Formal Context Representation and Calculus for Context-Aware Computing

Ammar Alsaig(✉), Vangalur Alagar, and Nematollaah Shiri

Concordia University, Montreal, QC, Canada
{a_alsaig,alagar,shiri}@encs.concordia.ca

Abstract. Context is a rich concept that is mostly understood and used with different representations and interpretations in many different fields. This variety of usage adds both richness and vagueness, thus creating more complexity to comprehension, interpretation, and reasoning with contexts. As pervasive computing technology becomes more and more intrusive there is a need to construct formally verifiable context-aware computing environment, in which human dignity is preserved through safety, security, and privacy. These features cannot be ensured unless context notion is formalized, both in representation and reasoning. Motivated by this concern this paper introduces a formal context representation and a context calculus which can be used to build context models for many applications.

Keywords: Context modeling · Context-awareness
Formal representation · Reasoning

1 Introduction

The term "context" has been around for centuries, and consists of the ancient Greek words "con" (meaning "together") and "texere" (meaning "to weave"). Context provides the circumstances where an event, statement, or idea occurs, and help to understand and evaluate the occurrence. Context has been the subject of numerous studies by philosophers, psychologists [7], and linguists [6]. Since early 1980s, the importance of context has been recognized in different research areas including information retrieval, knowledge representation and expert systems, mechanized formal reasoning in AI, and analysis of computer programs, most notably by Weyhrauch [27], McCarthy [19], Akman and Surav [1], and Giunchiglia [14]. Dowley et al. [10] was the first one to propose the notion of *dimensions* for contexts after recognizing that "hidden contexts" add *dimensions*

This work was supported in part by grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada. The first author is supported by a scholarship from the Saudi Arabian Government.

P. Cong Vinh and V. Alagar (Eds.): ICCASA 2018/ICTCC 2018, LNICST 266, pp. 3–13, 2019.
https://doi.org/10.1007/978-3-030-06152-4_1

to expressions. Much later in 1994, Schilit et al. [22] mentioned the necessity of "dimensions" for "context-aware computing" in system development. However, they neither proposed a "dimension-based" representation nor developed a reasoning framework. Almost 10 years later, Wan [25] proposed a context representation using dimensions and tags and used them to introduce context as a first class citizen in Lucid programming language extended with intensional semantics. The formal semantics of intentional programming and architectures for context-aware computing were also discussed by Wan [3,24].

In this paper, we pick up these notions, formalize, generalize and enrich them in three tiers. In Tier 1 we define *Context Schema*, in Tier 2 we derive *Typed Context Schemas*, and in Tier 3 we formalize *Contexts* as a set of families of *Context Instances* generated from Typed Schemas. For all the three tiers a uniform representation is used, under set theoretical setting. We defined operations in Tier 1 and show they are inherited in successive tiers, and in particular how it can be enriched in Tier 3. Our approach, because of its generality and simplicity, has the potential for generating different families of contexts for different applications within an application domain.

2 Related Work

The literature on context is vast and diverse and a comprehensive survey of the topic is beyond the scope of this paper. In what follows we review research most relevant to our work on context modeling and formalism. We group the current work on context into six categories detailed as follows:

1. *Interdisciplinary Emphasis:* The nature of diversity and interdisciplinary research can be seen in LNAI publication series "Modeling and Using Context", proceedings of CONTEXT International conference being held from 1997 [18]. More recent collection of papers in [4] only reinforces the practice of a variety of mostly informal notations and views in this interdisciplinary research on context.
2. *Human Computer Interface Group (HCI):* Early in 2001 the works of Dey et al. [8] and Winograd [28] in HCI are based on intuitive ad-hoc notations, bordering on vagueness and informality. These papers bring out also the disagreement within this group in conceptualizing and modeling contexts.
3. *Languages:* The works of Dowley et al. [10], Sato et al. [21], and Alagar et al. [3] use contexts at different levels of abstraction in intensional and functional programming languages. Sato et al. uses a formal context representation based on relational semantics while Sato et al. uses λ calculus notation.
4. *AI and Reasoning:* The works of Weyhrauch [27], McCarthy [19,20], Guha [16], Akman and Surav [1], Shoham [23], and Giunchiglia [14] are some of the early ground-breaking works on logic of contexts and context-based reasoning. None of them use any formal method to represent contexts.

5. *Pervasive Computing:* Based on the survey papers [9,11], it is evident that the notion of context in pervasive computing is still mostly ad-hoc. The formalisms attempted by a few are non-rigorous, either domain-specific or application dependent [5]. Besides, there is no consensus on what a context should be and how it should be represented for pervasive computing application domain. While formal representation and reasoning procedures are quite important to reason about pervasive computing applications that involve human safety and privacy, they are not emphasized in these works.
6. *Context-aware Computing:* The introduction of dimensions by Dowley et al. [10] to deal with hidden contexts in intensional programming languages perhaps influenced Schilit et al. [22] to incorporate it in developing context-aware systems, although there is no report such an attempt was made. Later on, the context representation was formalized by Wan [25] and subsequently used in building context-aware systems [3,24]. They also developed a context calculus, and showed how to reuse their context toolkit in different applications such as privacy and security enforcement [2].

Against this background we claim our contribution in this paper is both novel and new. In a computerized system context must capture both *internal* and *external* settings. Since, the external world (environment) is composed of several dimensions and the knowledge to execute different operations within the system will come different "worlds", context is "multi-dimensional". The work of Wan [25], which conceptualized this notion of context through an aggregation of $<dimension, tag>$ pairs had the restriction that each dimension occurs once in the aggregation, and with each dimension only one atomic value is associated. In our theory discussed below, context schemas, not just context instances, are defined, thereby the restrictions in [25] are removed.

3 Tier 1: Context Schema Representation and Calculus

Context schema (CS) is abstract, and its generality gives flexibility for a practical system designer to choose attributes to be associated with a dimension. Once we fix the (non-empty) set of dimensions $\mathbb{D}$ and the (non-empty) set of attributes $\mathbb{A}$, the set $S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$ of all CSs is fixed. Formally, a CS over the pair $(\mathbb{D}, \mathbb{A})$ is the set of pairs $\mathscr{C} = \{d : A_d \mid d \in \mathbb{D} \wedge A_d \subseteq \mathbb{A}\}$.

Example 1. To define a conference context schema we first define the dimension set $\mathbb{D} = \{Date, Time, Location\}$, and attribute set $\mathbb{A} = \{a_1, a_2, a_3, a_4\}$. Next the conference context schema can be defined as $\mathscr{C}_{\text{conf}} = \{Date : \{a_1, a_2\}, Time : \{a_3\}, Location : \{a_4\}\}$.

In order to access information of an already constructed context schema, two functions are defined. The "DIM" function extracts the set of dimensions in a context schema, and the "ATT" function extracts the set of attributes associated with a dimension in a context schema. Formally, $DIM : S_{\mathscr{C}} \rightarrow \mathbb{P}(\mathbb{D})$, $ATT : \mathbb{D} \times S_{\mathscr{C}} \rightarrow \mathbb{P}(\mathbb{A})$, such that $DIM(\mathscr{C}) = \{d | < d, A_d > \in \mathscr{C}\}$, and $ATT(d, \mathscr{C}) = A_d$.

Example 2. For the "conference context schema" introduced in Example 1 $DIM(\mathscr{C}_{\text{conf}}) = \{Date, Time, Location\}$, $ATT(Date, \mathscr{C}_{\text{conf}}) = \{a_1, a_2\}$, $ATT(Time, \mathscr{C}_{\text{conf}}) = \{a_3\}$, and $ATT(Location, \mathscr{C}_{\text{conf}}) = \{a_4\}$.

For the sake of completeness we include "Null Context Schema" and "Full Context Schema" in out theory. If $DIM(\mathscr{C}) = \phi$, $A_d = \phi$ we get the Null Context Schema $\mathscr{C}_\phi$. If $DIM(\mathscr{C}) = \mathbb{D}$, $A_d = \mathbb{A}$ we get "Full Context Schema". McCarthy [19] postulated that "every context is contained in an outer context". In general, a context can contain many inner contexts. However, no formal definition for "containment relationship" exists because of lack of formal representation. Using our formalism we can formally define containment relationship ($\sqsubseteq$) over the set $S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$. If $\mathscr{C} = \{< d, A_d > | d \in \mathbb{D} \ \wedge \ A_d \subseteq \mathbb{A}\}$ and $\mathscr{C}' = \{< d', A'_d > | d' \in \mathbb{D} \ \wedge \ A'_d \subseteq \mathbb{A}\}$ are two Context Schemas in $S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$, we say that $\mathscr{C}' \sqsubseteq \mathscr{C}$ if $DIM(\mathscr{C}') \subseteq DIM(\mathscr{C})$, and $\forall\ d' \in \mathscr{C}' \bullet ATT(d', \mathscr{C}') \subseteq ATT(d', \mathscr{C})$. The relation $\sqsubseteq$ on the set $S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$ is a *partial order* because it is *reflexive*, *anti-symmetric*, and *transitive*. Hence, $\langle S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}), \sqsubseteq \rangle$ is a *partially ordered set* (poset).

Example 3. Consider the context schema $\mathscr{C}'_{conf} = \{Date : \{a_1\}, Time : \{a3\}\}$ defined on the same set of dimensions and attributes as in Example 1. Because $DIM(\mathscr{C}'_{conf}) \subseteq DIM(\mathscr{C}_{conf})$, and $ATT(d', \mathscr{C}'_{conf})) \subseteq ATT(d', \mathscr{C}_{conf}))$, for all $d' \in DIM(\mathscr{C}'_{conf})$ it follows that $\mathscr{C}'_{conf} \sqsubseteq \mathscr{C}_{conf}$.

3.1 Operations and Calculus

Based on containment relation $\sqsubseteq$ we define equality ($=$). For two context schemas $\mathscr{C}$ and $\mathscr{C}'$, if $\mathscr{C} \sqsubseteq \mathscr{C}'$, and $\mathscr{C}' \sqsubseteq \mathscr{C}$, then $\mathscr{C} = \mathscr{C}'$. In order to relate and deal with every pair of schemas in the poset $\langle S_{\mathscr{C}}(D, A), \sqsubseteq \rangle$ we introduce the two operators "join" ($\oplus$) and "meet" ($\odot$). The "join" of two context schemas produces the "smallest" context schema that contains those two context schemas. The "meet" of two schemas produces the "largest" context schema that is contained in those two schemas. These operations, when implemented as part of context tool kit in an application will enable "exporting knowledge and reasoning" across contexts. Formally,

$$\oplus : S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}) \times S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}) \rightarrow S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$$

$$\mathscr{C} \oplus \mathscr{C}' = \{< d'', A_{d''} > | d'' \in DIM(\mathscr{C}) \cup DIM(\mathscr{C}') \ \wedge$$
$$A_{d''} = \{ATT(d'', \mathscr{C}) \cup ATT(d'', \mathscr{C}')\} \quad \blacksquare$$

$$\odot : S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}) \times S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}) \rightarrow S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$$

$$\mathscr{C} \odot \mathscr{C}' = \{< d'', A_{d''} > | d'' \in DIM(\mathscr{C}) \cap DIM(\mathscr{C}') \ \wedge$$
$$A_{d''} = \{ATT(d'', \mathscr{C}) \cap ATT(d'', \mathscr{C}')\} \quad \blacksquare$$

It is easy to observe that for any three schemas $\mathscr{C}_1, \mathscr{C}_2, \mathscr{C}_3$ in $\{S_{\mathscr{C}}(D, A), \sqsubseteq\}$, $\oplus$ and $\odot$ operators satisfy *absorption*, *commutative*, *associative*, and *distributive* properties. In particular, $\oplus$ is distributive over $\odot$ and vise versa:

- $\mathscr{C}_1 \oplus (\mathscr{C}_2 \odot \mathscr{C}_3) = (\mathscr{C}_1 \oplus \mathscr{C}_2) \odot (\mathscr{C}_1 \oplus \mathscr{C}_3)$
- $\mathscr{C}_1 \odot (\mathscr{C}_2 \oplus \mathscr{C}_3) = (\mathscr{C}_1 \odot \mathscr{C}_2) \oplus (\mathscr{C}_1 \odot \mathscr{C}_3)$

Example 4. Applying the join and meet operations on the two conference context schemas $\mathscr{C}_{conf}$ and $\mathscr{C}'_{conf}$ in Examples 1 and 3 we get two new context schemas:

$$\mathscr{C}_{conf} \oplus \mathscr{C}'_{conf} = \{Date : \{a_1, a_2\}, Time : \{a_3\}, Location : \{a_4\}\}$$
$$\mathscr{C}_{conf} \odot \mathscr{C}'_{conf} = \{Date : \{a_1\}, Time : \{a_3\}\}$$

3.2 Context Schema Lattice

With the join and meet operations on the poset $\langle S_{\mathscr{C}}, \sqsubseteq \rangle$ the set $\mathscr{L} = (S_{\mathscr{C}}, \sqsubseteq, \oplus, \odot)$ is a lattice [15]. In particular, the lattice $\mathscr{L}$ is also *closed* and *distributive*. It is closed because its minimum element is the null context schema and the maximum element is the context schema composed with all dimensions in set $\mathbb{D}$ and all attributes of $\mathbb{A}$ associated across the dimensions. It is distributive because for $x, y, z \in \mathscr{L}$, we can verify the two properties: (1) $(x \odot y) \oplus (x \odot z) = x \odot (y \oplus z)$, and (2) $(x \oplus y) \odot (x \oplus z) = x \oplus (y \odot z)$. Hence $\mathscr{L}(\mathbb{D}, \mathbb{A}) = (S_{\mathscr{C}}(\mathbb{D}, \mathbb{A}), \oplus, \odot, \sqsubseteq)$ is a complete lattice. These properties have enormous consequence on the system level. First, when the set $(\mathbb{D}, \mathbb{A})$ is fixed, the set of all context schemas in the lattice structure is closed with respect to the join and meet operations. Consequently, no typed context and thus context instances (defined in the later sections) are "left unaccountable". That means, we have a "closed world" of families of context instances and the knowledge they enclose. This closed world property fulfills the requirement of "sufficient completeness of actions in contexts" for ensuring safety and privacy properties at system execution stage. The distributive property enables "simplification" of expressions that involve context instances and hence the evaluation of "predicates" that need to be evaluated at context instances. In the rest of the discussion we agree that $\mathbb{D}, \mathbb{A}$ is fixed and simply use the notation $S_{\mathscr{C}}$ for context schema and $\mathscr{L}$ for the complete lattice.

4 Tier 2: Typed Context Schema Representation and Calculus

Typed Context Schema (TCS) is a context schema in which attributes are associated with types. Each attribute is typed in the sense that it has a domain of values with respect to the type associated with it. Representation and calculus of schemas in TCS are inherited from the un-typed schemas defined in Tier 1.

4.1 Typed Context Schema Representation

Let $\mathbb{T} = \{T_1, T_2, ..., T_n\}$ denote a finite set of types such that for each type $T_x \in \mathbb{T}$ there exists a pair $<V_x, OP_x>$, where V_x is a "maximal" set of values, and OP_x denotes a set of operations allowed on the set V_x. By "maximal" we mean (1) V_x does not overlap with the set V_y of values of any other type $T_y \in \mathbb{T}$, and (2) if V'_x is any other set containing the values of T_x then $V'_x \subset V_x$. A type assignment to attributes in $\mathbb{A}$ is a mapping (a finite function) $M : \mathbb{A} \rightarrow \mathbb{T}$ that associates a unique type for every attribute in $\mathbb{A}$. We denote the set of all such type assignments by $\mathbb{A}^{\mathbb{T}}$. Two mappings $M_1, M_2 \in \mathbb{A}^{\mathbb{T}}$ are *equal* only if $M_1(a) = M_2(a)$ for every $a \in \mathbb{A}$. It is possible that two mappings M_1 and M_2 are different while they could have some common attributes. Formally, with respect to $M \in \mathbb{A}^{\mathbb{T}}$, the typed version of context schema $\mathscr{C}$, denoted as $\mathscr{C}^M$, is $\mathscr{C}^M = \{<d, A_d^M> | d \in DIM(\mathscr{C}), A_d^M = \{a^{M(a)} | a \in A_d\}\}$.

Example 5. Let $\mathbb{D} = \{d_1, d_2\}$, and $\mathbb{A} = \{a_1, a_2, a_3\}$. A context schema $\mathscr{C}_1$ over $\mathbb{D}$ and $\mathbb{A}$ is $\mathscr{C}_1 = \{d_1 : \{a_1, a_2\}, d_2 : \{a_2, a_3\}\}$. We emphasize that any standard or abstract data type may be associated to an attribute, and more than one attribute may have the same type. As an illustration, consider the set of types such that $\mathbb{T} = \{Int, Char, Range\}$ where Int and $Char$ are the standard Integer and Character types. Assume that $Range$ type has the set of operations $\{=, After, Before, Lowest, Highest\}$, and has the domain of values $[1 \cdots 10]$. Let $M_1 = \{a_1 \rightarrow Int, a_2 \rightarrow Char, a_3 \rightarrow Range\}$ and $M_2 = \{a_1 \rightarrow Int, a_2 \rightarrow Int, a_3 \rightarrow Char\}$ be two mappings. The typed attribute sets corresponding to these mappings are $\{a_1^{Int}, a_2^{Char}, a_3^{Range}\}$, and $\{a_1^{Int}, a_2^{Int}, a_3^{Char}\}$. Thus, the two distinct typed schemas are

$$\mathscr{C}_1^{M_1} = \{d_1 : \{a_1^{Int}, a_2^{Char}\}, d_2 : \{a_2^{Char}, a_3^{Range}\}\}$$

$$\mathscr{C}_1^{M_2} = \{d_1 : \{a_1^{Int}, a_2^{Int}\}, d_2 : \{a_2^{Int}, a_3^{Char}\}\}$$

Notice that $\mathscr{C}_1^{M_1} \neq \mathscr{C}_1^{M_2}$ because of the different type assignment M_1 and M_2.

Both schema calculus and lattice definition are extended to $S_{\mathscr{C}^M}$, the family of schemas that have the same type. Schema operations that are defined earlier are well-defined only within each family. That is, operations are not extended across families of typed schemas. Thus, the operations $\mathscr{C}_1^{M_1} \oplus \mathscr{C}_1^{M_2}$, $\mathscr{C}_1^{M_1} \odot \mathscr{C}_1^{M_2}$ are not defined since $M_1 \neq M_2$. However, for typed context $\mathscr{C}_3^{M_1} = \{d_2 : \{a_1^{Int}\}\}$ we can define $\mathscr{C}_1^{M_1} \oplus \mathscr{C}_3^{M_1}$ and $\mathscr{C}_1^{M_1} \odot \mathscr{C}_3^{M_1}$. The family of context schema $S_{\mathscr{C}^M}(\mathbb{D}, \mathbb{A})$ with join and meet operators is a complete typed lattice $\mathscr{L}(\mathbb{D}, \mathbb{A})^M$. This lattice includes all possible context schemas of type M. In addition to the schema operations it may be possible to define additional operations induced by the operators associated with type T induced by mapping M. We emphasize that if $\alpha = |\mathbb{A}^{\mathbb{T}}|$ then there are potentially α mappings, each associating every attribute of $\mathbb{A}$ to a type in $\mathbb{T}$. Consequently, from one schema $\mathscr{C} \in S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})$ we can generate α different typed schemas. Thus, for a fixed set of dimensions/attributes set (D, A) we can generate $\alpha \times \beta$ typed schemas where $\beta = |S_{\mathscr{C}}(\mathbb{D}, \mathbb{A})|$. As an illustration, for a set of dimensions $\mathbb{D} = \{d_1, d_2\}$ and a set of attributes

$\mathbb{A} = \{a_1, a_2\}$, let $\mathscr{L}$ be a context schema lattice that includes the set of context schemas $S_{\mathscr{C}} = \{\mathscr{C}_\phi, \ \{d_1 : \{a_1\}\}, \ \{d_2 : \{a_2\}\}, \ \{d_1 : \{a_1\}, d_2 : \{a_2\}\}\}$. Assuming that we have applied M_1 and M_2 on all contexts in $S_{\mathscr{C}}$ to construct the two distinct typed schema types $S_{\mathscr{C}^{M_1}}$ and $S_{\mathscr{C}^{M_2}}$. From these we can construct two lattices $\mathscr{L}^{M_1}$ and $\mathscr{L}^{M_2}$, which are different typed versions of the general context schema lattice $\mathscr{L}$.

5 Tier 3: Context Instance Representation and Calculus

In all practical applications [11,17] the term "context" has been used as "meta information" annotating "certain scenarios" or "happenings". This is achieved by associating "values" (also called "tags") to "dimensions". In our theory we arrive at such "contexts" as "instances" of context schemas. The rationale is to provide a more abstract foundation from which we can generate several "families" of context instances from each typed schema. The advantages include (1) levels of abstractions to conceptualize and manipulate schemas and instances, (2) provide a strong typing for attributes, and (3) achieve a potentially infinite number of context instances corresponding to each "family of typed contexts". This rigorous and disciplined theory will enable the correct development of context toolkit and promote reuse for ubiquitous computing applications.

5.1 Context Instance Representation

A Context Instance (CI) is an instantiated TCS in the sense that the attribute names in a TCS are substituted by values from the associated type domain. Hereafter we refer to context instance as context. To formalize, we start with one $\mathscr{C}^M$ be TCS of the lattice $S(\mathbb{D}, \mathbb{A})^M$, and use the substitution notation $[x/v]$ to mean that v is substituted for x. Let θ be a substitution function that assigns to each typed attribute a^T a value from the domain V of values associated with T. By a substitution $\theta : \mathscr{C}^M \rightarrow I_1(\mathscr{C}^M)$, we get an instance $I_\theta(\mathscr{C}^M)$ for the typed context schema $\mathscr{C}^M$, where $DIM(I_\theta(\mathscr{C}^M)) = DIM(\mathscr{C}^M)$, $\forall \ \ < d : A_d^T >\in \mathscr{C}^M, \exists \ \ < d : val^V >\in I_\theta(\mathscr{C}^M)$, $val^V = \{[a_i\theta/v_i] \mid v_i \in V\}$. That is, from each node (schema) in $\mathscr{L}(\mathbb{D}, \mathbb{A})$ we can generate a context instance for a fixed substitution. Because an attribute can be substituted by any value from its associated type domain, for each type assignment to an attribute we get a family of contexts generated from one node in $\mathscr{L}(\mathbb{D}, \mathbb{A})$. Therefore, in addition to schema operations we can introduce the operations of the associated type to contexts within a family.

Example 6. Let $\theta_1 = \{a_1/1, a2/b, a3/[1-3]\}$. By applying θ_1 on $\mathscr{C}_1^{M_1}$ in Example 5 we get one context $I_1(\mathscr{C}_1^{M_1}) = \{d_1 : [1, \text{“}b\text{”}], d_2 : [\text{“}b\text{”}, [1-3]]\}$. By applying the substitution $\theta_2 = \{a_1/100, a2/e, a3/[2-4]\}$ to $\mathscr{C}_1^{M_1}$, another instance is derived $I_2(\mathscr{C}_1^{M_1}) = \{d_1 : [100, \text{“}e\text{”}], d_2 : [\text{“}e\text{”}, [2-4]]\}$. When $\theta_3 = \{a_1/10, a_2/20, a_3/c\}$ is applied to $\mathscr{C}_1^{M_2}$ we get the new instance $I_3(\mathscr{C}_1^{M_2}) = \{d_1 : [10, 20], d_2 : [20, \text{“}c\text{”}]\}$ of type M_2. A noteworthy remark is that operational consistency exists only for contexts within each family.

6 Modeling Example

Many different adhoc notations for modeling contexts can be found in [18]. From among them we have chosen "paper submission context" that has been modeled graphically using contextual graphs [13]. This example has been also used in [12]. The actors in "paper submission" process are "publisher, editor, author, and reviewer". They share some activities like "canceling, submitting, checking". However, each actor is independent with respect to many other activities. For instance, editor can edit a paper, but cannot reject a paper, only a reviewer can. We show in the following steps how this context example can be formally represented using our 3-tier context representation. Through this case study the flexibility and generality provided in our model are highlighted (Fig. 1).

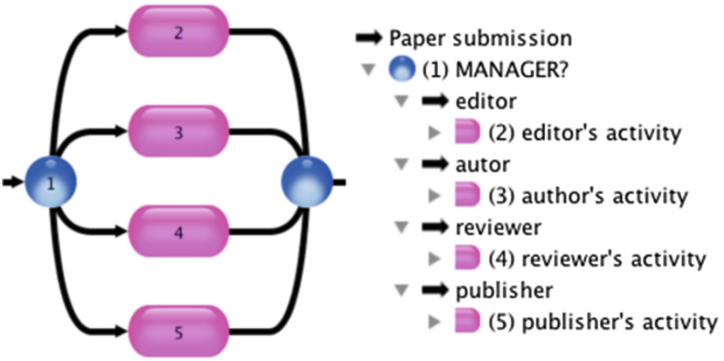

Fig. 1. Snapshot of the paper submission example in [13]

Step 1. We develop the context schema after identifying a set of dimensions, a set of attributes set, and a set of types from the graphical model. Each actor has information such as "name", "date-of-last-login", thus we find "Info" dimension is necessary. An actor also has "unique activities" and "common activities" shared among all types of actors. Therefore, the two dimensions "Unique", and "Common" should be included in our model. Therefore, the set of dimensions for our model is $\mathbb{D} = \{Info, Unique, Common\}$. The attributes of each dimension can be identified by the information that a dimension holds. "Info" dimension for instance, holds the "name", "lastlogindate" attributes. These attributes can simply be named anything, but meaningful names are used to make attributes readable and comprehensible. Based on example, each author has a list of unique activities, and a list of common activities. Therefore, both dimensions "Unique" and "Common" may have one or more attribute. For the sake of simplicity let us assume they have single attributes act_common, and act_unique. Thus, the set of attributes for our model is $\mathbb{A} = \{name, lastlogindate, act\_common, act\_unique\}$. The types can be customized/user defined, or can be basic types like integer and string. The information in our attributes are either string, integer,

or list of a generic type. Therefore, we define the set of types $\mathbb{T} = \{string, date, common, editor, reviewer, publisher, author\}$. The domain of values and operations for each type can be defined: (1) *string*: operations:$\{concat, intersect\}$, domain: set of alphanumeric values; (2) *integer*: operations: $\{+, -\}$, domain: (*date*); (3) *common* : operations: $\{before, after, add, remove\}$, domain: list of common activities; (4) *editor* : operations: $\{before, after, add, remove\}$, domain: list of unique activities for editor; (5) *reviewer* : operations: $\{before, after, add, remove\}$, domain: list of unique activities for reviewer; (6) *publisher* : operations: $\{before, after, add, remove\}$, domain: list of unique activities for publisher; (7) *author* : operations: $\{before, after, add, remove\}$, domain: list of unique activities for author. Based on the above choice the "Paper Submission Context Schema" is defined as

$$\mathscr{C}_{PS} = \{Info : \{name, lastlogindate\}, Unique : \{activities\}, Common : \{activities\}\}.$$

Step 2. We assign types to attributes and generate typed schemas. The above schema is general to all types of actors. The types are assigned as below:

$$\mathscr{C}_{PS}^{edi} = \{Info : \{name^{string}, lastlogindate^{date}\}, Unique : \{activities^{editor}\}, Common : \{activities^{common}\}\}$$
$$\mathscr{C}_{PS}^{rev} = \{Info : \{name^{string}, lastlogindate^{date}\}, Unique : \{activities^{reviewer}\}, Common : \{activities^{common}\}\}$$
$$\mathscr{C}_{PS}^{pub} = \{Info : \{name^{string}, lastlogindate^{date}\}, Unique : \{activities^{publisher}\}, Common : \{activities^{common}\}\}$$
$$\mathscr{C}_{PS}^{aut} = \{Info : \{name^{string}, lastlogindate^{date}\}, Unique : \{activities^{author}\}, Common : \{activities^{common}\}\}$$

Step 3. We generate context instances from the typed schema. In this level values are "substituted" to attributes. For instance, an editor instance can be $\mathscr{C}_{PS}^{edi} = \{Info : \{\text{“}Jhon\ Dazer\text{”}, \text{“}2017 - 05 - 04\text{”}\}, Common : \{[submit, cancel, delete]\}, Unique : \{[edit, submit\_notes, contact\_publisher]\}$. We can generate a family of instances for each type assignment.

7 Conclusion

The three tier approach introduced in this paper generalizes previous attempts for dimension-based context representation [3,25,26], and enriches context calculus. Our approach is general, expressive, and flexible. Whereas the three tiers provide generality through abstraction levels, within tiers 2 and 3 we achieve regularity and extensionality. For certain types, such as "categorical types", categories can be incrementally defined in order to suit the needs of specific application in tier 3. We achieve generating a large number of specialized contexts and

can keep track of them using the family structure. Consequently, context calculus remain formal and correct with respect to operations associated with types. It promotes a disciplined approach to generating contexts that are correct with respect to domain semantics, because the choice of dimensions, attributes, and their types are essentially guided by domain semantics. As illustrated in Sect. 6 our formal approach can generate all meaningful contexts, whereas in informal approaches, such as graphical notation, it is possible to identify contexts that are not meaningful. Consequently context calculus implementation based on our formalism will be type-correct, leading to dependable toolkit that can be reused. We are currently working to explore of the algebraic structure of context families, as well as integrating contexts with Datalog programs for contextual reasoning. We are currently working on a reasoning system that will serve to infer new facts from a knowledge-base systems, as well as prove formally whether or not certain critical behavior is satisfied by context-awareness in pervasive computing applications.

References

1. Akman, V., Surav, M.: The use of situation theory in context modeling. Comput. Intell. Int. J. **13**(3), 427–438 (1997)
2. Alaga, V., Wan, K.: Context based enforcement of authorization for privacy and security in identity management. In: de Leeuw, E., Fischer-Hübner, S., Tseng, J., Borking, J. (eds.) Policies and Research in Identity Management. The International Federation for Information Processing, vol. 261, pp. 25–37. Springer, Boston (2008). https://doi.org/10.1007/978-0-387-77996-6_3
3. Alagar, V., Mohammad, M., Wan, K., Hnaide, S.A.: A framework for developing context-aware systems. EAI Endorsed Trans. Context-Aware Syst. Appl. **14**(1) (2014). https://doi.org/10.4108/casa.1.1.e2
4. Brèzillon, P., Gonzalez, A.I.: Context in Computing: A Cross-Disciplinary Approach to Modeling Real World. Springer, Berlin (2014)
5. Brézillon, P.: Context in human-machine problem solving: a survey. LIP **6**(1996), 029 (1996)
6. Carnap, R.: Meaning and Necessity. Chicago University Press, Chicago (1947). Enlarged Edition 1956
7. Clark, H.H., Carlson, T.B.: Context for comprehension. In: Attention and Performance, pp. 313–330. Lawrence Erlbaum Associates, Hillside (1981)
8. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. **16**, 97–161 (2001)
9. Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)
10. Dowley, D., Wall, R., Peters, S.: Introduction to Montague Semantics. Reidel Publishing Company, Amsterdam (1981)
11. Bettini, C., et al.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**, 161–180 (2009)
12. García, K., Brézillon, P.: A contextual model of turns for group work. In: Christiansen, H., Stojanovic, I., Papadopoulos, G.A. (eds.) CONTEXT 2015. LNCS (LNAI), vol. 9405, pp. 243–256. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25591-0_18

13. García, K., Brézillon, P.: Contextual graphs for modeling group interaction. In: Brézillon, P., Turner, R., Penco, C. (eds.) CONTEXT 2017. LNCS (LNAI), vol. 10257, pp. 151–164. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57837-8_12
14. Giunchiglia, F.: Contextual reasoning. Epistemologia, special issue on I Linguaggi e le Macchine **16**, 345–364 (1993)
15. Grätzer, S.: Lattice Theory: First Concepts and Distributive Lattices. W. H. Freeman, San Francisco (1971)
16. Guha, R.V.: Contexts: A Formalization and Some Applications, vol. 101. Stanford University Stanford (1991)
17. Held, A., Buchholz, S., Schill, A.: Modeling of context information for pervasive computing applications. In: Proceedings of SCI, pp. 167–180 (2002)
18. Interdisciplinary and Internal Conference Series. Modeling and using context (1997)
19. McCarthy, J.: Notes on formalizing context (1993)
20. McCarthy, J., Buvac, S.: Formalizing context (expanded notes) (1997)
21. Sato, M., Sakurai, T., Kameyama, Y.: A simply typed context calculus with first-class environments. In: Proceedings of FLOPs 2001: the 5th International Symposium on Functional and Logic Programming, pp. 359–374 (2001)
22. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: 1994 First Workshop on Mobile Computing Systems and Applications, WMCSA 1994, pp. 85–90. IEEE (1994)
23. Shoham, Y.: Varieties of context. In: Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, pp. 393–408 (1991)
24. Wan, K., Alagar, V., Paquet, J.: An architecture for developing context-aware systems. In: Roth-Berghofer, T.R., Schulz, S., Leake, D.B. (eds.) MRC 2005. LNCS (LNAI), vol. 3946, pp. 48–61. Springer, Heidelberg (2006). https://doi.org/10.1007/11740674_4
25. Wan, K.: Lucx: lucid enriched with context. Ph.D. thesis, Concordia University (2006)
26. Wan, K., Alagar, V., Paquet, J.: A context theory for intensional programming. In: Workshop on Context Representation and Reasoning (CRR05). Citeseer, Paris, July 2005
27. Weyhrauch, R.: Prolegomena to a theory of mechanized formal reasoning. Artif. Intell. **13**, 133–170 (1980)
28. Winograd, T.: Architecture for context. Hum.-Comput. Inter. **16**, 401–419 (2001)