(h(c))(t) = o_{\mathsf{SYS}'}(c')(t)$. Thus, if every $\langle e_{\mathsf{SYS}}(c)(t), h(e_{\mathsf{SYS}}(c)(t))\rangle$ is in $R \subseteq \mathsf{SYS} \times \mathsf{SYS}'$ then $\langle e_{\mathsf{SYS}}(c)(t), e_{\mathsf{SYS}'}(h(c))(t)\rangle$ is also in R. In other words, this is $\langle e_{\mathsf{SYS}}(c)(t), e_{\mathsf{SYS}'}(c')(t)\rangle$ in R. R is a bisimulation.

Finally, prove the opposite way that the *bisimulation of* $R = \{\langle c, h(c)\rangle\} \Longrightarrow h$ *is a homomorphism*. In fact, the results of bisimulation are $o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(h(c))(t) = o_{\mathsf{SYS}'}(c')(t)$ and $\langle e_{\mathsf{SYS}}(c)(t), e_{\mathsf{SYS}'}(h(c))(t)\rangle$ in R. This gives rise to a definition as $h(e_{\mathsf{SYS}}(c)(t)) = e_{\mathsf{SYS}'}(h(c))(t)$. Thus, h is a homomorphism. □

Proposition 9 *Bisimulations are themselves context-awarenesses.*

Proof If R is a bisimulation between SYS and SYS', then $o_R:\ R \longrightarrow \mathbf{2}^T$ and $e_R:\ R \longrightarrow R^T$. For $\langle c, c'\rangle$ in R and t in T, the functions of o_R and e_R given by $o_R(\langle c, c'\rangle)(t) = \langle o_{\mathsf{SYS}}(c)(t), o_{\mathsf{SYS}'}(c')(t)\rangle$ and $e_R(\langle c, c'\rangle)(t) = \langle e_{\mathsf{SYS}}(c)(t), e_{\mathsf{SYS}'}(c')(t)\rangle$, define a context-awareness $\langle R, \langle o_R, e_R\rangle\rangle$. □

7 Context-awarenesses as coalgebras

7.1 Interfaces

The context-awareness $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ is considered as the structure observed through the following *interface*.

$$\odot \ = \ (\mathbf{2} \times \_)^T \tag{2}$$

Applying this interface to the observable state space SYS, we have

$$\odot\, \mathsf{SYS} \ = \ (\mathbf{2} \times \mathsf{SYS})^T \tag{3}$$

This interface of $\odot$ can be regarded as a mapping to decompose the observable state space SYS into an observation context $(\mathbf{2} \times \_)^T$ of the state space.

7.2 Coalgebras

Definition 6 Let $\odot : \mathsf{SYS} \longrightarrow \mathsf{SYS}$ be a functor on the category of states and functions. A $\odot$-*coalgebra* is a pair $\langle \mathsf{SYS}, f\rangle$ consisting of a set SYS and a function $f : \mathsf{SYS} \longrightarrow \odot\, \mathsf{SYS}$.

Context-awarenesses are coalgebras of the functor $\odot$, which is defined on sets SYS by $\odot\, \mathsf{SYS} = (\mathbf{2}\times \mathsf{SYS})^T$. Now for a context-awareness $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$, the functions $o_{\mathsf{SYS}} : \mathsf{SYS} \longrightarrow \mathbf{2}^T$ and $e_{\mathsf{SYS}} : \mathsf{SYS} \longrightarrow \mathsf{SYS}^T$ can be combined into one function $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle : \mathsf{SYS} \longrightarrow (\mathbf{2} \times \mathsf{SYS})^T$, which maps c in SYS into the pair $\langle o_{\mathsf{SYS}}(c), e_{\mathsf{SYS}}(c)\rangle$. In this way, the context-awareness $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ has been represented as a $\odot$-coalgebra $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle : \mathsf{SYS} \longrightarrow \odot\, \mathsf{SYS}$. Through this coalgebraic representation of context-awarenesses, there exist a number of notions and results on coalgebras in general, which can now be applied to context-awarenesses. Notably there are two Definitions 7 and 8 below.

Definition 7 Consider an arbitrary functor $\mathsf{T} : \mathsf{SYS} \longrightarrow \mathsf{SYS}$ and let (SYS, f) and (SYS', f') be two T-coalgebras. A function $h : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$ is a homomorphism of T-coalgebras, or T-*homomorphism*, if $\mathsf{T}\, h \circ f = f' \circ h$.

In order to apply this definition to the case of context-awarenesses, we still have to give the definition of the functor T on functions, which is as follows. For a function $h : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$, the function $\mathsf{T}\ h : (\mathbf{2} \times \mathsf{SYS})^T \longrightarrow (\mathbf{2} \times \mathsf{SYS}')^T$ is defined, for any head(of the stream) in $\mathbf{2}^T$ and tail (of the stream) in SYS^T by $\mathsf{T}(h)(\langle \mathsf{head}, \mathsf{tail}\rangle) = \langle \mathsf{head}, h \circ \mathsf{tail}\rangle$. Now consider two context-awarenesses, i.e., T-coalgebras, $\langle \mathsf{SYS}, \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle\rangle$ and $\langle \mathsf{SYS}', \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'}\rangle\rangle$, where $\langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle : \mathsf{SYS} \longrightarrow \mathsf{T}\ \ \mathsf{SYS}$ and $\langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'}\rangle : \mathsf{SYS}' \longrightarrow \mathsf{T}\ \ \mathsf{SYS}'$. According to the definition, a function $h : \mathsf{SYS} \longrightarrow \mathsf{SYS}'$ is a homomorphism of T-coalgebras if $\mathsf{T}\ \ h \circ \langle o_{\mathsf{SYS}}, e_{\mathsf{SYS}}\rangle = \langle o_{\mathsf{SYS}'}, e_{\mathsf{SYS}'}\rangle \circ h$, which is equivalent to $o_{\mathsf{SYS}}(c)(t) = o_{\mathsf{SYS}'}(h(c))(t)$ and $h(e_{\mathsf{SYS}}(c)(t)) = e_{\mathsf{SYS}'}(h(c))(t)$, for all c and t. Note that this is precisely the definition of homomorphism.

The commutativity is represented by the following diagram:

$$\begin{tikzcd}
\mathsf{SYS} \arrow[r, "f=\langle o_{\mathsf{SYS}}\text{,}t_{\mathsf{SYS}}\rangle"] \arrow[d, "h"'] & (\mathbf{2} \times \mathsf{SYS})^T \arrow[d, "\mathsf{T}\ h"] \\
\mathsf{SYS}' \arrow[r, "f=\langle o_{\mathsf{SYS}'}\text{,}t_{\mathsf{SYS}'}\rangle"] & (\mathbf{2} \times \mathsf{SYS}')^T
\end{tikzcd}$$

Definition 8 A relation $R \subseteq \mathsf{SYS} \times \mathsf{SYS}'$ is called a T-*bisimulation* between T-coalgebras $\langle \mathsf{SYS}, f\rangle$ and $\langle \mathsf{SYS}', f'\rangle$ if there exists a T-coalgebra structure g :

$R \longrightarrow \mathsf{T}\ R$ on R such that the projections $\pi_1 : R \longrightarrow \mathsf{SYS}$ and $\pi_2 : R \longrightarrow \mathsf{SYS}'$ are T-homomorphisms.

The commutativity is represented by the following diagram:

$$\begin{tikzcd}
\mathsf{SYS} \arrow[d, "f"] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, "g"] & \mathsf{SYS}' \arrow[d, "f'"] \\
\mathsf{T}\,\mathsf{SYS} & \mathsf{T}\,R \arrow[l, "\mathsf{T}\,\pi_1"'] \arrow[r, "\mathsf{T}\,\pi_2"] & \mathsf{T}\,\mathsf{SYS}'
\end{tikzcd}$$

For a functor $\mathsf{T} : \mathsf{SYS} \longrightarrow \mathsf{SYS}$, the family of T-coalgebras together with the T-homomorphisms between them, forms a *category* [15]. In this category, a coalgebra (Γ, f) is *final* if there exists from any coalgebra precisely one homomorphism into (Γ, f). The interest of final coalgebras lies in the fact that they satisfy the following *coinduction proof principle* [11, 13]: if there exists a T-bisimulation between γ and γ' in Γ then γ and γ' are equal. This follows the finality of (Γ, f).

8 Discussions

The aim of this paper has been both to give an in-depth analysis as well as to present the new material on the notion of context-aware computing. In this section, we briefly discuss our further development. By our approach, the notion of *Aspect/Component-Oriented* model for context-aware computing has been introduced, in which contextware is oriented by "aspect" development and stateware is oriented by "component" approach. Aspect-Oriented Contextware relies on "separation of concerns" in all phases of the contextware development lifecycle, while Component-Oriented Stateware greatly depends on pre-fabricated "components" for building the data processing needs of stateware. Incorporating both the emerging programming approaches based on aspect [6] and component [2] could unify the two specialized concepts of "modularity" and "separation of concerns" in model.

9 Conclusions

In this paper, we have rigorously approached to the notion of context-awareness in context-aware systems from which coalgebraic aspects of the context-awareness emerge. The coalgebraic model is used to formalize the unifying frameworks for context-awareness and evolution of the context-awareness processes, respectively.

Acknowledgements Thank you to NTTUFSTD[1] for the constant support of our work which culminated in the publication of this paper. As always, we are deeply indebted to the anonymous reviewers for their helpful comments and valuable suggestions which have contributed to the final preparation of the paper.

References

1. Barbosa LS (2000) Components as processes: an exercise in coalgebraic modeling. In: Smith SF, Talcott CL (eds) 4th international conference on formal methods for open object-based distributed systems, IFIP TC6/WG6.1, 6–8 September 2000. Kluwer Academic Publishers, Stanford, CA, USA, pp 397–417
2. Barbosa LS, Liu Z (eds) (2005) Proceedings of the 2nd international workshop on formal aspects of component software (FACS), Macao, 24–25 October 2005. UNU/IIST. ENTCS
3. Cockett R, Spencer D (1991) Strong categorical datatypes I. In: Seely RAG (ed) International summer meeting on category theory, AMS Canadian Mathematical Society, 23–30 June 1991. AMS, Montréal, Québec, Canada, pp 141–169
4. Hagino T (1987) A typed lambda calculus with categorical type constructors. In: Pitt DH, Poigné A, Rydeheard DE (eds) Category theory and computer science, September 1987. Lecture notes in computer science, vol 283. Springer–Verlag, Edinburgh, UK, pp 140–157
5. Jacobs B, Rutten J (1997) A tutorial on (Co)Algebras and (Co)Induction. Bulletin of EATCS 62:222–259
6. Kiczales G, Lamping J, Menhdhekar A, Maeda C, Lopes C, Loingtier JM, Irwin J (1997) Aspect-oriented programming. In: Akşit M, Matsuoka S (eds) 11th European conference on object-oriented programming (ECOOP), 10 June 1997. Lecture notes in computer science, vol 1241. The paper originating AOP, Springer–Verlag, Jyväskylä, Finland, pp 220–242
7. Kieburtz RB (1998) Reactive functional programming. In: Gries D, Roever WPde (eds) Programming concepts and methods (PROCOMET), IFIP international federation for information processing, 8–12 June 1998. Chapman and Hall, Shelter Island, NY, USA, pp 263–284
8. Levine M (1998) Categorical algebra. In: Benkart G, Ratiu TS, Masur HA, Renardy M (eds) Mixed motives, mathematical surveys and monographs, vol 57, chapter I, II, II of part II, American Mathematical Society, USA, pp 373–499
9. Meijer E, Fokkinga M, Paterson R (1991) Functional programming with bananas, lenses, envelopes and barbed wire. In: Hughes J (ed) ACM conference on functional programming languages and computer architecture, 26–30 August 1991. Lecture notes in computer science, vol 523. Springer–Verlag, Cambridge, MA, USA, pp 124–144
10. Rotman JJ (2002) Advanced modern algebra, 1st edn. Prentice Hall, USA
11. Rutten JJMM (1998) Automata and coinduction (an exercise in coalgebra). In: 9th international conference on concurrency theory (CONCUR), 8–11 September 1998. Lecture notes in computer science, vol 1466. Springer–Verlag, Nice, France, pp 194–218
12. Rutten JJMM (2000) Universal coalgebra: a theory of systems. Theor Comp Sci 249(1):3–80

[1]The NTTU foundation for Science and Technology Development.

13. Rutten JJMM (2001) Elements of stream calculus (an extensive exercise in coinduction). In: Proceedings of the 17th annual conference on mathematical foundations of programming semantics (MFPS ’01), Aarhus, Denmark, ENTCS, vol 45. Elsevier Science B.V., 66 p
14. Rutten JJMM (2005) Algebra, bitstreams, and circuits. Technical Report SEN-R0502, CWI, Amsterdam, The Netherlands
15. van Oosten J (2002) Basic category theory. Department of Mathematics, Utrecht University, The Netherlands
16. Vinh PC (2009) Autonomic computing and networking, chapter formal aspects of self-* in autonomic networked computing systems. Springer, pp 381–410
17. Vinh PC (2009) Dynamic reconfigurability in reconfigurable computing systems: formal aspects of computing, 1st edn. VDM Verlag, 236 p
18. Vinh PC (2011) Formal and practical aspects of autonomic computing and networking: specification, development, and verification, chapter formal specification and verification of self-configuring p2p networking: a case study in mobile environments, 1st edn. IGI Global, pp 170–188
19. Vinh PC (2012) Data intensive distributed computing in data aware self-organizing networks. In: Transactions on computational science XV: special issue on advances in autonomic computing: formal engineering methods for nature-inspired computing systems. Springer, Berlin, pp 74–107