Towards Ontology-Based Formal Verification Methods for Context Aware Systems*

Hedda R. Schmidtke and Woontack Woo**

GIST U-VR Lab., 500-712 Gwangju, South Korea
{schmidtk,wwoo}@gist.ac.kr

Abstract. Pervasive computing systems work within, and rely on, a model of the environment they operate in. In this respect, pervasive computing systems differ from other distributed and mobile computing systems, and require new verification methods. A range of methods and tools exist for verifying distributed and mobile concurrent systems, and for checking consistency of ontology-based context models. As a tool for verifying current pervasive computing systems both are not optimal, since the former cover mainly tree-based location models, whereas the latter are not able to address the dynamic aspects of computing systems. We propose to formally describe pervasive computing systems as distributed concurrent systems operating on the background of a mereotopological context model.

Keywords: context modelling, mereotopology, program verification, ontologies.

1 Introduction

Pervasive computing systems can be understood as distributed and mobile concurrent computing systems that are able to react flexibly to changes in their physical environment [30]. A model of the environment is therefore a fundamental part of a pervasive computing system, and research on context modelling methodology has led to novel data structures and ontologies for representing numerous aspects of context, such as location, time, social structure, computational structure, and generally the physical properties of the environment. However, current approaches to verification of pervasive computing systems [4, 27] based on Ambient Calculus [6] or the theory of Bigraphs [24] are focussed on tree-based location models, which are inappropriate to represent overlapping contexts, continuous domains of context, and continuous change. Ranganathan and Campbell [27] concluded that program verification and verification of context models are complementary tasks. However, important interactions exist and properties at the interface, where a process queries the context model or uses it to communicate information to other processes, should be verifiable [27].

* This research is supported by the UCN Project, the MIC 21st Century Frontier R&D Program in Korea.

** Corresponding author.

H. Tokuda et al. (Eds.): Pervasive 2009, LNCS 5538, pp. 309–326, 2009.

We propose a method for verification of pervasive computing systems with complex context models. We follow the suggestion of Ranganathan and Campbell [27] to separate verification of the context model from verification of the control structure. We argue that the semantics of context models can be described adequately within the theory of mereotopology [37], which is used widely in the area of formal ontology for describing domains such as space, time, partonomies, and taxonomies. We show that mereotopology is suitable to describe lattices, hierarchical structures that are not trees, and continuous domains, such as ranges of sensor values and uncertainty regions around GPS coordinates. On this background, we outline how querying of a context model and communication via activation of contexts in the context model can be added as new primitives to the semantics of a programming language. We thus separate the context model from the state of a program, so that constraints on both parts can be verified separately. The resulting simple example language is already expressive enough to allow specification of relevant distributed and mobile concurrent algorithms that operate with hybrid location models and are triggered by sensors.

The article consists of three main sections. In the next section (Sect. 2), we analyse related works on the representation of context from the areas of ubiquitous computing, formal verification methods for context-aware computing, and from the areas of knowledge representation and ontologies. Then, we outline a simple variant of the theory of mereotopology and describe how the theory applies to the problem of context modelling (Sect. 3). We show in Sect. 4 how this language can be combined with a classical CSP-style programming language.

2 Related Works

Ubiquitous computing became possible after distributed systems and mobile computing, in particular, the idea of wireless ad hoc networks, had been developed [30]. Accordingly, location is one of the best understood and most important parameters of context. The active map of Schilit and Theimer [31] organises locations of users and objects in a containment hierarchy of names for regions organised as a tree. Hybrid location models [20, 22] in addition allow for adding coordinate information into the hierarchical structure. The hierarchical structure is realised in [20, 31] with a tree data structure, and in [22] with a more general lattice structure that in contrast to a tree structure also allows that regions share sub-regions, that is, that regions can overlap. However, Leonhardt [22] demands that the leaf nodes of the hierarchical structure, the so-called zones, may not overlap; the zones thus provide a partitioning of space. A lattice structure based on partitions has also been suggested by Ye et al. [40]. Schmidtke and Woo [34] show that a partitioning of space can lead to problems: it imposes restrictions on the representation, leads to inflated hierarchies, and does not allow to properly reflect uncertainty of location information. We discuss in Sect. 3 how lattice-based location models can be described within the formal framework of mereotopology.

A range of approaches for verifying mobile systems that contain a notion of location has been suggested in the area of program verification, in particular approaches building upon *pi-calculus* [17], and the theories of *bigraphs* [4, 24], and *ambients* [6]. However, the notion of location or domains employed in these approaches is limited to tree-hierarchies [6] or allows for general graph structures [17]. Milner [24] suggests to combine a hierarchical tree-structure of localities with a global, unrestricted network structure. From the perspectives of knowledge representation [34] and context modelling [40], however, a tree structure is not sufficient for specifying location: in tree-based hierarchies, overlapping regions are not allowed and have to be split, and movement of an agent from one region to a neighbouring region is always discrete. Consequently, Ranganathan and Campbell [27] argue that proof of correctness with ambients [6] needs to be complemented with a proof of correctness of the context modelling parts of a pervasive computing environments, for which they use an ontology described in first-order logic [28]. However, it is not clear how the location model in an ambient specification should be made consistent with the more refined location model in an ontology [27].

We follow the suggestion in [27] and propose a coupled semantics of context modelling and programs, so that inconsistencies and unnecessary complexity can be avoided. Instead of using one of the above mentioned verification methods, which come with an in-built location model, we start from a simple textbook variant [2] of CSP as a widely familiar program verification language and complement it with a context modelling language [35] that allows for the description of contemporary lattice-based location models. The location model in our approach is maintained exclusively by the context model, or *context knowledge base*. The knowledge base is verified via a mereotopological, first-order logic theory of context aspects, that is, as an ontology, but not as a computing system. Our aim is to characterise a formal system that describes a broad range of existing context-aware pervasive computing systems in both their context modelling and context adaptation functionality.

With respect to context adaptation, two types of control structures for processing of contextual information have been distinguished [18]: *triggering* of program code is needed to start or activate applications in response to changes of context; and *branching* of program code is needed to let applications produce different behaviour depending on content of the context model at a given time. Dey [10] realised adaptation with a type-system and a condition-based publish-subscribe mechanism. We show in Sect. 4.2 how such conditions [10] and context dependent *branching* and *triggering* [18] of processes can be formalised.

Ontologies have been suggested as the key technology for adding semantics to applications. Ontologies for space and time have received considerable interest in the areas of formal ontologies and qualitative reasoning. One of the predominant approaches in these areas is mereotopology [37], with successful applications in robotics, geographical information science, the biosciences, and even for motion tracking. Surprisingly, it has not yet received much attention in pervasive computing or ambient intelligence [8, 16]. In contrast to point-set topology, which characterises neighbourhoods in a continuous domain, such as spatial regions or

temporal intervals, based on sets of extension-less points, mereotopology starts from extended portions of the world [37], that is, from the neighbourhoods or contexts themselves. Being independent from the number of dimensions of a domain, mereotopology cannot only be applied to describe spatial contexts but also temporal contexts and sensor value ranges, and can also be used to model discrete domains as well as concept hierarchies and collections [5].

The primitive relations in mereotopology are that of part-hood ($\sqsubseteq$) and connection (C) between regions, replacing the set-theoretic notion of membership ($\in$) between points and sets of points. The theory dates back to works on point-free geometry by Whitehead, Tarski, and Clarke [37]. It has been applied for formal ontologies in information systems by Randell et al. [26] and many others; a recent overview has been given by Varzi [37]. The idea to move away from point-sets to more meaningful primitive extended entities has not only been successful for the spatial domain but also for the temporal domain [1]. Moreover, results of Galton [14] suggest that concepts of qualitative reasoning, which make the mereotopological calculi attractive for applications in artificial intelligence [9], can be generalised, so as to cover the broad range of sensors employed for establishing context awareness [32]. We illustrate (Sect. 3.2) that concepts of mereotopology can be used to describe continuous quantitative domains of sensor values, in particular when uncertainty [7, 33] or privacy requirements [11, 29] are important. In these cases, sensory information may not be given by exact values or coordinates, but only in the form of intervals or uncertainty regions.

An important aspect in pervasive computing is how to identify and address specific entities (objects, places, users, services, times etc.) in the environment. One way to do this is to assign unique identifiers to every entity, for instance by attaching bar codes or RFID, and to give a web presence to people, places, and things. This approach poses considerable challenges for privacy, since users and their smart objects cannot remain anonymous, and it can also lead to inefficiency [3] caused by unnecessarily lengthy descriptions. A more context-oriented approach, which respects the privacy principle of locality [21] better, is to connect smart objects based on their current context [3]. Our model allows to formally capture such novel address methods. In general, a process in our model activates a context, whose activation is then transmitted either upward or downward to all processes that are registered with super-contexts or sub-contexts, respectively (Sect. 4). We can thus model contemporary methods of inter-process communication, such as *semantic triggering* of a processes when a change of state is detected, and *semantic broadcasts*, e.g., broadcasts over a certain spatial area, to a certain community, or to an otherwise anonymous recipient who has certain properties. Communication via IDs can be modelled realistically as a broadcast to a very specific context; one could then reason about whether a certain protocol can ensure formally that only a unique listener is addressed.

It has often been stated [28, 36, 38, 39] that modelling context with OWL, the de facto standard for ontologies, would not be possible. A main difficulty is that OWL (version 1.0) does not support to describe a relation as a partial ordering relation, which would be required to characterise location hierarchies

such as that of Ye et al. [40], other containment hierarchies, and ordering of sensor values, since reflexivity and antisymmetry of relations cannot be described in the description logic $\mathcal{SHOIQ}$ underlying OWL-DL. The solution chosen by many approaches to ontology-based context modelling is to use expressive logical languages, such as F-Logic [36] or first order logic [28, 38], which come at the price of less efficient reasoning. An alternative is the context-oriented logical language (for brevity called CL in this article) proposed in [35], which we use in this article as a context modelling language (Sect. 4). Although in several ways less expressive than OWL, CL supports hierarchical reasoning over multiple domains. The syntax is similar to that of first order logic, and it can easily be embedded into the first order logic needed to characterise mereotopological structures. CL is thus compatible with general first order logic ontology frameworks containing mereotopological notions, such as SUMO [25].

3 Mereotopology as a Theory for Context Modelling

In this section, we describe a mereotopological theory that captures notions of context found in the literature on context modelling and location modelling for mobile, ubiquitous, and pervasive computing. The general theory is given in Sect. 3.1 in the form of a domain-independent first-order logic axiomatisation. For illustration purposes, examples from the spatial domain and location modelling are used in this section. Domains beyond location are the topic of Sect. 3.2, where we introduce a more general first order language for context modelling and illustrate its applicability with multi-domain examples of context modelling. In Sect. 4 we outline the more tractable language CL [35], which serves as a sub-language for describing context and querying context models in a simple context-aware programming language.

3.1 The Language of Mereotopology

The framework of mereotopology in this section mainly follows the explications given by Varzi [37]. However, our focus is less on philosophical questions of ontology than on specifying properties underlying contemporary context models.[1] We assume the standard basic language of first order predicate logic with identity ($=$) as given.[2]

[1] We also deviate in syntax: where Varzi [37] – like many other authors in the field of mereotopology – uses prefix notation, we use mainly infix notation, for instance: we write $x \sqsubseteq y$ instead of Pxy for expressing that x is part of y.

[2] For increasing readability of formulae we introduce rules for saving brackets. The following precedence of logical connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and term connectives $\sim, \sqcap, \sqcup$ is assumed. Quantifiers $\forall$ and $\exists$ are to be read as having maximal scope, that is, until the first bracket closing an opening bracket before the quantifier or until the end of the formula. We highlight the boundaries of atomic formulae, such as $[x_1 \sqsubseteq x_2]$, with square brackets, so that complex terms and complex formulae are separated clearly. Sentences are numbered with respect to their function: axioms are indicated with the prefix A, definitions with D and examples with E.

Parts. The fundamental notion of mereotopology is the mereological primitive relation of *part-hood* ($\sqsubseteq$). In location modelling, the relation $\sqsubseteq$ corresponds to spatial containment and the entities that are ordered by $\sqsubseteq$ can be understood as spatial contexts or regions, whether obtained via symbolic location sensing or coordinate location sensing.

The relation $\sqsubseteq$ is characterised axiomatically as a partial ordering relation, that is, as reflexive, transitive, and antisymmetric. Reflexivity states that each x is a part of itself (A1). Transitivity demands that if x_1 is part of x_2, and x_2 is part of x_3, then x_1 is also part of x_3 (A2). Antisymmetry establishes the ordering: if x_1 is part of x_2, and x_2 is part of x_1, then they must be identical (A3).

$$\forall x : [x \sqsubseteq x] \tag{A1}$$

$$\forall x_1, x_2, x_3 : [x_1 \sqsubseteq x_2] \wedge [x_2 \sqsubseteq x_3] \rightarrow [x_1 \sqsubseteq x_3] \tag{A2}$$

$$\forall x_1, x_2 : [x_1 \sqsubseteq x_2] \wedge [x_2 \sqsubseteq x_1] \rightarrow [x_1 = x_2] \tag{A3}$$

$$[x_1 \sqsubset x_2] \overset{def}{\Leftrightarrow} [x_1 \sqsubseteq x_2] \wedge \neg [x_2 \sqsubseteq x_1] \tag{D1}$$

From the relation $\sqsubseteq$, we can define a strict variant: a *proper part* x_1 of an entity x_2 (relation symbol: $\sqsubset$) is a part that does not contain x_2 as a part (D1).

Overlap and Underlap. We further define two partial functions: for two entities x_1 and x_2, the smallest entity that contains both is called the *sum* (D2), and the largest entity which they both contain is called here the *intersection*[3] (D3).

$$[x_1 \sqcup x_2 = x] \overset{def}{\Leftrightarrow} [x_1 \sqsubseteq x] \wedge [x_2 \sqsubseteq x] \wedge \forall y : [y \sqsubseteq x] \wedge [x_1 \sqsubseteq y] \wedge [x_2 \sqsubseteq y] \rightarrow [y = x] \tag{D2}$$

$$[x_1 \sqcap x_2 = x] \overset{def}{\Leftrightarrow} [x \sqsubseteq x_1] \wedge [x \sqsubseteq x_2] \wedge \forall y : [x \sqsubseteq y] \wedge [y \sqsubseteq x_2] \wedge [y \sqsubseteq x_1] \rightarrow [y = x] \tag{D3}$$

$$\forall x : [\bot \sqsubseteq x] \wedge [x \sqsubseteq \top] \tag{A4}$$

With respect to common sense spatial intuition, these two functions are partial functions, since there can be no intersection between regions that do not share a part, and there can be no sum of two regions if there is no region that contains them both. It is one of the benefits of mereotopology from a philosophical point of view that it does not require arbitrary sums and intersections to exist. However for a lattice structure such as demanded for the hybrid location models [22, 40], the existence of arbitrary sums and intersections of regions has to be ensured. A simple way to achieve this is to introduce an empty region $\bot$ that is part of every region and a region $\top$ extending over the complete domain (A4).

With these special regions introduced, the relations of overlap and underlap can be defined: two regions *overlap* if and only if they share a part that is not

[3] We use the term intersection here as a mnemonic and reference to spatial intuition. The mathematically more appropriate term used by Varzi [37] is *product*.

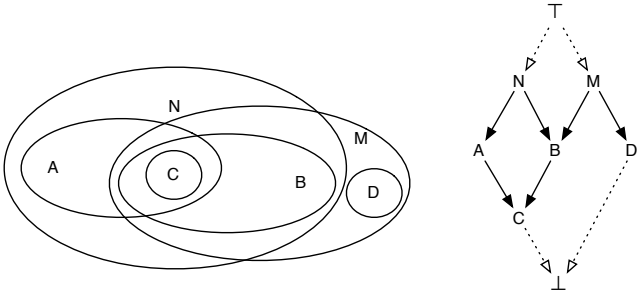

Fig. 1. Overlap and underlap in a location hierarchy: A and B have C as a common part and are both part of the region N

identical to the empty region (D4); and two regions *underlap* if and only if they are both part of a region that is not identical to the whole domain (D5).

$$[x_1 \,\mathrm{O}\, x_2] \overset{def}{\Leftrightarrow} \exists x : \neg[x = \bot] \wedge [x \sqsubseteq x_1] \wedge [x \sqsubseteq x_2] \tag{D4}$$

$$[x_1 \,\mathrm{Y}\, x_2] \overset{def}{\Leftrightarrow} \exists x : \neg[x = \top] \wedge [x_1 \sqsubseteq x] \wedge [x_2 \sqsubseteq x] \tag{D5}$$

$$[\sim x = y] \overset{def}{\Leftrightarrow} [x \sqcup y = \top] \wedge [x \sqcap y = \bot] \tag{D6}$$

Additionally, we can define a complement operation ($\sim$): the region y is the complement of x if and only if the sum of x and y is identical to the whole domain $\top$, and the intersection is identical to the empty region $\bot$ (D6).

In location models, the relation of overlap can be used to span accessibility graphs: if x overlaps y, then it is possible to go from x to y [40], or to enter y from x [28]. For a finite set of regions in a location model, the graph of the overlap relation already allows us to do path planning: if the hallway overlaps with Alice's office and Bob's office, then we can go from Alice's office to Bob's office via the hallway. The notions of overlap and underlap are crucial for modelling such notions of reachability in a context hierarchy (Fig. 1).

Connection. The above specification is sufficient to characterise containment hierarchies and we can already specify two regions as being connected by overlapping in a shared sub-region. For location modelling [34, 40] however, it can be more convenient to use a weaker form of connection that does not require the modeller to be committed to asserting that there is an overlap region, for instance: we might specify two roads as being connected without being committed to asserting that there is an overlap region; or we might want to contrast viable and relevant *overlap*, such as a monitored doorway region shared by a hallway and an office, with *external connection* as sharing of boundary parts, such as two offices separated by a shared wall.

However, topological notions such as boundary, interior, or closure cannot yet be expressed. At first glance, these notions seem to require a point-set perspective on regions. However, a purely region-based characterisation can be obtained if a relation of *connection* (C) between regions is introduced. Connection is

characterised as a reflexive and symmetric relation: every non-empty region is connected to itself (A5); and if x_1 is connected to x_2, then also x_2 is connected to x_1 (A6):

$$\forall x : \neg[x = \bot] \rightarrow [x \,\mathrm{C}\, x] \tag{A5}$$

$$\forall x_1, x_2 : [x_1 \,\mathrm{C}\, x_2] \rightarrow [x_2 \,\mathrm{C}\, x_1] \tag{A6}$$

$$\forall x_1, x_2 : [x_1 \sqsubseteq x_2] \rightarrow \forall y : [y \,\mathrm{C}\, x_1] \rightarrow [y \,\mathrm{C}\, x_2] \tag{A7}$$

Additionally, we need a bridging principle [37] between the mereological parts ($\sqsubseteq$) and the topological parts (C) of the theory. Following Varzi [37], we demand that C is monotonous with respect to $\sqsubseteq$: if x_1 is part of x_2, then any region y connected to x_1 is also connected to x_2 (A7). We can then define the well-known RCC relations [8, 26]: *external connection* (*EC*) is connection without overlap (D7); a *non-tangential proper part* (*NTPP*) is a part x of y that is only connected to regions z that overlap y (D8); and a *tangential proper part* (*TPP*) is a proper part that is not *NTPP* (D9).

$$[x \,\mathrm{EC}\, y] \overset{def}{\Leftrightarrow} [x \,\mathrm{C}\, y] \wedge \neg[x \,\mathrm{O}\, y] \tag{D7}$$

$$[x \,\mathrm{NTPP}\, y] \overset{def}{\Leftrightarrow} [x \sqsubset y] \wedge \forall z : [x \,\mathrm{C}\, z] \rightarrow [y \,\mathrm{O}\, z] \tag{D8}$$

$$[x \,\mathrm{TPP}\, y] \overset{def}{\Leftrightarrow} [x \sqsubset y] \wedge \neg[x \,\mathrm{NTPP}\, y] \tag{D9}$$

It is clear that O fulfils all three axioms of C, but if we set C to be identical to O, then the definitions of *EC* and *TPP* become empty and *NTPP* becomes identical to $\sqsubset$. In current location models, overlap and connection are not yet distinguished.

The idea of a distinction between overlap, as sharing of parts, and connection, as sharing of something that is not necessarily a relevant element of the domain, has been generalised in theories of granularity [12, 34]. As context models can easily become very large, modelling granularity is crucial to ensure scalability of hierarchical context models, cf. [33] for an approach applicable to mereotopologically specified context models.

3.2 Beyond Location

The above discussion mentioned mainly location models, and the notion of context is conceived broader in pervasive computing [32]. However, key properties for context modelling in many other domains are similar hierarchical structures and a related notion of connection. Mereotopology is a general domain-independent mathematical theory that can describe such structures. We demonstrate this claim by presenting hierarchical structures in five other domains.

Following Jang et al. [19], a representation of context should be expressive enough to answer at least the six questions: *who* caused a certain interaction with *what where when how* and *why*. Following this mnemonic, we can identify five domains with corresponding relations, besides spatial containment and spatial overlap, which are in the following individuated with symbols $\sqsubseteq_{\mathrm{where}}$ and $\mathrm{O}_{\mathrm{where}}$.

We need to model at least: *times* with a relation of temporal containment $\sqsubseteq_{\text{when}}$ of intervals, *groups of agents* with a containment relation between groups, *classes of objects* with a taxonomic subclass relation $\sqsubseteq_{\text{what}}$, *states*, including measured or inferred states of the environment, structured by an embedded logical entailment relation $\sqsubseteq_{\text{how}}$, and *events*, including externally available commands and actions of a system, partially ordered by a causation relation $\sqsubseteq_{\text{why}}$.

The contexts we mostly want to reason about have a meaningful extension in more than one of the six dimensions; a conference, for instance, has a time, location, and participants. We employ combined relations of general sub-contexts (D10) and partial sub-contexts (D11):

$$[x \sqsubseteq_\forall y] \overset{def}{\Leftrightarrow} [x \sqsubseteq_{\text{when}} y] \wedge [x \sqsubseteq_{\text{where}} y] \wedge [x \sqsubseteq_{\text{who}} y] \wedge [x \sqsubseteq_{\text{what}} y] \wedge [x \sqsubseteq_{\text{how}} y] \wedge [x \sqsubseteq_{\text{why}} y] \quad \text{(D10)}$$

$$[x \sqsubseteq_\exists y] \overset{def}{\Leftrightarrow} [x \sqsubseteq_{\text{when}} y] \vee [x \sqsubseteq_{\text{where}} y] \vee [x \sqsubseteq_{\text{who}} y] \vee [x \sqsubseteq_{\text{what}} y] \vee [x \sqsubseteq_{\text{how}} y] \vee [x \sqsubseteq_{\text{why}} y] \quad \text{(D11)}$$

$$[x =_m y] \overset{def}{\Leftrightarrow} [x \sqsubseteq_m y] \wedge [y \sqsubseteq_m x] \quad \text{(D12)}$$

The axioms of Sect. 3.1 hold for all six relations $\sqsubseteq_m$ if we replace identity ($=$) with appropriate equivalence relations $=_m$ (D12). Note that our limitation to a fixed set of relations allows us to avoid distinctions between different types of entities in the language and facilitates reification. The only types of entities we employ are *contexts* as portions of the world around us that can have an extension in one or more of the six dimensions.[4]

Temporal Containment and Ordering. The extension from spatial containment to temporal containment is straightforward if we consider time as a one-dimensional space. We can express with a relation of temporal containment that one context is during the time of another ($\sqsubseteq_{\text{when}}$), that contexts are synchronous ($=_{\text{when}}$), or temporally overlapping (O_{when}). Basic time conflicts can thus be represented:

$$[\text{MeetingA} \sqsubseteq_{\text{when}} \text{Aug/1/2008}] \wedge [\text{Aug/1/2008} \sqsubseteq_{\text{when}} \text{ConfB}] \wedge [\text{MeetingA} \sqsubseteq_{\text{where}} \text{CityA}] \wedge [\text{ConfB} \sqsubseteq_{\text{where}} \text{CityB}] \wedge \neg[\text{CityA } \text{O}_{\text{where}} \text{ CityB}] \quad \text{(E1)}$$

$$\forall c : [c \sqsubseteq_\forall \text{ConfB}] \rightarrow \neg[c \sqsubseteq_\forall \text{MeetingA}] \quad \text{(E2)}$$

If a calendar contains an entry for a meeting A on the day *Aug/1/2008*, and this day is during the time of a conference B; and A and B take place in two different, not overlapping city regions (E1), then we can conclude that any context c that is a proper sub-context of B cannot be a proper sub-context of A (E2).

[4] With a more classical AI representation format, we can conceive, for instance, $[a \sqsubseteq_{\text{when}} b] \wedge [a \sqsubseteq_{\text{where}} b]$ to be an abbreviation of $[\text{time}(a) \sqsubseteq_{\text{when}} \text{time}(b)] \wedge [\text{loc}(a) \sqsubseteq_{\text{where}} \text{loc}(b)]$, where the relations $\sqsubseteq_{\text{when}}$ and $\sqsubseteq_{\text{where}}$ are relations restricted to the types of times and regions, respectively. However, the operators $\sqcap$ and $\sqcup$ would also have to be replaced by functions $\sqcap_m$ and $\sqcup_m$ yielding the correct types.

In comparison to the interval-based calculus of time proposed by Allen [1], our notion of intervals includes not only convex intervals but also arbitrary sums of intervals, such as generalised intervals [23] and periodically recurring times, since we allow arbitrary sums and intersections. The directedness of time cannot be represented with temporal containment alone; but, given the relation $\sqsubseteq_{\text{why}}$ denoting causation, partial temporal ordering can be derived (E3):

$$[x \,\text{before}\, y] \leftrightarrow \neg[x \; \text{O}_{\text{when}} \; y] \wedge [x \sqsubseteq_{\text{why}} y] \tag{E3}$$

The relation $\sqsubseteq_{\text{why}}$ orders the domain of events, such as items in a plan or other program-like structures provided by users as required for anticipating situations and producing pro-active behaviour.[5] However, additional axioms would be needed to reason about relations between temporal structure and causation, in particular, a distinction between event tokens and event types, cf. Galton [15].

Logical Relations between States. In context modelling, important notions of states include status information, such as "on vacation," and sensory data, such as data from physiological or temperature sensors "temperature is higher than 25°C." States can be understood as reified propositions of an embedded logic, similar to a monadic predicate logic. A relation $\sqsubseteq_{\text{how}}$, which has the properties of a mereological containment relation, is the relation of implication between such formulae. For states, the operators $\sqcap$ and $\sqcup$ can be understood as conjunction and disjunction operators of the reified logic. An expression such as (E4) can be used for specifying the state of a context called *Measured*, reduced to the aspect of temperature, as being in the interval [25,30]. A corresponding formula in monadic predicate logic would be (E5).

$$[\text{Measured} \sqcap \text{Celsius} \sqsubseteq_{\text{how}} \geq 25 \sqcap \leq 30] \tag{E4}$$

$$\forall x : \text{Measured}(x) \wedge \text{Celsius}(x) \rightarrow \geq 25(x) \wedge \leq 30(x) \tag{E5}$$

We specify the computation of numerical comparisons, such as ≥ 25 in Sect. 4.2. By combining states and times, we can represent a state *Temp* changing over time:

$$\forall x : [x =_{\text{when}} t_1] \rightarrow [x \sqcap \text{Temp} \sqsubseteq_{\text{how}} \text{Cold}] \tag{E6}$$

$$\forall x : [x =_{\text{when}} t_2] \rightarrow [x \sqcap \text{Temp} \sqsubseteq_{\text{how}} \text{Warm}] \tag{E7}$$

Taxonomic Knowledge. A mereological axiomatisation of taxonomic knowledge has been discussed in detail by Bittner et al. [5]. Appropriate hierarchical relations for $\sqsubseteq_{\text{who}}$ and $\sqsubseteq_{\text{what}}$ are *group inclusion* on groups of agents and the *subclass* relation on classes of objects. With this interpretation, the properties of partial ordering relations, as stated above, are intuitively plausible. An example

[5] A candidate relation for $\sqsubseteq_{\text{why}}$ that fulfils the requirements of a partial ordering relation would, for instance, be the reflexive and transitive hull of the transition relation between programs in the verification of concurrent programs (cf. the relation $\longrightarrow^*$ in Sect. 4.2).

for transitivity of $\sqsubseteq_{\text{who}}$, for instance, is given in (E8): a group of users *Admin* included in the group of users *Staff* is also included in any group that includes the latter, such as *NotificationRecipient*. Knowing that Bob is in the group of administrators, we know that he is eligible to receiving a notification (E9).

$$[\text{Admin} \sqsubseteq_{\text{who}} \text{Staff}] \wedge [\text{Staff} \sqsubseteq_{\text{who}} \text{NotificationRecipient}] \rightarrow [\text{Admin} \sqsubseteq_{\text{who}} \text{NotificationRecipient}] \tag{E8}$$

$$[\text{Bob} \sqsubseteq_{\text{who}} \text{Admin}] \rightarrow [\text{Bob} \sqsubseteq_{\text{who}} \text{NotificationRecipient}] \tag{E9}$$

In our mereological framework, a single agent, such as Bob, or a single object in an interaction is always interpreted as a group of one agent or object. That is, Bob is interpreted not by a token (userID, name, etc.) corresponding to the user Bob but by a group containing only one individual. This may seem as a counter-intuitive by-effect of the mereological axiomatisation. However, this property has distinct advantages, for instance for obfuscation: if we do not distinguish between individuals and groups, every application needs in principle to be enabled to handle coarsened information [29]. Also, application objects are addressed by class not by ID, so that interoperability and greater flexibility of applications can be ensured easier.

4 Programming Dynamic Behaviour in a Context

We describe the syntax (Sect. 4.1) and semantics (Sect. 4.2) of a simple context-aware programming language for specifying pervasive computing systems that operate on the background of a context model. We use a widely known simple textbook variant [2] of an imperative concurrent programming language, so as to illustrate our approach with a particularly familiar example. Our aim is to present a methodology of how the semantics of a given language can be augmented with context-awareness.

For adding context-awareness to programs, a mechanism is needed with which the context-aware program can access the context model. We assume in the following that such an interface is provided by an ontology language in which queries to a knowledge base can be formulated. However, we do not restrict how the knowledge base is realised, whether as a database [18], a graph-based data structure [22], or as a logical reasoning mechanism [28, 35].

As a representation language, we use the context-oriented logical language (CL) proposed by [35], since it is most similar to the first order logical language used above for reasoning *about* context-models. The main difference between the two languages is that CL does not allow quantification over variables. We give an abstract grammar for CL derived from the definitions in [35] and extended it with numerical expressions and context terms for numerical comparison.

4.1 Syntax

We assume two predefined data types: a simple number type (*number*) and a data type of alphanumeric character strings (*alphanum*) for names of atomic

context terms. CL allows for arbitrarily complex context terms (D15). Numerical expressions (N) are numbers, numerical variables and arithmetic expressions constructed from these (D13).

$$N \stackrel{def}{=} \mathit{number} \mid \mathit{NVar} \mid -N \mid N + N \mid N - N \mid N * N \mid N \div N \mid N \bmod N \tag{D13}$$

$$\mathit{NumCT} \stackrel{def}{=} \leq N \mid \geq N \mid \#N \mid < N \mid > N \tag{D14}$$

$$\mathit{CT} \stackrel{def}{=} \mathit{alphanum} \mid \mathit{NumCT} \mid \mathit{CTVar} \mid \mathit{CT} \sqcap \mathit{CT} \mid \mathit{CT} \sqcup \mathit{CT} \mid \sim \mathit{CT} \tag{D15}$$

$$\mathit{CLF} \stackrel{def}{=} [\mathit{CT} \sqsubseteq_m \mathit{CT}] \mid \mathit{CLF} \wedge \mathit{CLF} \mid \mathit{CLF} \vee \mathit{CLF} \mid \neg \mathit{CLF} \mid \mathit{CLF} \rightarrow \mathit{CLF} \mid \mathbf{true} \mid \mathbf{false} \tag{D16}$$

Numerical comparison in the style of Dey [10] can be described with *numerical context terms* (*NumCT*) that consist of a numerical expression prefixed with one of the symbols $\#, \leq, \geq, <, >$ (D14). Intuitively, numerical context terms denote portions of a numerical domain: the interval $(5, 7]$ can be expressed with the term $>5 \sqcap \leq 7$ as the intersection of the intervals $(5, \infty)$ and $(-\infty, 7]$. The context term $\#5$ denotes the interval $[5, 5]$. In order for these expressions to receive the intended numerical meaning, we need to augment the semantics of CL terms with numerical computations (Sect. 4.2).

Context terms can be related with respect to one or more aspects of the six categories. For simplicity, we include only the fundamental six containment relations $\sqsubseteq_m$, where $m \in \{\text{when}, \text{where}, \text{who}, \text{what}, \text{how}, \text{why}\}$, and consider the formulae $[c =_m d]$, $[c\ \mathrm{O}_m\ d]$ to be abbreviations of the more complex formulae by which they are defined above. CL allows not only atomic formulae, but also for arbitrarily complex combinations constructed with the propositional connectives $\wedge, \vee, \rightarrow, \neg$ and two special formulae **true** and **false** (D16).

We characterise a simple imperative language for modelling distributed concurrent processes and supplement it with an additional type for context terms.

$$S \stackrel{def}{=} \mathbf{skip} \mid \mathit{CTVar} := \mathit{CT} \mid \mathit{NVar} := N \mid S\,;S \mid \mathbf{do}\ \mathit{NChoice}\ \mathbf{od} \mid \mathbf{if}\ \mathit{NChoice}\ \mathbf{fi} \tag{D17}$$

$$\mathit{NChoice} \stackrel{def}{=} \mathit{CLF} \rhd S \mid \mathit{NChoice} \,\Box\, \mathit{NChoice} \tag{D18}$$

$$P \stackrel{def}{=} S\,;\mathbf{do}\ \mathit{GChoice}\ \mathbf{od} \tag{D19}$$

$$\mathit{GChoice} \stackrel{def}{=} \mathit{CLF}\,;\mathit{IO} \rhd S \mid \mathit{GChoice} \,\Box\, \mathit{GChoice} \tag{D20}$$

$$\mathit{IO} \stackrel{def}{=} \mathit{CT}\,?\,\mathit{CTVar} \mid \uparrow \mathit{CT} \mid \downarrow \mathit{CT} \tag{D21}$$

$$\mathit{PSyst} \stackrel{def}{=} P \mid \mathit{PSyst} \| \mathit{PSyst} \tag{D22}$$

We allow assignment to context term variables and numerical variables in programs (D17). In non-deterministic choice (D18) and guarded commands (D20), Boolean expressions are replaced with CL-formulae. A pervasive computing system (D22) consists of one or more processes (D19) running in parallel. The

most prominent difference is the set of IO-commands (D21): the command $ct\,?\,x$ listens for activation of a context, the commands $\uparrow CT$ and $\downarrow CT$ send an activation *upward* through the hierarchical context knowledge base, to all more general contexts, or downward, to more specific contexts.

We can now, in a very simple manner, describe context-aware algorithms. Example E10 shows a template for a sensor-like component that consists simply of a loop that activates a context reflecting the current temperature by sending the sensed value v in a way that accuracy (± 1) and unit (*Celsius*) are reflected. Example E11 is a corresponding template for an actuator, as a process executing some action, such as turning a heater on, in response to being activated by a context *TooCold*.

$$\langle \text{sense } v \rangle;\ \textbf{do true};\ \uparrow(\geq\!(v-1) \sqcap \leq\!(v+1) \sqcap \text{Celsius}) \ \triangleright\ \langle \text{sense } v \rangle\ \textbf{od} \quad \text{(E10)}$$

$$\textbf{skip};\ \textbf{do true};\ \text{TooCold}\,?\,x \triangleright \langle \text{execute action} \rangle\ \textbf{od} \quad \text{(E11)}$$

$$\gamma = \{[\leq\!17 \sqcap \text{Celsius} \sqsubseteq_{\text{how}} \text{TooCold}]\} \quad \text{(E12)}$$

With a knowledge base γ specifying that an environment is too cold when the temperature is lower than 17°C (E12), the process (E11) should be triggered if $v < 16$ holds after $\langle \text{sense } v \rangle$. We can prove such properties if a semantics is given.

4.2 Semantics

We describe an operational semantics for the language [2]. The basic notion in the semantics is a *transition* relation $\longrightarrow \subseteq \Gamma \times \Gamma$ between *configurations* Γ of a computing system. The main idea is that the transition relation describes how one step of execution of a program in a state moves the program on and changes the state. The relation $\longrightarrow^*$ is the transitive and reflexive closure of $\longrightarrow$.

A configuration in our framework is represented as a triple $\langle S, \sigma, \gamma \rangle$, where S is a program, process, or pervasive system to be run, σ is a state, and γ a context knowledge base. A terminal configuration $\langle E, \sigma, \gamma \rangle$ is a configuration, in which the program to be run is the empty program E, where $E\,;S$ and $S\,;E$ are the same as S. A state σ is a substitution function, which can be thought of as a mathematical realisation of a list of variable bindings. For instance, applying $\sigma = [x : 0 \mid y : 2]$ to the numerical expression $(x + y)$ results in the numerical expression $(x + y)\sigma = (0 + 2)$, whereas applying $\sigma = [x : 0 \mid y : 2]$ to the context term $\leq\!(x + y)$ results in the context term expression $\leq\!(x + y)\sigma = \leq\!(0 + 2)$.

We define an interpretation function $I_n : N \to \mathbb{R}$ representing evaluation of numerical expressions and an interpretation function I_c that assigns a value to numerical context terms based on I_n. For $x \in \mathit{number}$, $I_n(x)$ trivially maps x to the number in $\mathbb{R}$ that it represents. The compound numerical expressions map to their corresponding evaluations (D23). To give a meaning to the numerical context terms we characterise them as corresponding to the orderings $<$, $\leq$, $>$, $\geq$ on the domain $\mathbb{R}$ (A8-A10).

$$I_n(x+y) = I_n(x) + I_n(y) \qquad I_n(x-y) = I_n(x) - I_n(y) \tag{D23}$$
$$I_n(x*y) = I_n(x) * I_n(y) \qquad I_n(x \div y) = I_n(x) \div I_n(y)$$
$$I_n(-x) = -(I_n(x)) \qquad I_n(x \bmod y) = I_n(x) \bmod I_n(y)$$
$$\forall x, y \in N : [\leq x \sqsubseteq_\forall \leq y] \leftrightarrow I_n(x) \leq I_n(y) \tag{A8}$$
$$\forall x, y \in N : [\geq x \sqsubseteq_\forall \geq y] \leftrightarrow I_n(x) \geq I_n(y) \tag{A9}$$
$$\forall x \in N : [> x = \sim \leq x] \wedge [< x = \sim \geq x] \wedge [\# x = \leq x \sqcap \geq x] \tag{A10}$$

The transition relation can be characterised using axiom schemata and inference rules. The rules for assignment of numerical variables (A11) and context variables (A12) rely on the above interpretation functions. The standard constructs **skip** (A13) and sequential composition (A14) receive standard semantics extended with the parameter γ representing the knowledge base, which can change independently from the program.

$$\langle v := e, \sigma, \gamma\rangle \longrightarrow \langle E, [\sigma \mid v : I_n(e)\sigma], \gamma'\rangle, \text{ if } v \in NVar \text{ and } e \in N, \tag{A11}$$
$$\langle v := e, \sigma, \gamma\rangle \longrightarrow \langle E, [\sigma \mid v : I_c(e)\sigma], \gamma'\rangle, \text{ if } v \in CTVar \text{ and } e \in CT \tag{A12}$$
$$\langle \mathbf{skip}, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle \tag{A13}$$
$$\frac{\langle s_0, \sigma, \gamma\rangle \longrightarrow \langle s_0', \sigma', \gamma'\rangle}{\langle s_0 \,;\, s_1, \sigma, \gamma\rangle \longrightarrow \langle s_0' \,;\, s_1, \sigma', \gamma'\rangle} \tag{A14}$$

The CL formulae in non-deterministic choice (A15) and loops (A16) are evaluated with respect to the knowledge base γ.

$$\langle \mathbf{if}\ \phi_0 \rhd s_0 \,\Box \ldots \Box\, \phi_n \rhd s_n\ \mathbf{fi}, \sigma, \gamma\rangle \longrightarrow \langle s_i, \sigma, \gamma'\rangle, \text{ if } \gamma\sigma \models \phi_i \text{ for any } \phi_i \tag{A15}$$
$$\langle \mathbf{if}\ \phi_0 \rhd s_0 \,\Box \ldots \Box\, \phi_n \rhd s_n\ \mathbf{fi}, \sigma, \gamma\rangle \longrightarrow \langle E, \mathbf{fail}, \gamma'\rangle, \text{ if } \gamma\sigma \not\models \phi_i \text{ for all } \phi_i$$
$$\langle \mathbf{do}\ \phi_0 \rhd s_0 \,\Box \ldots \Box\, \phi_n \rhd s_n\ \mathbf{od}, \sigma, \gamma\rangle \longrightarrow$$
$$\langle s_i \,;\, \mathbf{do}\ \phi_0 \rhd s_0 \,\Box \ldots \Box\, \phi_n \rhd s_n\ \mathbf{od}, \sigma, \gamma'\rangle, \text{ if } \gamma\sigma \models \phi_i \text{ for some } \phi_i \tag{A16}$$
$$\langle \mathbf{do}\ \phi_0 \rhd s_0 \,\Box \ldots \Box\, \phi_n \rhd s_n\ \mathbf{od}, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle, \text{ if } \gamma\sigma \not\models \phi_i \text{ for all } \phi_i$$

All input and output is related via the knowledge base within whose scope a process is working. The command $c\,?\,v$ registers a process with the context c. The process then waits for a context term related to c to be activated by the knowledge base and stores the activated context it received in a context variable v. The command $\uparrow t$ activates all processes registered to contexts c that are in some aspect higher in the hierarchy $[t \sqsubseteq_\exists c]$ (A17); likewise, $\downarrow t$ activates processes registered to contexts below the context t (A18). Upward and downward activation are non-blocking operations that can always be executed and do not change the state σ (A19, A20).

$$\frac{\langle \uparrow t, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle}{\langle c\,?\,v, \sigma, \gamma\rangle \longrightarrow \langle E, [\sigma \mid v : I_c(t)\sigma], \gamma'\rangle},\ \text{if } \gamma \models [t \sqsubseteq_\exists c], v \in CTVar \tag{A17}$$

$$\frac{\langle \downarrow t, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle}{\langle c\,?\,v, \sigma, \gamma\rangle \longrightarrow \langle E, [\sigma \mid v : I_c(t)\sigma], \gamma'\rangle},\ \text{if } \gamma \models [c \sqsubseteq_\exists t], v \in CTVar \tag{A18}$$

$$\langle \uparrow c, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle,\ \text{where } c \in CT \tag{A19}$$

$$\langle \downarrow c, \sigma.\gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle,\ \text{where } c \in CT \tag{A20}$$

We can then specify communication between parallel processes via the knowledge base in a standard way. If none of the conditions of the process can be fulfilled, the process finishes (A21). If one of the conditions ϕ_i holds and the IO-command ioc_i that it guards can be executed, the body s_i is executed (A22). For simplicity, parallel composition is characterised in the same way as sequential composition, except that the order of execution is irrelevant (A23).

$$\langle \mathbf{do}\ \phi_0\,;ioc_0 \rhd s_0 \,\square \ldots \square\, \phi_n\,;ioc_n \rhd s_n \mathbf{od}, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma, \gamma'\rangle, \quad \text{if } \gamma\sigma \not\models \phi_i \text{ for all } \phi_i \tag{A21}$$

$$\frac{\langle ioc_i, \sigma, \gamma\rangle \longrightarrow \langle E, \sigma', \gamma'\rangle}{\langle \mathbf{do}\ \phi_0\,;ioc_0 \rhd s_0 \,\square \ldots \square\, \phi_n\,;ioc_n \rhd s_n \mathbf{od}, \sigma, \gamma\rangle \longrightarrow \langle s_i\,;\mathbf{do}\ \phi_0\,;ioc_0 \rhd s_0 \,\square \ldots \square\, \phi_n\,;ioc_n \rhd s_n \mathbf{od}, \sigma', \gamma'\rangle},\ \text{if } \gamma\sigma \models \phi_i \text{ for some } \phi_i \tag{A22}$$

$$\frac{\langle p_0, \sigma, \gamma\rangle \longrightarrow \langle p_0', \sigma', \gamma'\rangle}{\langle p_0 \| p_1, \sigma, \gamma\rangle \longrightarrow \langle p_0' \| p_1, \sigma', \gamma'\rangle} \text{ and } \frac{\langle p_1, \sigma, \gamma\rangle \longrightarrow \langle p_1', \sigma', \gamma'\rangle}{\langle p_0 \| p_1, \sigma, \gamma\rangle \longrightarrow \langle p_0 \| p_1', \sigma', \gamma'\rangle} \tag{A23}$$

We can now prove that given a knowledge base γ that contains (E12) and a system consisting of the two processes (E10) and (E11), the ⟨execute action⟩ is reached if $v < 16$ holds for the sensed value v. The crucial step in the proof is that of activation of a context term. The sensor activates all contexts above the term $\geq(v-1) \sqcap \leq(v+1) \sqcap \text{Celsius}$. The mereotopological axiom for transitivity (A2) together with the definition of $\sqcap$ (D3) and the axioms for the interpretation of numerical context terms (A8-A10) then entail $\geq(v-1) \sqcap \leq(v+1) \sqcap \text{Celsius} \sqsubseteq_{\text{how}} \leq 17 \sqcap \text{Celsius}$. We can infer that the context $\leq 17 \sqcap \text{Celsius}$ is activated, and it follows that $\geq(v-1) \sqcap \leq(v+1) \sqcap \text{Celsius} \sqsubseteq_{\text{how}} \text{TooCold}$ holds (A2). From the knowledge base γ (E12), we finally obtain that the context TooCold is also activated. By (A17) and (A22), the process (E11) can therefore proceed to the ⟨execute action⟩ statement.

The parameter γ allows us to model changes in the external world and the context knowledge base independent from changes of the computational state σ. The facts that the physical context of a program can change during execution and that this change highly influences the program are the primary difference between context aware computing systems and classical distributed computing systems. Consequently, we need a proof theory that allows us to reason not only about changes evoked by the program, but also about changes evoked in the external world. A user carrying a mobile device leaving a room or a room becoming warmer are not the result of a computation process but of processes

external to the computational system. Computational processes might be able to influence physical processes: a user might be alarmed with a signal to leave a room, a room may warm up after a heater turned on. However, a theory of computational processes should not be burdened with modelling the processes in the physical reality together with computational processes: the user might not be able to hear the signal because he is wearing a noise protection device, the heater might be out of order. In the simple example, γ remained unchanged over the whole execution. For more realistic scenarios and to obtain robust pervasive computing systems, we need to evaluate under which stability assumptions proper execution can be guaranteed. Tractable theories of physical reality, as investigated in the field of knowledge representation within the areas of *naive physics* and *qualitative reasoning* [13, 14], can be used to formulate and efficiently reason about such stability conditions.

5 Conclusions

We described context-aware computing as a proper extension of distributed computing with ontology reasoning. Our description is faithful to principles employed in existing context-aware computing frameworks, so that it can be used to evaluate and develop pervasive computing systems with a wide range of frameworks. The proposed theory integrates developments from two areas providing formal models relevant for pervasive computing: the area of formal ontologies for information systems and the area of formal verification of programming languages.

For formally specifying context models, we proposed to apply mereotopological theories. We showed that key ideas of mereotopology, such as hierarchical organisation and overlap are meaningful for many aspects of context relevant in pervasive computing, such as for modelling uncertainty, ensuring privacy through obfuscation, and context-based address methods beyond unique ids.

Our approach allows to describe context-aware algorithms and pervasive computing systems on a high level of abstraction. The state of a system, which can be directly influenced by a computational process, is cleanly separated from external physical processes reflected in an external knowledge base to which the process is connected. We showed for the familiar example of a CSP-style programming language, how a language can be extended with high-level concepts of context-awareness. However, the method is general enough to be applicable also to other programming languages.

References

[1] Allen, J.: Towards a general theory of action and time. Artificial Intelligence 23, 123–154 (1984)
[2] Apt, K.R., Olderog, E.-R.: Verification of Sequential and Concurrent Programs. Springer, Heidelberg (1991)
[3] Beigl, M., Zimmer, T., Decker, C.: A location model for communicating and processing of context. Personal and Ubiquitous Computing 6(5/6), 341–357 (2002)

[4] Birkedal, L., Debois, S., Elsborg, E., Hildebrandt, T.T., Niss, H.: Bigraphical models of context-aware systems. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 187–201. Springer, Heidelberg (2006)
[5] Bittner, T., Donnelly, M., Smith, B.: Individuals, universals, collections: On the foundational relations of ontology. In: Varzi, A., Vieu, L. (eds.) Third Conference on Formal Ontology in Information Systems. IOS Press, Amsterdam (2004)
[6] Cardelli, L., Gordon, A.D.: Mobile ambients. Theoretical Computer Science 240(1), 177–213 (2000)
[7] Chalmers, D., Dulay, N., Sloman, M.: Towards reasoning about context in the presence of uncertainty. In: Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham, UK (2004)
[8] Chen, H., Perich, F., Finin, T., Joshi, A.: SOUPA: Standard ontology for ubiquitous and pervasive applications. In: International Conference on Mobile and Ubiquitous Systems: Networking and Services (2004)
[9] Cohn, A.G., Hazarika, S.M.: Qualitative spatial representation and reasoning: An overview. Fundamenta Informaticae 46(1-2), 1–29 (2001)
[10] Dey, A.K.: Providing Architectural Support for Building Context-Aware Applications. PhD thesis, Georgia Institute of Technology (2000)
[11] Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) Pervasive 2005. LNCS, vol. 3468, pp. 152–170. Springer, Heidelberg (2005)
[12] Euzenat, J.: Granularity in relational formalisms - with application to time and space representation. Computational Intelligence 17(3), 703–737 (2001)
[13] Forbus, K.D.: Qualitative process theory. Artificial Intelligence 24(1-3), 85–168 (1984)
[14] Galton, A.: Qualitative Spatial Change. Oxford University Press, Oxford (2000)
[15] Galton, A.: Operators vs. arguments: the ins and outs of reification. Synthese 150, 415–441 (2006)
[16] Gottfried, B., Guesgen, H.W., Hübner, S.: Spatiotemporal reasoning for smart homes. In: Augusto, J.C., Nugent, C.D. (eds.) Designing Smart Homes, pp. 16–34 (2006)
[17] Hennessy, M.: A Distributed Pi-Calculus. Cambridge University Press, Cambridge (2007)
[18] Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Pervasive and Mobile Computing 2, 37–64 (2006)
[19] Jang, S., Ko, E.-J., Woo, W.: Unified user-centric context: Who, where, when, what, how and why. In: Ko, H., Krüger, A., Lee, S.-G., Woo, W. (eds.) Personalized Context Modeling and Management for UbiComp Applications, vol. 149, pp. 26–34 (2005) CEUR-WS
[20] Jiang, C., Steenkiste, P.: A hybrid location model with a computable location identifier for ubiquitous computing. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, pp. 246–263. Springer, Heidelberg (2002)
[21] Langheinrich, M.: Privacy by design - principles of privacy-aware ubiquitous systems. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201, pp. 273–291. Springer, Heidelberg (2001)
[22] Leonhardt, U.: Supporting Location Awareness in Open Distributed Systems. PhD thesis, Imperial College, London, UK (1998)
[23] Ligozat, G.: Generalized intervals: A guided tour. In: Proceedings of the ECAI 1998 Workshop on Spatial and Temporal Reasoning, Brighton, UK (1998)
[24] Milner, R.: Bigraphs and their algebra. Electronic Notes on Theoretical Computer Science 209, 5–19 (2008)

[25] Pease, A., Niles, I., Li, J.: The suggested upper merged ontology: A large ontology for the semantic web and its applications. In: AAAI 2002 Workshop on Ontologies and the Semantic Web (2002)
[26] Randell, D., Cui, Z., Cohn, A.: A spatial logic based on region and connection. In: Knowledge Representation and Reasoning, pp. 165–176. Morgan Kaufmann, San Francisco (1992)
[27] Ranganathan, A., Campbell, R.H.: Provably correct pervasive computing environments. In: PerCom, pp. 160–169 (2008)
[28] Ranganathan, A., McGrath, R.E., Campbell, R.H., Mickunas, M.D.: Use of ontologies in a pervasive computing environment. The Knowledge Engineering Review 18(3), 209–220 (2003)
[29] Rashid, U., Schmidtke, H.R., Woo, W.: Managing disclosure of personal health information in smart home healthcare. In: Stephanidis, C. (ed.) International Conference on Universal Access in Human-Computer Interaction, Held as Part of HCI International, pp. 188–197. Springer, Heidelberg (2007)
[30] Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal Communications, 10–17 (2001)
[31] Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. IEEE Network 8(5), 22–32 (1994)
[32] Schmidt, A., Beigl, M., Gellersen, H.-W.: There is more to context than location. Computers and Graphics 23(6), 893–901 (1999)
[33] Schmidtke, H.R., Woo, W.: A formal characterization of vagueness and granularity for context-aware mobile and ubiquitous computing. In: Youn, H.Y., Kim, M., Morikawa, H. (eds.) UCS 2006. LNCS, vol. 4239, pp. 144–157. Springer, Heidelberg (2006)
[34] Schmidtke, H.R., Woo, W.: A size-based qualitative approach to the representation of spatial granularity. In: Veloso, M.M. (ed.) Twentieth International Joint Conference on Artificial Intelligence, pp. 563–568 (2007)
[35] Schmidtke, H.R., Hong, D., Woo, W.: Reasoning about models of context: A context-oriented logical language for knowledge-based context-aware applications. Revue d'Intelligence Artificielle 22(5), 589–608 (2008)
[36] Strang, T., Linnhoff-Popien, C., Frank, K.: CoOL: A context ontology language to enable contextual interoperability. In: Stefani, J.-B., Demeure, I., Hagimont, D. (eds.) DAIS 2003. LNCS, vol. 2893, pp. 236–247. Springer, Heidelberg (2003)
[37] Varzi, A.C.: Spatial reasoning and ontology: Parts, wholes, and locations. In: Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.) Handbook of Spatial Logics, pp. 945–1038. Springer, Heidelberg (2007)
[38] Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: PerCom Workshops, pp. 18–22. IEEE Computer Society Press, Los Alamitos (2004)
[39] Ye, J., Coyle, L., Dobson, S., Nixon, P.: Ontology-based models in pervasive computing systems. The Knowledge Engineering Review 22, 315–347 (2007)
[40] Ye, J., Coyle, L., Dobson, S., Nixon, P.: A unified semantics space model. In: Hightower, J., Schiele, B., Strang, T. (eds.) LoCA 2007. LNCS, vol. 4718, pp. 103–120. Springer, Heidelberg (2007)