Pervasive and Mobile Computing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

A top-level ontology for smart environments

Juan Ye [*], Graeme Stevenson, Simon Dobson

School of Computer Science, University of St Andrews, UK

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Ontologies
Smart home
Activity recognition
Semantics
Domain knowledge

ABSTRACT

Recognising human activities is a problem characteristic of a wider class of systems in which algorithms interpret multi-modal sensor data to extract semantically meaningful classifications. Machine learning techniques have demonstrated progress, but the lack of underlying formal semantics impedes the potential for sharing and reusing classifications across systems. We present a top-level ontology model that facilitates the capture of domain knowledge. This model serves as a conceptual backbone when designing ontologies, linking the meaning implicit in elementary information to higher-level information that is of interest to applications. In this way it provides the common semantics for information at different levels of granularity that supports the communication, reuse and sharing of ontologies between systems.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

The study of human activities has implication in domains such as personalised healthcare, the development of more comfortable home environments, and improved energy conservation [1,2]. Sensor researchers have made substantial progress in producing small, easily pluggable, and energy-efficient sensors, making it possible to deploy smart home technologies in real-world settings [3]. Machine learning techniques have been successfully applied to recognising user activities from these sensor data [1,4]. However, data, knowledge, and application heterogeneity [5] restricts the wide deployment of these techniques in ordinary homes. Data heterogeneity, caused by the diversity of sensors and the lack of a uniform markup language, hinders their seamless exchange, integration, and reuse. Knowledge heterogeneity, caused by the lack of uniformity and formality, prevents the sharing of defined or learned domain knowledge across different systems. Different applications require the recognition of different human activities, while the lack of semantics to relate the activities leads to application heterogeneity.

A variety of computer science fields, including e-commerce [6,7], information integration [8–10], and the semantic web [11,12] recognise ontology, as a promising means for knowledge sharing and reuse. Ontology is a formal explicit specification of a shared conceptualisation [13] in that it supports the capture and specification of domain knowledge with its intrinsic semantics through consensual terminology and formal axioms. Ontologies support a set of modelling primitives to define classes, individuals, their attributes and their relations. The formal axioms specify the intended meaning of the terminology with the modelling primitives and their *a priori* relationships in an explicit way [14]. The consensual terminology makes it possible to share and reuse data and knowledge across different components in a system and across different systems [15,5]. Pervasive computing projects have applied ontologies in modelling and reasoning on concepts and knowledge [16–18]. These works demonstrate that ontology is a promising technique that can be used to address data, knowledge, and application heterogeneity.

Most ontologies developed for pervasive computing concentrate on modelling specific properties and internal structures of concepts in individual domains such as *Location*, *Person*, and *Computational Entity* [5], while ignoring the common

[*] Corresponding author. Tel.: +44 1334461630.
E-mail addresses: ye@cs.st-andrews.ac.uk (J. Ye), gs@cs.st-andrews.ac.uk (G. Stevenson).

1574-1192/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2011.02.002

ARTICLE IN PRESS

semantics underlying them. This leaves developers a burden of engineering a large amount of domain knowledge, which can be relieved through a higher-level uniform approach, and as well as of completing tasks such as detecting inconsistency of context, configuring new activities, and customising application behaviours on a number of human activities.

For these reasons, it is desirable to underlay these domain ontologies with a foundational ontology model, where domain concepts and knowledge are modelled uniformly among different types of information. Based on the classic set theory, we propose such a model to address the above issues. The contribution of this work is to provide a top-level ontology to model and reason on domain knowledge in a precise and traceable manner, serving as a conceptual backbone for developing domain and application ontologies for smart environments.

We start by capturing essential semantics common to all dimensions of an information space; i.e., *levels of granularity*, *conflicting*, and *overlapping* relationships. These common semantics provide a uniform way to represent and reason on domain knowledge. Ontologies built using them are straightforwardly used and interpreted by evaluating and comparing these underlying semantics of the concepts and terms they define. Based on these semantics, we employ a generic approach to derive knowledge about *context predicates* and *activities* in a structured manner. We define a context predicate as a two-dimensional product space of information, while an activity is a multiple-dimensional product space of information that indicates a situation occurring in the environment. Through reasoning about the relationships between context predicates, we can detect inconsistency between context; that is, when two pieces of context report conflicting states of reality. We support defining two types of composite activities and inferring knowledge between them, which can be used to check the consistency of activity derivation rules and to aid application design.

We structure the remainder of the paper as follows. Section 2 reviews literature pertaining to ontology-based models in smart home environments. Section 3 presents how we represent and evaluate the common semantics of domain concepts, based on which Sections 4 and 5 describe how to infer relationships between context predicates and activities. Section 6 presents the implementation of the top-level ontology and demonstrates the use of the ontology model through a real-world smart home case study. Finally, Section 7 summarises our work and outlines future research directions.

2. Related work

Gruber et al. introduce a precise definition of ontology: ‘‘a formal explicit specification of a shared conceptualisation’’ [19,13]. This definition describes the philosophical nature of ontology, which Guarino et al. advances as ‘‘a logical theory accounting for the intended meaning of a formal vocabulary’’ [20,14]. That is, when an ontology models a particular aspect of reality – an *intended domain* – it specifies an *ontological commitment* that is the underlying conceptualisation about the intended domain, including identities of concepts and internal structures between them.

Following the logical definition of ontology, we propose an ontology on top of domain and application ontologies (in Fig. 1) that aims to provide a precise and traceable representation of the conceptualisation. Domain ontologies model each dimension of information, including Object, Location, and Equipment, while application ontologies describe concepts in particular application domains; for example, Patient and PersonAndMedical in healthcare applications, Household in smart home applications, and MeetingVideoEvent in intelligent meeting applications. The top-level ontology captures the common semantics among dimensions of information that are described in the lower-level ontologies and exploits the semantics in compositions of the information that are represented in predicates and derivation rules in the lower-level ontologies. Based on the common semantics, this top-level ontology provides generic rules to facilitate pervasive tasks including detecting inconsistency of information, generating new knowledge such as new activities for different applications and new relationships between concepts, contexts, and activities, and designing applications in a hierarchical way. It also encourages developers to define their own generic rules to perform similar system-level tasks.

In the literature of ontological models in smart environments, the ontology proposed by Schmidtke et al. is a top-level ontology that is similar to ours. Their ontology is built on the theory of mereotopology, a formal theory of part and whole that has been widely used in the area of formal ontology [21]. From a primitive relation *part-hood*, they discuss the *overlap*, *underlap*, and *connection* relations between contexts. Two contexts overlap if and only if they share a part that is not identical to the empty (i.e., non-existing) context, while two contexts underlap if and only if they are both part of a context that is not identical to the whole domain. A connection can be introduced between two contexts, which is a reflexive and symmetric relation. These relations are considered crucial in modelling the notions of reachability in a conceptual hierarchy. Our ontology studies the semantics of information also based on partial ordering, while our work instead focuses on structural semantics across different levels of information from domain concepts, to context predicates, and to activities.

Most smart environment ontologies can be classified as either domain or application [22–27,5]. Ye et al. review the most popular ontologies in modelling and reasoning on contexts in pervasive computing [18]. Among them, SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) is one of the most comprehensive ontologies. SOUPA not only includes standard domain ontologies such as *FOAF*, *DAML-Time*, *Region Connection Calculus*, and *Rei Policy* ontology, but also defines ontologies specific to pervasive computing like software agents, mobile devices, and events like meetings. It offers a formal and well-structured way to model context and thus provides rich semantics for programming [18].

Gu et al. [24,27] present an ontology-based model for context fusion in activity recognition. It is built in a hierarchical manner from an upper-level ontology to a domain-specific ontology. The upper-level ontology captures features in each type of basic contextual entities such as Person, Device, and Location, while domain-specific ontologies are a collection of

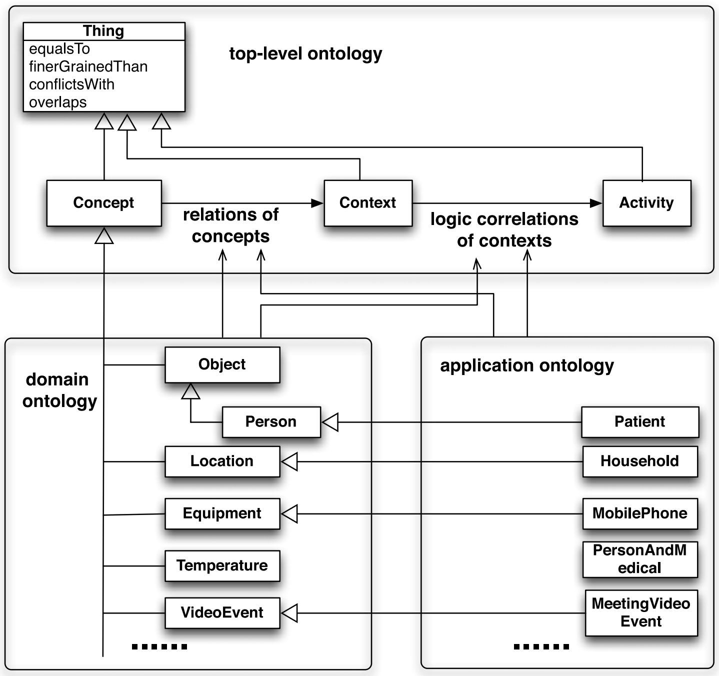


Fig. 1. Ontology classifications.

ontologies that define the features of a sub-domain, such as smart homes or smart health care. The upper-level and domain-specific ontologies map to the domain and application ontologies in Fig. 1 respectively.

Beyond semantic enrichment at a conceptual level, ontology can also serve as an effective basis for the development of knowledge-centric software, for example agent frameworks and middleware for context-aware systems [28,29,5]. Gaia is an infrastructure for smart spaces, which aims to bring the functionality of an operating system to physical spaces [30]. The Gaia ontologies work as a system specification by providing a standard taxonomy to describe different kinds of entities including applications, services, devices, users, and data sources. The ontologies are mainly used for service discovery, matchmaking, interoperability between entities, and interaction between human users and computers [30].

Chen et al. [5] propose a layered conceptual architecture for semantic smart homes. The novelty of this architecture is that there exists a Semantic layer and an Intelligent Service layer between the Data and Application layers. The Semantic layer achieves data interoperability and machine understandability by using ontologies to provide a homogeneous view over heterogeneous data, while the Intelligent Service layer delivers the capability of interoperability and high-level automation by exploiting semantics and descriptive knowledge in the ontologies.

Latfi et al. [28] also propose an ontology-based model of the Telehealth Smart Home. It aims to provide assistance to a patient by preventing any potentially dangerous situation that could endanger his life. The ontologies are partitioned into seven sub-domains, including the Habitat ontology that describes the structure of the housing facility; the PersonAndMedical ontology for representing patients' medical history and caregivers' caring duties; the Equipment ontology, which describes the equipment setup in the house; the Software application ontology for the modules of the system; the Task ontology, which describes the tasks that the patient, the operation actor, and the system aims to achieve; the Behaviour ontology, which captures patients' life habits and critical physiological parameters; and the Decision ontology, which describes behaviours when a critical situation or a change of habits is detected. These ontologies serve not only as a conceptual model to represent domain concepts, but they also provide an effective basis for the development, configuration, and execution of software applications.

In summary, most of the above ontologies follow the philosophical nature of ontology; that is, defining a commonly agreed vocabulary for representing data and knowledge structure; i.e., domain concepts, attributes, and relations between them. The vocabulary makes data understandable, sharable, and reusable by both humans and machines. For such ontologies, we provide a complete methodology to help developers analyse and capture common semantics in any type of information, from low-level elementary information to high-level activity descriptions. This model enables reasoning on the underlying correlation across the levels of information, automatically deriving higher-level concepts and inferring their relationships. The resultant knowledge can facilitate system-level tasks such as checking the consistency of a system, and also aid the process of application design.

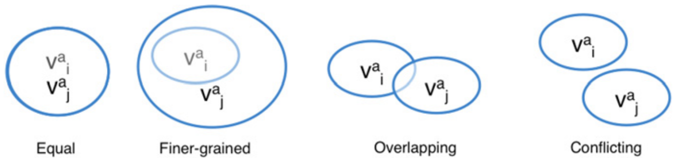


Fig. 2. Relationships between abstract values in one dimension of information. Note: each circle represents a set of ground values that an abstract value maps to.

3. Concept model

This section introduces a formal concept model, where we employ a generic approach to analyse and structure an information space. This model forms a basis for reasoning about the common semantics of context predicates and activities.

3.1. Dimensions of information

An information space can be divided into multiple dimensions, each dimension with its own structural and relational characteristics, distinguishable from other dimensions of information. For example, one dimension is Location, which can be represented as a coordinate consisting of three numerical values with a given unit of length measurement or as a descriptive term, such as `livingRoom`, which indicates a symbolic place in a house. Time is another information dimension; its information can be represented as an instant, interval, or a term with associated temporal semantics, such as `Monday` or `evening`. Other dimensions of information include Distance, Speed, Acceleration, Temperature, Humidity, Person, and Heart rate.

3.2. Abstraction of information

Each dimension of information contains a set of irreducible *ground values*; that is, the smallest values perceivable. In one dimension, ground values are disjoint and exhaustive so as to cover the whole information space of the dimension. Depending on the dimension, a ground value set may be finite or infinite, continuous or discrete. A dimension may have multiple sets of ground values, each characterised by a measurement system. Throughout the rest of this paper we assume the existence of a principal measurement system for the ground value set in each dimension of an information space, to which we can convert ground value sets expressed using other measurement systems.

For example, we can compose the ground value set of the Location dimension using GPS coordinate points that cover an environment, where no two coordinates are the same. This set is infinite and continuous, and we can convert coordinate points expressed using other coordinate systems (e.g., a user-defined Cartesian coordinate system) to this principal measurement system [31]. Temperature ground values can be any numerical value expressed in, say, Kelvin, Celsius or Fahrenheit, which are also infinite and continuous. Another dimension example is Boolean state, whose ground value set of `true` and `false` is finite and discrete.

Each information dimension can have a set of *abstract values*, which are usually human-friendly or application-interesting concepts. For example in the Location dimension, an abstract value may be the symbolic location `livingRoom`, which maps to a set of coordinate points bound to the living room of a house. In the Temperature dimension, an abstract value of `hot` may indicate an individual's perception of the temperature through a mapping to a range of ground values expressed in Celsius. Similarly, we may define the abstract values for the Acceleration dimension as `fast` or `slow`, each mapped to a range of ground values appropriate to the application at hand. We give a formal definition of an abstract value set in Definition 1.

Definition 1. [1] Let V^g be a ground value set of one dimension of information. A set of abstract values V^a is defined via a mapping function $\mu : V^a \to \mathcal{P}(V^g)$, where $\mathcal{P}(V^g)$ is the power set of V^g.

3.3. Semantics of information

Fig. 2 describes the core semantics of abstract values that are common to all information dimensions – *equal to*, *finer-grained*, *conflicting* and *overlapping* – using a set-theoretic approach. Definition 2 defines how these are evaluated from the mapping functions of abstract values. We base the definition of more complex semantics on these elementary semantics; for example we may define a *strictly finer-grained* relationship to represent one abstract value that is finer-grained than but not equal to another.

Definition 2. Given two abstract values in one dimension of information: $v^a_i, v^a_j \in V^a$,

- v^a_i is finer-grained than v^a_j, denoted as $v^a_i \preceq v^a_j$, iff $\mu(v^a_i) \subseteq \mu(v^a_j)$. Particularly, v^a_i is equal to v^a_j, denoted as $v^a_i =_c v^a_j$, iff $\mu(v^a_i) = \mu(v^a_j)$;

[1] The rest of the paper follows the same symbolism and terminology used in this definition.

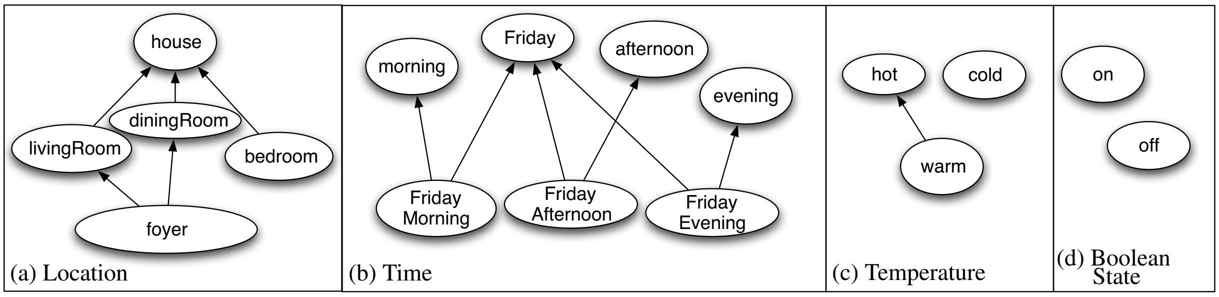


Fig. 3. Context examples in different domain concepts.

- v_i^a conflicts with v_j^a, denoted as $v_i^a \nparallel v_j^a$, iff $\mu(v_i^a) \cap \mu(v_j^a) = \emptyset$;
- v_i^a overlaps with v_j^a, denoted as $v_i^a \bowtie v_j^a$, iff $\mu(v_i^a) \cap \mu(v_j^a) \neq \emptyset$, $\mu(v_i^a) \nsubseteq \mu(v_j^a)$, and $\mu(v_j^a) \nsubseteq \mu(v_i^a)$.

The interpretation and evaluation of these semantics varies with the dimension of information under consideration. For example, we interpret relationships in the Location dimension as spatial relationships between symbolic locations, which we evaluate by projecting their coordinate sets onto a coordinate system and comparing them [32]. Fig. 3(a) illustrates that: (1) the abstract value of `house` contains another three abstract values `livingRoom, diningRoom`, and `bedroom` (*granularity*); (2) `livingRoom` and `diningRoom` are disjoint from `bedroom` (*conflicting*); and (3) `livingRoom` and `diningRoom` share a common location `foyer` (*overlapping*).

With Time, we define temporal terms using a range of time instances, and infer relations between two temporal terms by projecting their ranges on the time axis. As shown in Fig. 3(b), `morning, afternoon` and `evening` conflict with each other, while the three share an overlapping time period with `Friday`, defined as `FridayMorning`, `FridayAfternoon`, and `FridayEvening`. Similarly with Temperature, we define each term using by a degree range in Celsius, and evaluate their relations by comparing these ranges. In Fig. 3(c), `hot` conflicts with `cold` if their degree ranges are exclusive, while `warm` is finer-grained than `hot` if its degree range is contained within that of `hot`. The ground value set of Boolean information consists of `true` and `false`. All of its abstract values map to one member of this pair, but may take different names in different contexts, such as `on` or `off` as shown in Fig. 3(d). Boolean values are either equal or conflicting.

Through explicit mappings to ground values, Definition 2 provides an approach through which we compare terms or concepts defined in different ontologies in a quantifiably measurable way. This forms a basis for mapping, communicating, and sharing ontologies across different smart environments.

The above examples demonstrate how we use Definitions 1 and 2 to "compute" the common semantics of information. We use these semantics directly as a medium with which to model expert knowledge about one information dimension in a certain domain; for example, a space map, or knowledge about the relationships between a person's different social circles (e.g., an overlap between `friends` and `colleagues`).

3.4. Meta-properties

As shown in the previous section, our concept model provides a uniform way to model knowledge with the same inherent semantics in any domain. We use these semantics in this Section to develop a generic reasoning approach, uncovering the implicit knowledge in any single dimension. The meta-properties of the four relationships in Definition 2 support this process.

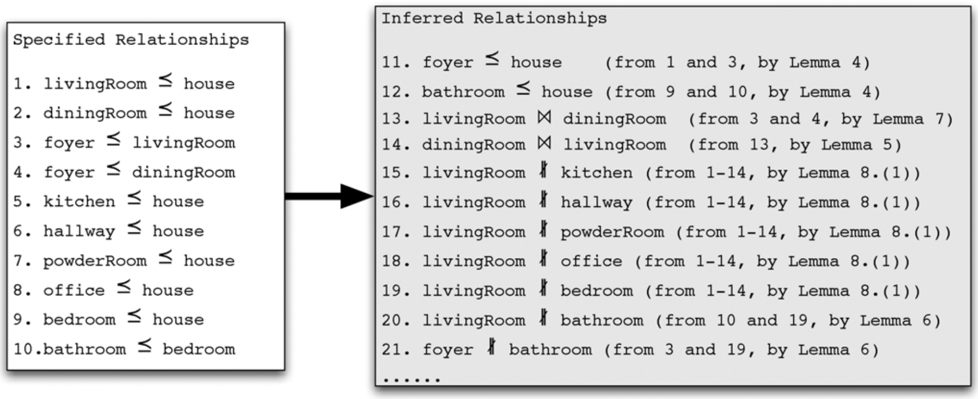
Lemma 3. *An equal relationship is reflexive, transitive, and symmetric.*

Lemma 4. *A finer-grained relationship is reflexive, transitive, and antisymmetric.*

Lemma 5. *Conflicting and overlapping relationships are symmetric.*

Lemmas 3–5 provide the meta-properties of the four relationships, which describe how we derive relationships between concepts in a model that are not explicitly stated. There are also generic properties relating to the structure of these relationships that make it possible to derive further knowledge. That is, if two abstract values conflict, then any of their finer-grained abstract values also conflict with each other (Lemma 6); also, if one abstract value is finer-grained than another two abstract values, neither of which are finer-grained than the other, then the latter two values overlap (Lemma 7). The *finer-grained*, *conflicting*, and *overlapping* relationships form an exhaustive and mutually exclusive set of relationships between any two concepts; that is, any two concepts must have one and only one of these three relationships (Lemma 8). For example, if two concepts are neither finer-grained than each other nor overlapping with each other, then they conflict with each other.

Lemma 6. $\forall v_i^a, v_j^a \in V^a$, *if* $v_i^a \nparallel v_j^a$, *then* $\forall v_{i'}^a \preceq v_i^a$ *and* $\forall v_{j'}^a \preceq v_j^a$, $v_{i'}^a \nparallel v_{j'}^a$.



```
Specified Relationships

1. livingRoom ≤ house
2. diningRoom ≤ house
3. foyer ≤ livingRoom
4. foyer ≤ diningRoom
5. kitchen ≤ house
6. hallway ≤ house
7. powderRoom ≤ house
8. office ≤ house
9. bedroom ≤ house
10.bathroom ≤ bedroom
```

```
Inferred Relationships

11. foyer ≤ house    (from 1 and 3, by Lemma 4)
12. bathroom ≤ house (from 9 and 10, by Lemma 4)
13. livingRoom ⋈ diningRoom  (from 3 and 4, by Lemma 7)
14. diningRoom ⋈ livingRoom  (from 13, by Lemma 5)
15. livingRoom ∦ kitchen (from 1-14, by Lemma 8.(1))
16. livingRoom ∦ hallway (from 1-14, by Lemma 8.(1))
17. livingRoom ∦ powderRoom (from 1-14, by Lemma 8.(1))
18. livingRoom ∦ office (from 1-14, by Lemma 8.(1))
19. livingRoom ∦ bedroom (from 1-14, by Lemma 8.(1))
20. livingRoom ∦ bathroom (from 10 and 19, by Lemma 6)
21. foyer ∦ bathroom (from 3 and 19, by Lemma 6)
......
```

Fig. 4. An example to demonstrate the inference of new knowledge.

Proof. By Definition 2,

$v_i^a \nparallel v_j^a \Rightarrow \mu(v_i^a) \cap \mu(v_j^a) = \emptyset$ (1),
$v_{i'}^a \preceq v_i^a \Rightarrow \mu(v_{i'}^a) \subseteq \mu(v_i^a)$ (2),
$v_{j'}^a \preceq v_j^a \Rightarrow \mu(v_{j'}^a) \subseteq \mu(v_j^a)$ (3).

From (1), (2), and (3) $\Rightarrow \mu(v_{i'}^a) \cap \mu(v_{j'}^a) = \emptyset$, by Definition 2 $\Rightarrow v_{i'}^a \nparallel v_{j'}^a$. □

Lemma 7. *Given $v_i^a, v_j^a, v_k^a \in V^a$, if $\mu(v_k^a) \neq \emptyset$, $v_k^a \preceq v_i^a$, $v_k^a \preceq v_j^a$, $v_i^a \npreceq v_j^a$, and $v_j^a \npreceq v_i^a$, then $v_i^a \bowtie v_j^a$.*

Proof. By Definition 2,

$v_k^a \preceq v_i^a \Rightarrow \mu(v_k^a) \subseteq \mu(v_i^a)$ (1),
$v_k^a \preceq v_j^a \Rightarrow \mu(v_k^a) \subseteq \mu(v_j^a)$ (2).

From (1) and (2) $\Rightarrow \mu(v_k^a) \subseteq \mu(v_i^a) \cap \mu(v_j^a)$ (3).
From (3) and $\mu(v_k^a) \neq \emptyset \Rightarrow \mu(v_i^a) \cap \mu(v_j^a) \neq \emptyset$ (4).

$v_i^a \npreceq v_j^a \Rightarrow \mu(v_i^a) \nsubseteq \mu(v_j^a)$ (5),
$v_j^a \npreceq v_i^a \Rightarrow \mu(v_j^a) \nsubseteq \mu(v_i^a)$ (6).

From (4), (5), and (6), by Definition 2 $\Rightarrow v_i^a \bowtie v_j^a$. □

Lemma 8. *Given $v_i^a, v_j^a \in V^a$,*
(1) if $\neg(v_i^a \bowtie v_j^a)$, $v_j^a \npreceq v_i^a$, and $v_i^a \npreceq v_j^a$, then $v_i^a \nparallel v_j^a$;
(2) if $\neg(v_i^a \nparallel v_j^a)$, $v_j^a \npreceq v_i^a$, and $v_i^a \npreceq v_j^a$, then $v_i^a \bowtie v_j^a$;
(3) if $\neg(v_i^a \bowtie v_j^a)$, $\neg(v_i^a \nparallel v_j^a)$, and $v_j^a \npreceq v_i^a$, then $v_i^a \preceq v_j^a$;
(4) if $\neg(v_i^a \bowtie v_j^a)$, $\neg(v_i^a \nparallel v_j^a)$, and $v_i^a \npreceq v_j^a$, then $v_j^a \preceq v_i^a$.

The above lemmas provide a theoretical foundation for generic reasoning. This keeps developers from laborious coding of domain knowledge. We need only define mapping functions, or specify the set of primary relationships. We apply the generic reasoning schema uniformly to the whole information space to derive more knowledge about the relationships between concepts. Fig. 4 demonstrates how we use the above lemmas to infer additional knowledge from that which is pre-specified. Without defining a mapping function on the coordinates of the house in Fig. 3, developers define the relationships on the left-hand side of Fig. 4; that is, the immediately spatial containment relationships between the locations. Lemmas 3–8 are then used to automatically infer new knowledge that has not been explicitly defined, as shown on the right-hand side of Fig. 4.

It is worth noting at this point that the above meta-properties are uniformly applied at all levels of the model allowing us to explore the knowledge about context predicates and activities as well. We will return to this later.

4. Context model

We use the similar definitions proposed by Yao et al. [33] and Loke [34] to define context as "any information acquired from a system or an environment". Developers or users can profile context, or acquire it from physical or virtual sensors [35]. A piece of context asserts a state of reality, which we model as a relation between two abstract values, each belonging to a dimension of information [36]. Fig. 5 shows examples of relations between abstract values in different dimensions of information. We define two relations on Object and Location: `locatedIn`, which links objects to the symbolic location that contains them, and `contains`, which is the inverse of this relationship. Also shown are relations on the Person dimension and other dimensions via the relations `hasBloodPressure`, `hasHeartRate`, and `accesses`.

ARTICLE IN PRESS

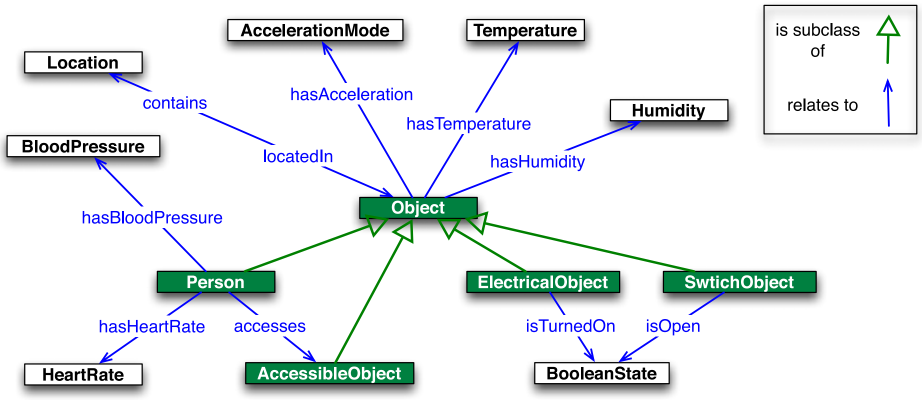


Fig. 5. An example of relations between dimensions of information.

4.1. *Context predicate*

A *context predicate* encapsulates two abstract values engaging in a relation, which corresponds to a product space these two values form in an information space. We represent this as a triple $\langle s, p, o\rangle$, where the subject s and object o are abstract values in two dimensions; for example, $\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{livingRoom}\rangle$. A pair of abstract values *validate* a context predicate when they fall into its corresponding product space. For example, the pair $(\{\text{Bob}\}, \texttt{foyer})$ validates the context predicate $\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{livingRoom}\rangle$ according to their spatial relationships in Fig. 3(a). A context predicate can also incorporate variables; for example $\langle ?b, \texttt{locatedIn}, ?l\rangle$, where b and l indicate any abstract value from the Object and Location dimensions. Any pair of abstract values in Object and Location that are related by the `locatedIn` relationship validates such a predicate. Context, also called an *instantiated* context predicate, is a validated context predicate with a timestamp, which represents a state of reality, denoted as $\langle\langle s, p, o\rangle, t\rangle$. For example, we represent that Bob is in the bedroom during a specific time period as $\langle\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{bedroom}\rangle, [2006-08-23T23:48:43Z, 2006-08-23T23:48:45Z]\rangle$. We provide a formal definition of a context predicate as follows.

Definition 9. Let p be a relation: $S^a \times O^a$, where S^a and O^a are abstract value sets from two dimensions of information.

- We define a context predicate as $\langle s, p, o\rangle \to \{0, 1\}$, where $s \in S^a$ and $o \in O^a$.
- A pair of values (s', o') validate a context predicate $\langle s, p, o\rangle$, where $s' \in S^a$ and $o' \in O^a$, iff $s' \preceq s$ and $o' \preceq o$.
- An instantiated context predicate is a valid context predicate with a timestamp, denoted as $\langle\langle s, p, o\rangle, t\rangle$, where t is either a time instant or interval.

4.2. *Relationships between context predicates*

We use the relations that has been defined within the information dimensions to study semantics between context predicates. Intuitively, when the user Bob is in the living room, we consider that (1) he is also in the house ($\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{livingRoom}\rangle$ is finer-grained than $\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{house}\rangle$); (2) he cannot be in the bedroom ($\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{livingRoom}\rangle$ conflicts with $\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{bedroom}\rangle$); and (3) he may also be in the dining room ($\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{livingRoom}\rangle$ overlaps with $\langle\{\text{Bob}\}, \texttt{locatedIn}, \texttt{diningRoom}\rangle$). We formally define these relationships in Definition 10.

Definition 10. Given two context predicates $\langle s_i, p, o_i\rangle$ and $\langle s_j, p, o_j\rangle$ that share the same relation $p : S^a \times O^a$,

- $\langle s_i, p, o_i\rangle$ is finer-grained than $\langle s_j, p, o_j\rangle$, denoted as $\langle s_i, p, o_i\rangle \preceq \langle s_j, p, o_j\rangle$, iff $\forall(s, o) \in S^a \times O^a$, (s, o) validates $\langle s_i, p, o_i\rangle$ implies that $\langle s_j, p, o_j\rangle$ is also validated;
- $\langle s_i, p, o_i\rangle$ conflicts with $\langle s_j, p, o_j\rangle$, denoted as $\langle s_i, p, o_i\rangle \nparallel \langle s_j, p, o_j\rangle$, iff $\nexists(s, o) \in S^a \times O^a$, (s, o) validates both $\langle s_i, p, o_i\rangle$ and $\langle s_j, p, o_j\rangle$;
- $\langle s_i, p, o_i\rangle$ overlaps with $\langle s_j, p, o_j\rangle$, denoted as $\langle s_i, p, o_i\rangle \bowtie \langle s_j, p, o_j\rangle$, iff $\exists(s, o), (s', o'), (s'', o'') \in S^a \times O^a$, (s, o) validates both $\langle s_i, p, o_i\rangle$ and $\langle s_j, p, o_j\rangle$, (s', o') validates $\langle s_i, p, o_i\rangle$ but not $\langle s_j, p, o_j\rangle$, and (s'', o'') validates $\langle s_j, p, o_j\rangle$ but not $\langle s_i, p, o_i\rangle$.

As we define a context predicate on dimensions of an information space, we can infer the above relationships based on the relationships in their corresponding dimensions of information, as we show in Lemma 11.

Lemma 11. *Given two context predicates* $\langle s_i, p, o_i\rangle$ *and* $\langle s_j, p, o_j\rangle$, *where* $s_i, s_j \in S^a$ *and* $o_i, o_j \in O^a$,

- $\langle s_i, p, o_i\rangle \preceq \langle s_j, p, o_j\rangle$, *if* $s_i \preceq s_j$ *and* $o_i \preceq o_j$;
- $\langle s_i, p, o_i\rangle \nparallel \langle s_j, p, o_j\rangle$, *if* $s_i =_c s_j$ *and* $o_i \nparallel o_j$, *or* $o_i =_c o_j$ *and* $s_i \nparallel s_j$;
- $\langle s_i, p, o_i\rangle \bowtie \langle s_j, p, o_j\rangle$, *if* $s_i =_c s_j$ *and* $o_i \bowtie o_j$, *or* $o_i =_c o_j$ *and* $s_i \bowtie s_j$, *or both.*

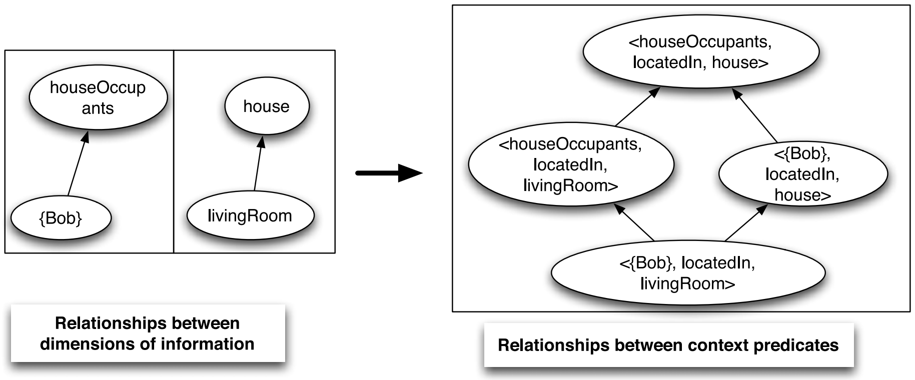

houseOccupants
{Bob}
house
livingRoom
<houseOccupants, locatedIn, house>
<houseOccupants, locatedIn, livingRoom>
<{Bob}, locatedIn, house>
<{Bob}, locatedIn, livingRoom>
Relationships between dimensions of information
Relationships between context predicates

Fig. 6. An example of inferring relations between context predicates from those between their involved abstract values.

Proof. By Definition 9, $\forall(s, o) \in S^a \times O^a$, if (s, o) validates $\langle s_i, p, o_i\rangle$, then $s \preceq s_i$ (1) and $o \preceq o_i$ (2).

Finer-granularity. If $s_i \preceq s_j$ and $o_i \preceq o_j$, from (1) and (2) by Lemma 4, we get $s \preceq s_j$ and $o \preceq o_j$. By Definition 9, $\langle s_j, p, o_j\rangle$ is validated by (s, o). By Definition 10, $\langle s_i, p, o_i\rangle \preceq \langle s_j, p, o_j\rangle$.

Conflict. If $s_i =_c s_j$ and $o_i \nparallel o_j$, from (2) by Lemma 6, we get $o \nparallel o_j$. Given $s_i =_c s_j$, (s_i, o) cannot validate $\langle s_j, p, o_j\rangle$, so any pair of value that validates $\langle s_i, p, o_i\rangle$ cannot validate $\langle s_j, p, o_j\rangle$. By Definition 10, $\langle s_i, p, o_i\rangle \nparallel \langle s_j, p, o_j\rangle$. In the same way we can prove that when $o_i =_c o_j$ and $s_i \nparallel s_j$, these two context predicates also conflict.

Overlap. Given that $s_i =_c s_j$ and $o_i \bowtie o_j$, there exists an abstract value $o_l \in O^a$ such that $\mu(o_l) = \mu(o_i) \cap \mu(o_j) \neq \emptyset$, so $o_l \preceq o_i$ and $o_l \preceq o_j$. So (s_i, o_l) validates both $\langle s_i, p, o_i\rangle$ and $\langle s_j, p, o_j\rangle$ by Definition 9. Also by Definition 2, $o_i \bowtie o_j$ implies that there exists o_m and $o_n \in O^a$ such that $\mu(o_m) = \mu(o_i) \setminus \mu(o_j) \neq \emptyset$ and $\mu(o_n) = \mu(o_j) \setminus \mu(o_i) \neq \emptyset$, where $\setminus$ is the set minus operator. $\forall o_{m'} \preceq o_m$, $(s_i, o_{m'})$ validates $\langle s_i, p, o_i\rangle$ but not $\langle s_j, p, o_j\rangle$. Similarly, $\forall o_{n'} \preceq o_n$, $(s_j, o_{n'})$ validates $\langle s_j, p, o_j\rangle$ but not $\langle s_i, p, o_i\rangle$. By Definition 10, $\langle s_i, p, o_i\rangle \bowtie \langle s_j, p, o_j\rangle$. In the same way, we can prove that when $o_i =_c o_j$ and $s_i \bowtie s_j$, they overlap. □

4.3. Applications of relationships between context predicates

We use Lemma 11 to uncover knowledge about the relations between context predicates from the relations between the abstract values in each information dimension. In Fig. 6, we specify *finer-grained* relationships in the Person and Location dimensions: (1) `Bob` is one of `houseOccupants` and (2) `livingRoom` is spatially contained by `house`. Using Lemma 11, we can derive well-structured but implicit relationships between the four possible context predicates formed by these abstract values, as shown on the right-hand side of Fig. 6.

The derived knowledge aids developers in defining application actions using more general concepts. For example in a house emergency application a "calling ambulance" action may trigger if sensors detect an occupant in the house ⟨`houseOccupants, locatedIn, house`⟩ when a fire is detected. Our approach ensures that the assertion of any finer-grained context predicate will validate this more-general predicate (see Fig. 6), leading to the execution of the corresponding action.

The relations between context predicates are also useful in detecting inconsistency between context. Inconsistency between context is a well-known issue in activity recognition research; that is, when two pieces of context report two conflicting states at the same time. For example, ⟨⟨{`Bob`}, `locatedIn`, `bedroom`⟩, [`2006-08-23T23:48:43Z, 2006-08-23T23:48:45Z`]⟩ is inconsistent with ⟨⟨{`Bob`}, `locatedIn`, `foyer` ⟩, [`2006-08-23T23:48:44Z, 2006-08-23T23:48:46Z`]⟩. Our model supports a generic way to write an inconsistency detection rule that we may apply uniformly to any context; i.e., given two pieces of context, if their context predicates conflict while their temporal values do not then these two pieces of context are inconsistent:

Let $\langle\langle s_i, p, o_i\rangle, t_i\rangle$ *and* $\langle\langle s_j, p, o_j\rangle, t_j\rangle$ *be two instantiated context predicates that share the same relation p. They are called inconsistent if* $\neg(t_i \nparallel t_j)$ *and* $\langle s_i, p, o_i\rangle \nparallel \langle s_j, p, o_j\rangle$.

Whatever the root cause of inconsistency, be it sensor failure or some other problem such as a break-in accident, inconsistency detection provides useful information that can aid the smooth running of systems and applications. Beyond detecting inconsistency, we can base quantification of the uncertainty in context on their underlying semantics with ground values, and use an uncertainty resolving technique, such as Fuzzy Logic, to integrate the inconsistent context [36].

5. Activity model

Activities, or situations, are a higher-level concept that represents states of affair that are interesting to applications [37]. Here, we consider activities as a term that indicates a user behaviour or a state relevant to the user, such as "watching

TV", "using a computer", or "making coffee". We define activities on a logical description of context predicates. When the information in the environment satisfies an activity description, we assume the user is engaging in that activity. In the following section, we explore the semantics of activities based on the semantics of the context predicates that define them.

5.1. *Ground activities*

To begin, we analyse activities as another dimension of information using the same set-theoretic approach adopted above. As with other dimensions, there exists a ground value set for the Activity dimension, which is a collection of activity terms that experts or developers define. This ground value set is not physically measurable as those in other dimensions are. Additionally, what one perspective considers as a ground activity, another may not [23]. For example, one perspective may view "making coffee" as a ground activity, while from another we may break down this activity into finer-grained activities such as "boiling water", "retrieving coffee", and "retrieving cups". A general principle to define a ground activity set is that they should be the finest granularity of activities identifiable by available sensor information.

We relate each ground activity to a collection of context predicates using a derivation rule. For example, Roy et al. [27] define a "watching TV" activity as follows:
$\langle \texttt{?user}, \texttt{locatedIn}, \texttt{livingRoom} \rangle \wedge \langle \texttt{TVSet}, \texttt{locatedIn}, \texttt{livingRoom} \rangle \wedge$
$\langle \texttt{TVSet, isTurnedOn, true} \rangle \Rightarrow \langle \texttt{?user, engagesIn,} \{\text{"watching TV"}\} \rangle$
We can infer knowledge about the relationships between activities from their context predicates. Intuitively, if an activity is finer-grained than another activity, then when we consider a user to engage in the former activity then he is also engaging in the latter activity. If two activities conflict with each other, then a user cannot engage in both at the same time, like "lying down" and "making coffee". If two activities overlap then the user can engage in both at the same time, for example "watching TV" and "using a computer".

Definition 12. We define a ground activity on *le*, where *le* is a logical expression that takes a collection of context predicates as input and applies logical operators on them. Given two ground activities v_i^g and v_j^g defined on le_i and le_j respectively,

- v_i^g is finer-granularity than v_j^g, denoted as $v_i^g \preceq v_j^g$, iff $le_i \vdash le_j$, where $\vdash$ is the logical entailment;
- v_i^g conflicts with v_j^g, denoted as $v_i^g \nparallel v_j^g$, iff $le_i \wedge le_j = \texttt{FALSE}$;
- v_i^g overlaps with v_j^g, denoted as $v_i^g \bowtie v_j^g$, iff $le_i \nvdash le_j$, $le_j \nvdash le_i$, and there exists a logical expression le' such that $le' \neq \texttt{FALSE}$, $le' \vdash le_i$, and $le' \vdash le_j$.

With the relationships between context predicates and the above definition, we can infer the relationships between ground activities. However, we note that the complexity of comparing logical expressions computationally varies with different systems. For example, the type of logical operators supported in different reasoning engines will have a major effect on their evaluation, possibly requiring customised software to perform this task.

5.2. *Abstract activities*

Compared with other dimensions of information, activities are higher-level concepts with richer semantics, consisting of *generalisation* and *composition* relations. Correspondingly, we propose two ways to define abstract activities. A generalised activity means that we consider any of its ground activities as a *type of* it [38]. For example, if we define a generalised activity "working" on the set of activities {"doing paperwork", "using a desktop", "reading"}, then any of these activities is a type of the "working" activity. A derivation rule of a generalised activity is a disjunction of the logical descriptions of all its corresponding ground activities.

A composite activity contains all of its ground activities [35]. For example, if we define an "actively watching TV while drinking" activity on {"actively watching TV", "drinking"} [39], then these activities form part of the composite activity, but only when we recognise all do we recognise the composite activity. Accordingly, a derivation rule of a composite activity is a conjunction of the logical descriptions of all its corresponding ground activities.

Generalised and composite activities provide two ways for developers to flexibly define new higher-level activities. Defining a derivation rule for an activity can be challenging, involving significant knowledge engineering effort [40], or needing a large number of training data for machine learning techniques to generate rules [39]. We provide a methodology to allow developers to structurally analyse relationships from the finest-grained activities, and to generate other activities using different semantics to meet requirements of different applications.

We can infer relationships in generalised and composite activities from their corresponding ground activities. Take an example using generalised activities. If we define a generalised activity "information or leisure" on {"watching TV", "reading", "listening to music", "using a computer"}, this "information or leisure" activity overlaps with the above "working" activity if the ground activity "using a desktop" is finer-grained than the ground activity "using a computer". That is, when the user is using his desktop, he may be working or entertaining. The following lemmas present how to infer relationships between generalised and composite activities. We list their proofs in Appendices A and B.

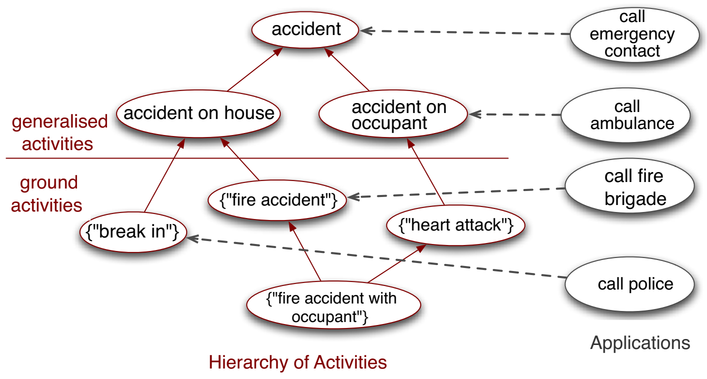


Fig. 7. An example of defining applications on a hierarchy of activities.

Lemma 13. *Let V^a and V^g be a generalised abstract and ground activity set and $\mu : V^a \rightarrow \mathcal{P}(V^g)$ be the mapping function. Given any two generalised activities v_i^a and $v_j^a \in V^a$,*

- $v_i^a \preceq v_j^a$, *if* $\forall v_k^g \in \mu(v_i^a)$ *and* $\exists v_l^g \in \mu(v_j^a)$, $v_k^g \preceq v_l^g$;
- $v_i^a \nparallel v_j^a$, *if* $\forall v_k^g \in \mu(v_i^a)$ *and* $\forall v_l^g \in \mu(v_j^a)$, $v_k^g \nparallel v_l^g$;
- $v_i^a \bowtie v_j^a$, *if* $\exists v_k^g \in \mu(v_i^a)$ *and* $\exists v_l^g \in \mu(v_j^a)$, (1) $v_k^g \bowtie v_l^g$; *or* (2) $v_k^g \preceq v_l^g$ *or* $v_l^g \preceq v_k^g$, *with the following condition* $\exists v_{k'}^g (\neq v_k^g) \in \mu(v_i^a), \forall v_{l'}^g \in \mu(v_j^a), v_{k'}^g \npreceq v_{l'}^g$, *and* $\exists v_{l'}^g (\neq v_l^g) \in \mu(v_j^a), \forall v_{k'}^g \in \mu(v_i^a), v_{l'}^g \npreceq v_{k'}^g$.

Lemma 14. *Let V^a and V^g be a composite abstract and ground activity set and $\mu : V^a \rightarrow \mathcal{P}(V^g)$ be the mapping function. Given any two composite activities v_i^a and $v_j^a \in V^a$,*

- $v_i^a \preceq v_j^a$, *if* $\forall v_k^g \in \mu(v_j^a), \exists v_l^g \in \mu(v_i^a)$ *such that* $v_l^g \preceq v_k^g$;
- $v_i^a \nparallel v_j^a$, *if* $\exists v_k^g \in \mu(v_i^a)$ *and* $\exists v_l^g \in \mu(v_j^a)$, $v_k^g \nparallel v_l^g$.
- $v_i^a \bowtie v_j^a$, *if* (1) $\forall v_k^g \in \mu(v_i^a)$ *and* $\forall v_l^g \in \mu(v_j^a)$, $\neg(v_k^g \nparallel v_l^g)$, *and* (2) $\exists v_{k'}^g \in \mu(v_i^a)$ *and* $\exists v_{l'}^g \in \mu(v_j^a)$, $v_{k'}^g \bowtie v_{l'}^g$ *or* $v_{k'}^g \preceq v_{l'}^g$ *or* $v_{l'}^g \preceq v_{k'}^g$, *and* (3) $\exists v_{k''}^g (\neq v_{k'}^g) \in \mu(v_i^a)$ *and* $\exists v_{l''}^g (\neq v_{l'}^g) \in \mu(v_j^a)$, $v_{k''}^g \npreceq v_{l''}^g$ *and* $v_{l''}^g \npreceq v_{k''}^g$.

5.3. Applications of semantics in activities

Exploration of the semantics between ground activities can help developers to improve their derivation rules. For example, if we independently define two similar activities, such as "eating" and "drinking", there is a chance that imprecision in one or the other of their derivation rules may result in the inference that "drinking" is finer-grained than "eating". If an application requires that it be possible to distinguish between these two activities, then the inferred finer-grained relationship between them serves as a prompting for developers to provide more precise rules.

We can also use the semantics to check the consistency of relationships between activities. If developers understand the activities well; that is, which cannot co-occur, then they can use their expert knowledge to check the conflicting and overlapping relationships that a reasoner infers. We detect inconsistency (1) if the reasoner infers any two activities that the developer knows cannot co-occur as overlapping or if the model declares one as finer-grained than the other; and (2) if the reasoner infers any two activities that the developer knows can co-occur as being in conflict. In either of these cases, the developer can check and update the activity derivation rules to resolve the inconsistency.

A structured activity hierarchy can also augment the knowledge produced by other techniques; for example, machine learning algorithms. When the reasoner infers that a user is engaging in one activity, we can use the activity hierarchy to tell (1) which additional activities the user is engaging in (from the granularity relationship); (2) which activities the user cannot be engaging in (from the conflicting relationship) [21]; and (3) which activities the user may also be engaging in (from the overlapping relationship). These rich inferences about activities can act as a guideline for application design. Developers should not define applications that negate the effect of each other on non-conflicting activities. An activity automatically inherits any applications that developers specify against its coarser-grained activities.

Take the example in Fig. 7 that launches phone calling actions to different situations in a smart home. On the left-hand side is a hierarchy of activities, where ground activities include "break in", "fire accident", "fire accident with occupant", and "heart attack". Among them, "fire accident with occupant" is finer-grained than "fire accident". Generalised activities include "accident on house", "accident on occupant", and "accident". On the right-hand side are a set of phone calling actions defined on the occurrence of an activity:

- when any accident occurs, call the house occupants' emergency contact;
- when any accident occurs involving an occupant, call the ambulance service;

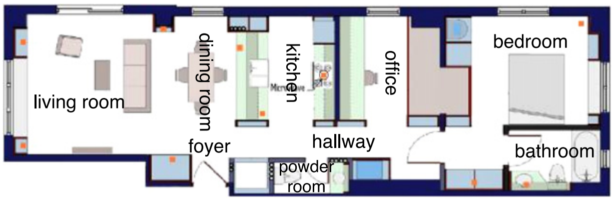


Fig. 8. The space layout of the PlaceLab.

- when any fire occurs, call the fire brigade;
- when a break in occurs, call the police.

As we show in Fig. 7, when there is a fire accident with a house occupant, then we also recognise all its coarser-grained activities. Thus we launch the applications specified on each, including calling the fire brigade (from "fire accident"), the ambulance (from "accident on occupant"), and the emergency contact (from "accident"). The hierarchy of activities can help developers to design applications in a more concise and accurate way, compared to the way that developers currently assign actions to individual non-related situations or activities.

6. Implementation and demonstration

This section demonstrates the application of the top-level ontology model to a real-world smart home scenario. It describes an implementation of the ontology model through a set of rules and software components that operate over it. We use OWL DL to describe the model as it provides a distributed formal model with advantages over alternative modelling approaches [16]. As a description logic, OWL DL also lends itself to the application of reasoning. We implement the model using the Named Graphs for Jena (NG4J) software library.[2] The NG4J extension to Jena supports modelling named graphs, which, as we will discuss, allows us to easily model relations between context predicates. These are not the only formalisms and tools applicable, and the reader may implement our model using other formalisms and software of their choice. The interested reader can download the ontologies and rules we describe in this section from http://ontonym.org.

6.1. Real-world smart home—PlaceLab

The *PlaceLab* smart home environment [3], developed by MIT and the TIAX company, is a residential "living laboratory" where it is possible to test and evaluate new technologies and design concepts in the context of everyday living. As part of the project, researchers released a data set (known as *PLCouple1*[3]) that captures the interactions of a married couple over two weeks' residence. The data covers a broad range of commonly used smart home sensors and consequently we choose the PlaceLab to demonstrate the use of our ontology.

The PlaceLab consists of a living room, a dining room, a kitchen, an office, a bedroom, a bathroom, and a powder room, as shown in Fig. 8. The PlaceLab is instrumented with over nine hundred sensors, including: infra-red sensors to detect motion in different rooms, object motion sensors to detect access and movement of everyday articles such as the television remote control, sensors to monitor the usage of electrical current, water and gas, RFID sensors, and switch sensors that sense the open and closed states of doors. It requires a considerable amount of knowledge engineering effort to describe each aspect of information sensed by these sensors, and is additionally challenging to define relations between these aspects of information.

We now show that with the help of generic rules our ontology model allows developers to specify a smaller percentage of the knowledge, deriving the remainder automatically. We also demonstrate how we use the common semantics in contexts to define a generic approach to detect inconsistency between inherently imperfect sensor data.

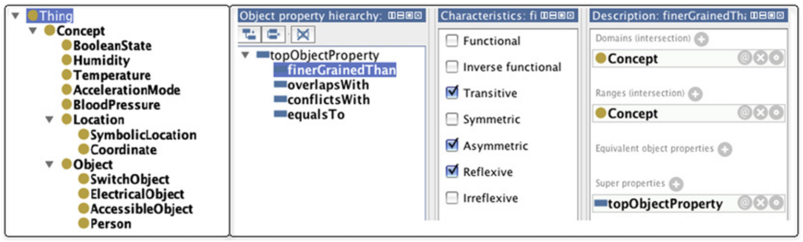
The PlaceLab is equipped with an audio-visual recording infrastructure that records the activities of the participants living in the PlaceLab during the data collection period. The monitored activities include actively watching TV or movies, doing paperwork, using a computer, reading, preparing a meal, drinking, and hygiene. The participants can perform multiple activities simultaneously; e.g., having a drink while watching TV. We demonstrate how to compose these ground activities into generalised and composite abstract activities and derive the semantic relationships between them. We also show how these relationships facilitate the design of applications.

6.2. Concept ontology

We define a class, `Concept`, as a superclass of all the dimensions of information, including `Location`, `Humidity`, `BooleanState`, and `Temperature`. Fig. 9 shows the screenshots of the concept ontology in *Protégé* [41]. Four

[2] Named Graphs for Jena: http://www4.wiwiss.fu-berlin.de/bizer/ng4j/, while the rules operate using the general purpose rule engine feature of the Jena software. Ontologies in this section are represented in TriG, a syntax to serialise named graphs: http://www4.wiwiss.fu-berlin.de/bizer/TriG/.

[3] The PLCouple1 data set can be downloaded from http://web.media.mit.edu/~intille/data/PLCouple1/.



- [ ] Functional
- [ ] Inverse functional
- [x] Transitive
- [ ] Symmetric
- [x] Asymmetric
- [x] Reflexive
- [ ] Irreflexive

Domains (intersection): Concept
Ranges (intersection): Concept
Equivalent object properties
Super properties: topObjectProperty

(a) Class structure of the domain concept ontology (b) Object properties of the domain concept ontology

Fig. 9. The concept ontology.

predicates describe the common semantic relations between any two abstract values in one dimension: `equals`, `finerGrainedThan`, `conflictsWith`, and `overlapsWith`. The basic meta-properties of these relationships as we describe in Lemmas 3–5 are part of the OWL specification, and we specify these as characteristics of the corresponding object properties, as we show in Fig. 9(b). We implement the more complex properties in Lemmas 6–8 as rules in Listing 1.

Listing 1 Generic rules implemented for Lemmas 6–8.(1)

```
# Lemma 6:
[conflictInheritance: (?a concept:conflictsWith {?}b), (?c concept:finerGrainedThan ?a),
    (?d concept:finerGrainedThan {?}b) -> (?c concept:conflictsWith ?d)]

# Lemma 7:
[sharedGranularityImpliesOverlap: (?a concept:finerGrainedThan {?}b),
    (?a concept:finerGrainedThan ?c), noValue({?}b concept:finerGrainedThan ?c),
    noValue(?c concept:finerGrainedThan{?}b) -> ({?}b concept:overlapsWith ?c)]

# Lemma 8.(1):
[onlyConflict: (?a concept:conflictsWith {?}b) <- (?a rdf:type ?ta), ({?}b rdf:type ?tb),
    equal(?ta, ?tb), noValue(?a concept:finerGrainedThan {?}b),
    noValue({?}b concept:finerGrainedThan ?a), noValue(?a concept:overlapsWith {?}b)]
```

Each concrete dimension of information has its own structure and relationships. In order to exploit the common semantic relationships we implement above, the application developer can either associate the generic relationships with the relationships specific to each dimension or provide a mapping from the concrete dimension of information to the generic form. Listing 2 shows an example of defining the generic relationships from the spatial relationships between location concepts.

Listing 2 Defining the generic relationships from the spatial relationships

```
# Granularity: if a location a contains another location b, then b is finer-grained than a.
[granularityLoc: (?a location:contains {?}b) -> ({?}b concept:finerGrainedThan ?a)]

# conflicts: if a location a is disjoint with another location b, then a conflicts with b.
[conlfictLoc: (?a location:disjointWith {?}b) -> (?a concept:conflictsWith {?}b)]
```

Listing 3 gives an example of defining an abstract temperature value `hot`. The naming and definition of concepts can vary with different environments, users, and applications, without risk of conflict as we identify each by a URI. We define the abstract value on a set of ground values on the Celsius scale, using the `temp:lowerBound` and `temp:upperBound` properties to state the lower and upper bounds of each respectively. Symbolic terms in other dimensions of information can be similarly defined, such as `high`, `low`, and `off` in the usage of electrical current. Next, Listing 4 shows how to write a rule to map the containment relationship between two temperature ranges to the `finerGrainedThan` relationship. Mappings of the other generic relationships are similarly implemented. Once achieved, we may infer that `warm` is finer-grained than `hot`, and that the concepts `warm` and `hot` both conflict with the concept of `cold`, as shown in Fig. 3.

In the above examples, we have demonstrated two principal methods to define common semantics in individual dimensions of information, which link the top-level ontology with domain or application ontologies. One of the advantages of defining the common semantic relationships between domain concepts is that we can use generic rules to derive new knowledge; this reduces the knowledge engineering effort required of developers. There typically exist $O(n^2)$ of these relationships among n concepts, given that any two concepts can either be conflicting, overlapping, or at different levels of granularity. Using these rules, developers need only to define the immediately finer-grained relationships between concepts

Listing 3 Defining concepts within the temperature domain

```
  :hot a temp:SymbolicTemperature ;
    temp:lowerBound
      [ a muo:QualityValue ;
        muo:numericalValue "20"^^xsd:integer ;
        muo:measuredIn ucum:degree-Celsius
      ] ;
    temp:upperBound
      [ a muo:QualityValue ;
        muo:numericalValue "50"^^xsd:integer ;
        muo:measuredIn ucum:degree-Celsius
      ].
```

Listing 4 Defining the temperature mapping rules

```
# Granularity: if the temperature range of a contains the temperature
#              range of b then b is finer-grained than a.
[granularityTemp: (?a temp:lowerBound ?alb), (?alb muo:numericalValue ?alow),
    (?a temp:upperBound ?aub), (?aub muo:numericalValue ?ahigh),
    ({?}b temp:lowerBound {?}blb), ({?}blb muo:numericalValue {?}blow),
    ({?}b temp:upperBound {?}bub), ({?}bub muo:numericalValue {?}bhigh),
    le(?alow, {?}blow), ge(?ahigh, {?}bhigh) -> ({?}b concept:finerGrainedThan ?a)]
```

in the model, that is $O(n)$. Take an example of encoding the PlaceLab map shown in Fig. 8, which consists of 10 symbolic locations (including the house itself). The left-hand side of Fig. 4 lists the 10 immediately containment relationships between two locations. A reasoner uses the rules to infer the remaining 78 relationships, some of which are listed in the right-hand side of Fig. 4.

Listing 5 Derive the conflicting relationship between accessible objects

```
[conflictWithobj_loc: (?a location:locatedIn ?al), ({?}b location:locatedIn {?}bl),
    (?al concept:conflictsWith {?}bl) -> (?a concept:conflictsWith {?}b)]
```

We also use common semantic relationships to derive new knowledge across dimensions of information. For example, we can use the inferred relationships between the locations to further derive a conflicting relationship between two static objects by assuming that they cannot be simultaneously accessed by the same user if they are not co-located. This is achieved in one simple rule as defined in Listing 5. Using this rule, we derive 32 170 conflicting relationships between 605 objects that are extracted from the RFID sensors and the object motion sensors in the PlaceLab sensor configuration file. These examples show that the common semantics allow a generic way to specify domain knowledge, which can help to reduce the load of knowledge engineering effort.

6.3. *Context ontology*

In Definition 9 we define a context predicate to be a triple consisting of a subject, a predicate, and an object, where the subject and object are abstract values in two domains. For example,

```
:bob location:locatedIn :diningRoom.
```

We express the validation of context predicates, which we base on the occurrence of finer-grained context predicates in the model (also Definition 9), using the rule in Listing 6. This rule allows us, for example, to infer the statement

```
:bob location:locatedIn :house.
```

as `:diningRoom` is finer-grained than `:house` and by definition, `:bob` is finer-grained than `:bob`.

Listing 6 The rule for validating context predicates

```
# Validation rule: infers a context predicate based on the presence of finer
#                  grained context predicates in the model
[contextPredicateValidation: (?p rdf:type contextPredicate:RelationProperty),
(?a ?p {?}b) (?a concept:finerGrainedThan ?y), ({?}b concept:finerGrainedThan ?z) -> (?y ?p ?z)]
```

We make use of named graphs to express the possible relations between context predicates (see Lemma 11) in our model. A named graph is a collection of one or more context predicates, identified by a URI. Named graphs provide a pointer to a statement or a set of statements, and we express a relation between two (or more) statements as a relation between the graphs that contain them. For example if a named graph `G1` contains context predicates `C1` and `C2`, and a named graph `G2` contains a context predicate `C3`, and we wish to express that both `C1` and `C2` conflict `C3`, we do this by writing `G1` conflicting with `G2`. Context predicates may belong to more than one graph if we need to describe multiple relationships. Listing 7 models that Bob is reported in the bedroom and the bathroom (graph `g:G1`), this conflicts (graph `g:G3`) the context predicate that describes Bob's location as within the foyer (graph `g:G2`). We represent the other two relationships between context predicates in the same way.

Listing 7 Expressing the conflicts between context predicates

```
g:G1 {
    :bob location:locatedIn :bedroom.
    :bob location:locatedIn :bathroom.
    }
g:G2 {
    :bob location:locatedIn :foyer .
    }
g:G3 {
    g:G1 contextPredicate:conflictsWith g:G2.
    }
```

Although it is straightforward to represent relations between context predicates, we are aware of no reasoner for OWL models that supports named graphs. Therefore, as a work around, we extend the standard Jena reasoner with a number of functions that allow us to work with relations between graphs. To illustrate, consider the task of identifying when two pieces of context report conflicting states at the same time. Following Listing 7, the graphs `g:G4` and `g:G5` in Listing 8 present an example where sensors report Bob to be in the `bathroom, bedroom`, and `foyer` during the overlapping periods. To detect the inconsistency in the model, we must compare not only the context predicates, but also their validity time. Listing 9 shows the rule used to detect this form of inconsistency. Its implementation makes use of two external methods: `temporalConflict(...)` takes the components of the two predicates we are interested in and compares the timestamps associated with each statement, while if we detect a conflict, the `conflictDetected(...)` method asserts a conflict between the graphs that contain the conflicting statements. The representation of this model is similar to that in Listing 8.

Listing 8 Time extension on context predicates

```
g:G4 {
    :G1 temporal:validDuring
        [a owltime:Interval ;
        owltime: hasBeginning;
         [a owltime:Instant ;
            owltime:inXSDDateTime "2006-08-23T23:48:43Z"] ;
        owltime: hasEnd;
         [a owltime:Instant ;
            owltime:inXSDDateTime "2006-08-23T23:48:45Z"] ;
    }
g:G5 {
    :G2 temporal:validDuring
        [a owltime:Interval ;
        owltime: hasBeginning;
         [a owltime:Instant ;
            owltime:inXSDDateTime "2006-08-23T23:48:44Z"] ;
        owltime: hasEnd;
         [a owltime:Instant ;
            owltime:inXSDDateTime "2006-08-23T23:48:46Z"] ;
    }
g:G6 {
    :G4 context:conflictsWith :G5 .
    }
```

To prove the concept, we take one day's infra-red sensor data from the PlaceLab data set, which reports possible motions in each room between 18:00:00 and 23:59:57 on 23rd August 2006. We assume that a user cannot be in two *conflicting*

Listing 9 A portion of the rule to detect conflicts between context predicates

```
[conflictDetectionOnObject:  ..., (?a ?p {?}b),  (?a ?p ?c),
    ({?}b concept:conflictsWith ?c), temporalConflict(?a, ?p, {?}b, ?c)
    -> conflictDetected(?a, ?p, {?}b, ?c)]
```

rooms at the same time. Applying the above detection rule, we detect 178 inconsistent pairs out of 2316 pieces of sensor data.[4]

6.4. Activity ontology

As we specify how we compose ground activities from context predicates externally from the model, the process of inferring the relationships between them is carried out by the custom software. The software works by taking each pair of activities in turn and performing a pairwise comparison of all of their constituent context predicates to discover the relations defined in Definition 12. For example, Listing 10 defines derivation rules for the two ground activities "actively watching TV or movies" and "watching TV".

Listing 10 Derivation rules for two "watching TV" ground activities

```
[activelyWatchTVRule: (?p location:locatedIn :livingRoom),
    (tv eleObject:isTurnedOn true), (?p accObject:accesses :couchInLivingRoom)
    -> (?p activity:engagesIn :activelyWatchTV)]

[watchTVRule: (?p location:locatedIn :house), (tv eleObject:isTurnedOn true),
    -> (?p person:engagesIn :watchTV)]
```

The above definitions differ by a context predicate `(?p location:locatedIn :livingRoom)` and `(?p location:locatedIn :house)`. By inspection of the model we know that `livingRoom` is finer-grained than `house`, thus we can infer that the latter context predicate entails the former and therefore by Definition 12. By evaluating the other parts of logical expressions on these two activities and the semantics of the logical connectives used in them, we can infer that the activity "actively watching TV or movies" is finer-grained than "watching TV". We express the resultant relation as part of the model:

```
:activelyWatchTV activity:finerGrainedThan :watchTV.
```

We can compose ground activities into high-level abstract activities. Listing 11 gives two examples that define generalised and composite activities from ground activities. We use the properties `generalisedFrom` and `composedOf` to represent the relationships between abstract activities discussed in Section 5. We use another software add-on to analyse the relationships between activities using Lemmas 13 and 14 and add the resulting relationships to the model. The implementation of these lemmas are straightforward, and Algorithm 1 shows an example where we derive the finer-grained relationships from two generalised abstract activities.

Listing 11 An example of defining abstract activities from ground activities

```
:working activity:generalisedFrom :usingDesktop, :reading, :doingPaperWork .
:informationOrLeisure activity:generalisedFrom :watchingTV, :reading, :listeningToMusic,
     :usingComputer .
:activelyWatchingTVWhileDrinking activity:composedOf :activelyWatchTV, :drinking .
:bathingWhileDrinking activity:composedOf :bathing, :drinking .
```

Using the implemented lemmas, we can infer the relationships between the abstract activities in Listing 11 as follows:

```
:working activity:overlapsWith :informationOrLeisure .
:activelyWatchTVWhileDrinking activity:conflictsWith :bathingWhileDrinking .
```

The rules in Listing 12 apply the relationships between abstract activities in recognising activities. These rules guarantee that when a developer defines an application on a finer-grained activity it will take place on its coarser-grained activities. In this way, developers may define applications on activities in a hierarchical and traceable way. The rules also prompt the system when conflicting activities are inferred at the same time. Finally, we specify application actions as rules that, when triggered, call appropriate methods in application code. For example, Listing 13 shows how we express the application rules from Section 5.3 that call the fire brigade and emergency contact.

[4] In the PlaceLab data set the location sensors are not person-specific, therefore inconsistent pairs imply that either sensor data is inaccurate or that the participants are in different rooms during these times.

ARTICLE IN PRESS

Algorithm 1: The algorithm of deriving the finer-grained relationships on two generalised activities

```
input : Two abstract activities and their associated ground activities $(v_i^a, \mu(v_i^a)), (v_j^a, \mu(v_j^a))$
isFinerGrainedThan = true;
for $v_k^g \in \mu(v_i^a)$ do
    foundAFinerGround = false;
    for $v_l^g \in \mu(v_j^a)$ do
        if $v_k^g$.finerGrainedThan($v_l^g$) then
            foundAFinerGround = true;
            break;
    if !foundAFinerGround then
        isFinerGrainedThan = false;
        break;
if isFinerGrainedThan then
    $v_i^a$.addFinerGrainedActivity($v_j^a$)
```

Listing 12 Rules to relay specified relationships between abstract activities in activity recognition

```
# Granularity: if an activity a is inferred and a is finer-grained than
               another activity b, then b will be inferred, too.
[granularityAct: (?p activity:engagesIn ?a), (?a activity:finerGrainedThan {?}b)
    -> (?p activity:engagesIn {?}b)]

#Conflict: if an activity a is inferred and its conflicting activity b is also
           inferred, then an inconsistency is detected.
[conflictAct: (?p activity:engagesIn ?a), (?p activity:engagesIn {?}b),
    (?a activity:conflictsWith {?}b) -> activityConflictDetected(?p, ?a, {?}b)]
```

Listing 13 A set of application rules defined on the occurrence of activities

```
[generalAccident: (?p activity:engagesIn :accident) -> callEmergencyContact()]

[fireAccident: (?p activity:engagesIn :fire) -> callFireBrigade()]
```

6.5. Discussion

In this section we discuss the utility of our top-level ontology model, and the practical issues of implementing and using it from the perspective of software developers.

6.5.1. Formality

The proposed top-level ontology model partitions the problem of how to transform raw sensor data into application-level data in a set of well defined steps. Each step is structured, and governed by a set of generic rules that describe how to transform or augment knowledge from the previous step into the necessary knowledge required by the next step in the process. Whereas previous approaches to managing sensed data operate over one big general purpose model with no strict rules on how one piece of information relates to another, our model supports the correlation of knowledge across all layers in a sound reasoning schema.

6.5.2. Consistency

Relations in this model are fully integrated: from ground values at one extreme, to complex activities at the other. Therefore, the knowledge the reasoner automatically infers from that which is explicitly stated can inform developers about inconsistencies between sensed context, and across activity derivation rules. This is useful, both in the development and debugging of systems. The formal semantics also provide a form of provenance; for example, when we find two activities to be inconsistent, we can pinpoint the specific context predicates that are the cause of the inconsistency. Not only is this useful to the developer, but runtime resolution techniques may be available in some instances that can resolve inconsistencies given this information as an input.

6.5.3. Rich semantics

The top-level ontology starts from the classic set theory by analysing ground value ranges in each dimension of an information space. It extracts four basic structural semantics in concepts, which we also study in relational compositions of concepts; that is, contexts and activities. We demonstrate that these common semantics across different levels of information play an important role in aiding tasks particular to pervasive computing, including detecting inconsistency of sensor inputs, and designing applications in a hierarchical and traceable manner.

The richness of the semantics in this ontology model, however, can be improved by incorporating the semantics of predicates and temporal semantics. So far, we have not taken into consideration the relationships that can exist between predicates, such as the inverse relationship of `locatedIn` and `notLocatedIn`. Adding such semantics to our model will increase the amount of knowledge that we will be able to infer automatically, and enhance the expressiveness of our model.

Temporal semantics have been identified as another principal feature of activities [42,1]. Augusto et al. [43] take the initiative in introducing temporal operators in defining derivation rules in recognising activities in smart homes. Bui et al. [44] propose a Hidden Permutation Model to learn high-level activities from temporally ordered lower-level activities. To better support activity recognition techniques and activity-aware applications, our model needs to be extended with temporal semantics.

6.5.4. Engineering effort

From the perspective of an implementer of the model (i.e., the data storage and reasoning aspects), the different parts of our model correspond to a set of orthogonal layers, each of which the developer may optimise for the set of tasks each carries out. Application developers only need to define abstract values in each information dimension, specify the necessary amount of domain knowledge on them allowing the reasoner to derive further relationships, and define activity derivation rules. After developers carry out these processes in an environment, the information is reusable across multiple applications. Over time, application developer effort decreases as previously specified domain concepts are reused.

We contend that this top-level ontology model complements, rather than replaces, domain and application ontologies. Developers still need to define structures and relationships of concepts and specify derivation rules for activities in domain and application ontologies. These processes, especially specifying derivation rules, still involve significant engineering effort, while hand-coding rules in ontology- or logic-based models is common. Our ontology model can assist in reducing, rather than eliminating, the effort in engineering domain knowledge by using generic rules. As demonstrated in Section 6.2, by predefining 10 spatial relationships, we can automatically derive the other 78 spatial relationships and 32 170 conflicting relationships between the objects.

We also expect application developers to reuse existing activity derivation rules and compose new ones from those already defined so as to meet different application requirements. Not only does this keep developers from laborious coding but also facilitates knowledge reuse about activities across different applications and environments.

7. Conclusion and future work

This paper presents a top-level ontology that captures in a uniform manner the inherent semantics that different types of domain knowledge share. We model the semantics across information at different levels of abstraction and reason on them by using a sound reasoning schema. This ontology model serves as a conceptual backbone for developing ontologies that accommodate these semantics. Developers are encouraged to use provided generic rules and define their own rules to facilitate performing system-level tasks, such as checking the consistency of both context and derivation rules that describe activities, and integrating the model with statistical techniques to help build activity recognition models; wrapping these semantics around their output [45].

In the future, we will extend the ontology with the semantics of predicates and temporal semantics of abstract activities. We will define a new type of abstract activities and introduce an additional structure to represent the temporal sequences between them. Based on this information, we may infer the temporal relationships between abstract activities from their associated ground values. The implementation of the rules described in this paper is currently restricted by the lack of a reasoner for working with named graphs; requiring us to augment the Jena reasoner with custom operations. We will address this problem by integrating the Jess reasoning framework with the NG4J software library.

Acknowledgements

This work is partially supported by the FP7 FET proactive project *SAPERE—self-aware pervasive service ecosystems*, under grant no. 256873. We thank all the reviewers for their constructive comments.

Appendix A. Proof of Lemma 13: relationships between generalised abstract activities

Proof. Let a logical expression of a generalised abstract activity v^a be represented as $le_1 \vee \cdots \vee le_m$ where $le_1 \ldots le_m$ are logical expressions of ground activities on v^a. Let a logical expression of v_i^a and v_j^a be represented as $le_{i1} \vee \ldots \vee le_{im}$ and $le_{j1} \vee \cdots \vee le_{jn}$ respectively.

Finer-granularity. $\forall v_k^g \in \mu(v_i^a)$ and $\exists v_l^g \in \mu(v_j^a)$, given $v_k^g \preceq v_l^g$, we get $le_k \vdash le_l$. So $le_{i1} \vee \cdots \vee le_{im} \vdash le_{j1} \vee \cdots \vee le_{jm}$. It is possible that there exists one logical expression le_{js} $(1 \leq s \leq n)$ that multiple expressions on v_i^a entail, so for matching, le_{js} can be repeatedly used in the expression $le_{j1} \vee \cdots \vee le_{jm}$ with $le_{js} \vee \cdots \vee le_{js} = le_{js}$. If there exists any other expression on v_j^a that does not match, then $le_{j1} \vee \cdots \vee le_{jm} \vdash le_{j1} \vee \cdots \vee le_{jn}$, so $le_{i1} \vee \cdots \vee le_{im} \vdash le_{j1} \vee \cdots \vee le_{jn}$. By Definition 12, $v_i^a \preceq v_j^a$.

Conflict. $\forall v^g_k \in \mu(v^a_i)$ and $\forall v^g_l \in \mu(v^a_j)$, $v^g_k \nparallel v^g_l$, so $le_l \wedge le_k = \texttt{FALSE}$. So $le_k \wedge (le_{j1} \vee \ldots \vee le_{jn}) = \texttt{FALSE}$. $(le_{i1} \vee \ldots \vee le_{im}) \wedge (le_{j1} \vee \ldots \vee le_{jn}) = (le_{i1} \wedge (le_{j1} \vee \ldots \vee le_{jn})) \vee (le_{im} \wedge (le_{j1} \vee \ldots \vee le_{jn})) = \texttt{FALSE} \vee \cdots \vee \texttt{FALSE} = \texttt{FALSE}$. By Definition 12, $v^a_i \nparallel v^a_j$.

Overlap. $\exists v^g_k \in \mu(v^a_i)$ and $\exists v^g_l \in \mu(v^a_j)$, given $v^g_k \bowtie v^g_l$, we get $le_k \nvdash le_l$ (1), $le_l \nvdash le_k$ (2), and there exists le' such that $le' \vdash le_k$, $le' \vdash le_l$ (3).

From (1) $\Rightarrow le_{i1} \vee \cdots \vee le_{im} \nvdash le_{j1} \vee \cdots \vee le_{jn}$ (4),
From (2) $\Rightarrow le_{j1} \vee \cdots \vee le_{jn} \nvdash le_{i1} \vee \cdots \vee le_{im}$ (5),
From (3) $\Rightarrow le' \vdash le_{i1} \vee \cdots \vee le_{im}$ and $le' \vdash le_{j1} \vee \cdots \vee le_{jn}$ (6),
From (4), (5), and (6), by Definition 12 $\Rightarrow v^a_i \bowtie v^a_j$.

In another case, $v^g_k \preceq v^g_l \Rightarrow le_k \vdash le_l$ (7);
From (7) and $le_l \vdash le_{j1} \vee \ldots \vee Le_{jn} \Rightarrow le_k \vdash le_{j1} \vee \ldots \vee le_{jn}$ (8);
From (8) and $le_k \vdash le_{i1} \vee \ldots \vee le_{im} \Rightarrow le_k$ entails the logical expressions on both v^a_i and v^a_j (9).

Similarly from $v^g_l \preceq v^g_k$, we can derive that le_l entails the logical expressions on both v^a_i and v^a_j (10).

From (9) or (10), we conclude that there exists a logical expression that entails the logical expressions on both v^a_i and v^a_j. According to the rest of the condition, $\exists v^g_{k'}(\neq v^g_k) \in \mu(v^a_i), \forall v^g_{l'} \in \mu(v^a_j), v^g_{k'} \npreceq v^g_{l'}$, and $\exists v^g_{l'}(\neq v^g_l) \in \mu(v^a_j), \forall v^g_{k'} \in \mu(v^a_i), v^g_{l'} \npreceq v^g_{k'}$, we can derive $le_{i1} \vee \cdots \vee le_{im} \nvdash le_{j1} \vee \cdots \vee le_{jn}$ and $le_{j1} \vee \cdots \vee le_{jn} \nvdash le_{i1} \vee \cdots \vee le_{im}$. By Definition 12, $v^a_i \bowtie v^a_j$. Therefore, we have proved that $v^a_i \bowtie v^a_j$ under two cases. □

Appendix B. Proof to Lemma 14: relationships between composite abstract activities

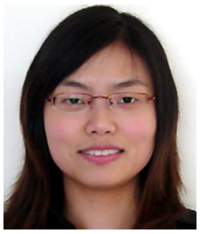
Proof. Let a logical expression of a composite abstract activity v^a be represented as $le_1 \wedge \cdots \wedge le_m$ where $le_1 \ldots le_m$ are logical expressions of ground activities on v^a. Let a logical expression of v^a_i and v^a_j be represented as $le_{i1} \wedge \cdots \wedge le_{im}$ and $le_{j1} \wedge \cdots \wedge le_{jn}$ respectively. The proof is similar to that of Lemma 13 in Appendix A. □

References

[1] D.J. Cook, S.K. Das, How smart are our environments? An updated look at the state of the art, Pervasive and Mobile Computing 3 (2) (2007) 53–73.
[2] D.J. Cook, J.C. Augusto, V.R. Jakkula, Ambient intelligence: technologies, applications, and opportunities, Pervasive and Mobile Computing 5 (4) (2009) 277–298.
[3] B. Logan, J. Healey, M. Philipose, E.M. Tapia, S.S. Intille, A long-term evaluation of sensing modalities for activity recognition, in: Ubicomp 2007: Proceedings of the Nineth International Conference on Ubiquitous Computing, Innsbruck, Austria, 2007, pp. 483–500.
[4] L. Atallah, G.-Z. Yang, The use of pervasive sensing for behaviour profiling—a survey, Pervasive and Mobile Computing (2009) 447–464.
[5] L. Chen, C. Nugent, M. Mulvenna, D. Finlay, X. Hong, Semantic smart homes: towards knowledge rich assisted living environments, Intelligent Patient Management 189 (2009) 279–296.
[6] L. Obrst, R.E. Wray, H. Liu, Ontological engineering for B2B e-commerce, in: FOIS'01: Proceedings of the International Conference on Formal Ontology in Information Systems, ACM Press, New York, NY, USA, 2001, pp. 117–126.
[7] in: Eckstein, R., Tolksdorf, R., Bizer, C. (Eds.), SWEB 2004: Proceedings of the International Workshop on Semantic Web Technologies in Electronic Business, 2004.
[8] V. Guidetti, Intelligent information integration systems: extending a lexicon ontology, Master's Thesis, Computer Science, University of Modena and Reggio Emilia, 2002.
[9] A. Varzi, L. Vieu, Formal ontology in information systems, in: FOIS'04: Proceedings of the Third International Conference on Formal Ontology in Information Systems, IOS Press, Turin, Italy, 2004.
[10] J.E. López de Vergara, V.A. Villagrá, J. Berrocal, J.I. Asensio, R. Pignaton, Semantic management: application of ontologies for the integration of management information models, in: Proceedings of the IFIP/IEEE Eighth International Symposium on Integrated Network Management, 2003, pp. 131–134.
[11] Y. Gil, E. Motta, V.R. Benjamins, M.A. Musen (Eds.), ISWC'05: Proceedings of the 4th International Semantic Web Conference, in: Lecture Notes in Computer Science, vol. 3729, Springer, Galway, Ireland, 2005.
[12] Y. Sure, J. Domingue (Eds.), ESWC'06: Proceedings of the 3rd European Semantic Web Conference on the Semantic Web: Research and Applications, in: Lecture Notes in Computer Science, vol. 4011, Springer, Budva, Montenegro, 2006.
[13] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human–Computer Studies 43 (5–6) (1995) 907–928.
[14] N. Guarino, Formal ontology and information systems, in: FOIS'98: Proceedings of the First International Conference on Formal Ontology in Information Systems, IOS Press, Trento, Italy, 1998, pp. 3–15.
[15] I. Roussaki, M. Strimpakou, N. Kalatzis, M. Anagnostou, C. Pils, Hybrid context modeling: A location-based scheme using ontologies, in: PERCOMW'06: Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops, IEEE Computer Society, Washington, DC, USA, 2006, p. 2.
[16] T. Strang, C. Linnhoff-Popien, A context modeling survey, in: Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management as Part of UbiComp 2004, Nottingham, England, 2004.
[17] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers, K. De Bosschere, Towards an extensible context ontology for ambient intelligence, Ambient intelligence 3295 (2004) 148–159.
[18] J. Ye, L. Coyle, S. Dobson, P. Nixon, Ontology-based models in pervasive computing systems, Knowledge Engineering Review 22 (2007) 315–347.
[19] T.R. Gruber, Toward principles for the design of ontologies used for knowledge sharing, International Journal of Human–Computer Studies 43 (1995) 907–928. http://portal.acm.org/citation.cfm?id=219666.219701.
[20] N. Guarino, P. Giaretta, Ontologies and knowledge bases, in: N.J. Mars (Ed.), Towards Very Large Knowledge Bases—Knowledge Building and Knowledge Sharing 1995, IOS Press, Amsterdam, 1995, pp. 25–32.

ARTICLE IN PRESS

[21] H.R. Schmidtke, W. Woo, Towards ontology-based formal verification methods for context aware systems, in: Pervasive'09: Proceedings of the 7th International Conference on Pervasive Computing, Springer-Verlag, Nara, Japan, 2009, pp. 309–326.
[22] H. Chen, T. Finin, A. Joshi, Semantic web in the context broker architecture, in: PERCOM'04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications, 2004, pp. 277–286.
[23] R. Nevatia, J. Hobbs, B. Bolles, An ontology for video event representation, in: CVPRW'04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop, vol. 7, IEEE Computer Society, Washington, DC, USA, 2004, p. 119.
[24] T. Gu, H.K. Pung, D.Q. Zhang, A service-oriented middleware for building context-aware services, Journal of Network and Computer Applications 28 (1) (2005) 1–18.
[25] E.M. Tapia, T. Choudhury, M. Philipose, Building reliable activity models using hierarchical shrinkage and mined ontology, in: Pervasive'06: Proceedings of the International Conferences on Pervasive Computing, 2006, pp. 17–32.
[26] N. Yamada, K. Sakamoto, G. Kunito, Y. Isoda, K. Yamazaki, S. Tanaka, Applying ontology and probabilistic model to human activity recognition from surrounding things, Information Processing Society of Japan Digital Courier 3 (2007) 506–517.
[27] N. Roy, T. Gu, S.K. Das, Supporting pervasive computing applications with active context fusion and semantic context delivery, Pervasive and Mobile Computing 6 (1) (2010) 21–42.
[28] F. Latfi, B. Lefebvre, C. Descheneaux, Ontology-based management of the telehealth smart home, dedicated to elderly in loss of cognitive autonomy, in: Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, vol. 258 of CEUR Workshop Proceedings, 2007.
[29] M. Klein, A. Schmidt, R. Lauer, Ontology-centred design of an ambient middleware for assisted living: the case of soprano, in: Kirste, T., Knig-Ries, B., Salomon, R. (Eds.), AIM-CU: Proceedings of Towards Ambient Intelligence: Methods for Cooperating Ensembles in Ubiquitous Environments, Co Exists with the 30th Annual German Conference on Artificial Intelligence, KI 2007, 2007.
[30] R.E. McGrath, A. Ranganathan, R.H. Campbell, M.D. Mickunas, Use of ontologies in a pervasive computing environment, Tech. Rep. UIUCDCS-R-2003-2332 UILU-ENG-2003-1719, Department of Computer Science, University of Illinois, Urbana-Champaign, Urbana, Illinois, 2003.
[31] J. Ye, L. Coyle, S. Dobson, P. Nixon, A unified semantics space model, in: J. Hightower, B. Schiele, T. Strang (Eds.), Location-and Context-Awareness, in: LNCS, vol. 4718, Springer, 2007, pp. 103–120.
[32] G. Stevenson, J. Ye, S. Dobson, P. Nixon, Loc8: a location model and extensible framework for programming with location, IEEE Pervasive Computing 9 (2009) 28–37.
[33] S.S. Yau, D. Huang, H. Gong, Y. Yao, Support for situation awareness in trustworthy ubiquitous computing application software, Software: Practice and Experience 36 (9) (2006) 893–921.
[34] S.W. Loke, Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective, Knowledge Engineering Review 19 (3) (2004) 213–233.
[35] K. Henricksen, J. Indulska, T. McFadden, Modelling context information with ORM, Lecture Notes in Computer Science 3762 (2005) 626–635.
[36] J. Ye, S. McKeever, L. Coyle, S. Neely, S. Dobson, Resolving uncertainty in context integration and abstraction, in: ICPS 2008: Proceedings of the International Conference on Pervasive Services, ACM, New York, NY, USA, 2008, pp. 131–140.
[37] P.D. Costa, G. Guizzardi, J.P.A. Almeida, L.F. Pires, M. van Sinderen, Situations in conceptual modeling of context, in: EDOCW'06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference Workshops, Hong Kong, China, October 2006, pp. 6–16.
[38] J. Ye, S. Dobson, Human-behaviour study with situation lattices, in: SMC'09: Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, San Antonio, Texas, USA, 2009, pp. 343–348.
[39] P. Palmes, H.K. Pung, T. Gu, W. Xue, S. Chen, Object relevance weight pattern mining for activity recognition and segmentation, Pervasive and Mobile Computing 6 (2010) 43–57.
[40] Q. Yang, Activity recognition: linking low-level sensors to high-level intelligence, in: IJCAI'09: Proceedings of the 21st International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, pp. 20–25.
[41] N.F. Noy, R.W. Fergerson, M.A. Musen, The knowledge model of protege-2000: combining interoperability and flexibility, in: EKAW'00: Proceedings of the 2nd International Conference on Knowledge Engineering and Knowledge Management, Springer-Verlag, 2000, pp. 17–32.
[42] V.R. Jakkula, A.S. Crandall, D.J. Cook, Knowledge discovery in entity based smart environment resident data using temporal relation based data mining, in: ICDMW'07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, IEEE Computer Society, Washington, DC, USA, 2007, pp. 625–630.
[43] J.C. Augusto, C. Nugent, The use of temporal reasoning and management of complex events in smart homes, in: ECAI'04: Proceedings of the European Conference on Artificial Intelligence, 2004, pp. 778–782.
[44] H.H. Bui, D. Phung, S. Venkatesh, H. Phan, The hidden permutation model and location-based activity recognition, in: AAAI'08: Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI Press, 2008, pp. 1345–1350.
[45] J. Ye, L. Coyle, S. Dobson, P. Nixon, Using situation lattices in sensor analysis, in: PerCom'09: Proceedings of Eighth IEEE International Conference on Pervasive Computing and Communications, 2009, pp. 1–11.



Juan Ye currently is a research fellow in School of Computer Science at University of St. Andrews, UK. Her research areas are pervasive and ubiquitous computing and wireless sensor network. Her speciality is in ontology, context modelling and reasoning, and uncertainty resolving techniques. She has published in a number of international journals and conferences including Knowledge Engineering Review, PerCom, Pervasive, LoCA, and ICPS.

Graeme Stevenson is a Ph.D. candidate at the University of St. Andrews. His research interests include programming languages, middleware for smart spaces, and the Semantic Web. Stevenson has an M.Phil. in computer science from the University of Strathclyde.

Simon Dobson is currently a professor at School of Computer Science, University of St. Andrews, UK. He has been a co-founder of the Systems Research Group at UCD Dublin, Ireland's largest university. His research centres around adaptive pervasive computing and novel programming techniques, addressing both theory and practice and being supported by an extensive record of published work (including papers in CACM, TAAS, JPDC, EHCI and ECOOP) and primary authorship on grants worth over EUR 3M as well as collaborating in grants worth a further EUR 28M, feeding around EUR 1.5M directly into his own research programme. He serves on the steering or programme committees of many international conferences and workshops including PERVASIVE, AN, ICAC, ICOST, MUCS and MPAC. He is a reviewer for journals including ACM Transactions on Autonomous and Adaptive Systems, SOFTWARE—Practice and Experience, and IEEE Communications. He has also been an invited editor for special issues of Computer Networks, IJIPT and JNSM. Moreover, he is a member of the editorial boards of the Journal of Network and Systems Management and the International Journal of Autonomous and Adaptive Communications Systems, and participates in a number of EU strategic workshops and working groups. He is National Director for the European Research Consortium for Informatics and Mathematics, a board member of the Autonomic Communication Forum (at which he chairs the semantics working group), and a member of the IBEC/ICT Ireland standing committee on academic/industrial research and development. He holds a B.Sc. and D.Phil. in computer science, is a Chartered Engineer and Chartered IT Professional, and member of the BCS, IEEE and ACM.