Using Situation Lattices in Sensor Analysis*

Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon
System Research Group, School of Computer Science and Informatics,
UCD, Dublin, Ireland

E-mail: juan.ye@ucd.ie

Abstract

Highly sensorised systems present two parallel challenges: how to design a sensor suite that can efficiently and cost-effectively support the needs of given services; and to extract the semantically relevant interpretations, or "situations", from the flood of context data collected by the sensors. We describe mathematical structures called situation lattices that can be used to address these two problems simultaneously, allowing designers to both design and refine situation identification whilst offering insights into the design of sensor suites. We validate the accuracy and efficiency of our technique against a third-party data set and demonstrate how it can be used to evaluate sensor suite designs.

1. Introduction

A pervasive computing environment assumes a number of invisible sensing/computational entities that collect information about users and an environment. With the help of these entities, a pervasive computing system will be able to understand users' situations or demands so that it can deliver services in a contextual manner. To facilitate this understanding, we propose a well-founded mathematical structure, called the *situation lattices*, to study relationships between a large number of low-level sensed data and high-level situations or human activities.

Within a situation lattice, sensor data are abstracted and organised with respect to their semantics (such as different abstraction levels and conflicts between them). The situation lattice will apply basic semantics to learn richer semantics that describe relationships between sensor data, between sensor data and situations, and between situations.

*This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 "Towards a semantics of pervasive computing", 07/CE/I1147 "Clarity, the centre for sensor web technologies", and 04/RPI/1544 "Secure and predictable pervasive computing".

Situation lattices have been proposed in our earlier work [22, 23], which allowed developers to manually organise abstracted sensor data in a lattice and to label them with situations. The preliminary evaluation results were promising when we constructed a simple situation lattice to describe research activities in an office environment by using a small number of sensor data. This paper extends the initial model of situation lattices to deal with more complicated environments where a larger number of sensors are involved and relationships between sensor data and situations are less explicit.

This paper proposes new approaches to automatically generate a situation lattice with respect to *context predicates*, which are abstracted from sensor data; and to automatically label context predicates with situations. We will use situation lattices to help system developers to evaluate the performance of sensors, including evaluating whether a new sensor (or a new type of sensors) should be introduced, depending on whether existing sensors can sufficiently recognise situations.

To test the general applicability of a situation lattice, we demonstrate its feasibility by building it from the third-party real world data set – the PlaceLab data set [7]. The PlaceLab is an instrumented home that contains over nine hundred sensors. The data set was gathered in the real world conditions in that a married couple (who were unaffiliated with the PlaceLab research) lived in the PlaceLab over a period of 15 days. During this period, they were encouraged to maintain their life routine as normal as possible. The Placelab was also instrumented with the audio-visual recording infrastructure that was used to record activities of the subjects except for private activities (such as bathing). The video was annotated by a third party, which provided the ground truth of this data set. So far, only the activities of the male subject were annotated. Besides the sensor data and annotated diary, this data set involved a location map of this home, and a sensor metadata file[1]. This metadata

[1]The sensor metadata file is available online. http://web.media.mit.edu/~intille/data/PLCouple1/

file records the meta information about each sensor input, including its type, its identity (ID), where (e.g., the living room) it is installed, and which object (e.g., the couch in the living room) it is attached to. More details about the PlaceLab data set will be described in Section 4.1.

This paper is organised as follows. Section 2 compares our approach with the main stream of work in situation identification and activity recognition. Section 3 describes a conceptual model of information flow in a pervasive computing system such as sensor data, context, and situations and introduces the theoretical foundation of situation lattices. Section 4 describes the PlaceLab data set (including the sensors and the primary activities of the subjects). It demonstrates how a situation lattice is constructed on this data set: abstracting context predicates from sensor data, defining semantics between context predicates, generating a situation lattice, and training and pruning it. Section 5 applies situation lattices in inferring situations and analysing sensors. In the end, we will draw a conclusion of our work in Section 6.

2. Related Work

There are two typical approaches to define situations: specification-based approaches using expert knowledge [6, 8, 14, 20], and learning-based approaches using machine learning techniques [1, 4, 5, 7, 9, 10, 18, 23]. The complexity of a pervasive computing environment undermines the effectiveness of these two approaches. With the specification-based approach, since information in a pervasive computing covers sensor data, domain knowledge about an environment (that is, a property of an environment like a space map) and users (like social network or tasks), developers must consider all this information and then be able to provide good definitions of situations.

With learning-based approaches, developers widely apply machine learning techniques such as decision tree [1, 7], Bayesian inference [15, 23], and Hidden Markov Models [5, 18]. Bao *et al* [1] used decision tree to learn user body motions (such as bicycling, shaking hands, or typing) using the raw sensor data from accelerators on human body. Kasteren *et al* [15] carried out activity recognition in a Bayesian framework. They used a static Bayesian model to learn the relationship between different sensor data and human activities, and also used a dynamic Bayesian network to model the temporal aspects of activities. Wojek *et al* [18] proposed to use a multi-level hidden markov model to recognise activity from the audio and video data. These approaches consume a large number of training data to build a model or estimate parameters [14].

All these machine learning techniques are good at learning activities from "flat" information; that is, information consists of individual pairs of attribute and value, and there is no direct relationship between attributes. These techniques do not facilitate the expression of the semantics of information. If two pieces of information are conflicting or at different abstraction levels, these techniques will not be able to represent and use the relationships in their learning process. However, these rich semantic relationships of context are one of the characteristics of context in pervasive computing.

We assume that developers might have more accurate and sound knowledge in each local domain than their knowledge about the whole domain; that is, how to integrate and relate the knowledge in local domains to situations. Situation lattices use a semi-learning means in that they allow experts to express their knowledge about local domains and use this knowledge to exploit the knowledge about the whole domain. This approach will be able to take the advantages of both specification-based approaches and learning-based approaches, and compensate for their deficiencies.

3. Situation Lattice

This section will provide an understanding of information flow in a pervasive computing, including sensor data, context, and situations. Based on this basic understanding, we will describe the theoretical model of a situation lattice.

3.1. Sensor Data, Context, and Situations

In a pervasive computing environment, sensors gather information about users and their surroundings. A piece of sensor data consists of at least a time stamp, a sensor ID, and a sensor reading that represents how this sensor characterises a property of a user or an environment.

To facilitate the reuse and sharing of information in a system, sensor data need to be translated and represented in a structured representation – *context predicates*. Context predicates are defined as characteristic functions on sensor data. For example in the *electrical current flow* context, a context predicate `currentInLivingRoomOn` holds if the current flow value from a sensor is above 100.00. In the *time* context, a context predicate `17-18` holds if a time instant is between the hour 17:00 and 18:00. *Contexts* are instantiated context predicates that represent a concrete snapshot of reality.

In one type of context, context predicates can have semantic relationships: various abstraction levels and conflict. In the *time* context, a predicate `17-24` holds when a time instant is between the hour 17:00 and 24:00, which is at a coarser granularity than another predicate `17-18`. Also, two context predicates can be conflicting with each other, if any sensor datum that satisfies one of these predicates will

`PLObjects_oct23.xml`

never satisfy the other one. Ye *et al* [24] has applied set theory to formally discuss the semantic relationships between context predicates.

Situations are defined as an external semantic interpretation of context by Coutaz *et al* [3]. Each situation corresponds to invariant characteristics of contexts and their combination [16]. Each of the characteristics can be expressed as a logical description that takes context predicates as input and applies logical operators between them. A user is considered to *be in* a situation when its logical description is satisfied by the current context related to the user.

This paper aims to propose a model to express the above semantic relationships between context predicates. The model will be able to integrate the local knowledge and exploit richer semantics between machine-understandable sensor data and human-understandable situations.

3.2. Theoretical Model of Situation Lattices

Based on lattice theory [2], we propose a new data structure to study the relationship between logical descriptions and situations, called a *Situation Lattice*.

A situation lattice consists of a set of nodes that are organised with the *specialisation* relationship. Each node represents a logical description of context predicates, which is associated with a set of situations. The semantics of each node is that when the logical description of a node holds (or a node is activated), any situation in its situation set is possible to occur. Alternatively, any situation that is outside this situation set is impossible to occur.

There are two types of nodes in a situation lattice: *preliminary nodes*, whose logical description is a single context predicate; and *compound nodes*, whose logical description is a conjunction of context predicates. A compound node is created from preliminary or other compound nodes.

The specialisation relationship is a partial order defined between nodes. A node is considered more *specific* ($\sqsubseteq$) than another node, if and only if the logical description of the former node entails that of the latter node. The semantics of the specialisation relationship is that whenever a more specific node is activated, then all its more general nodes are activated. The formal definition of a situation lattice is given as follows.

Definition 1. *A situation lattice $L = (N, \sqsubseteq)$ is defined as follows.*

- *$\forall n \in N$, n has a logical description $\mathbf{n.l}$ and n corresponds to a set of situations $\mathbf{n.S}$ that are interesting to applications.*
- *$n_i \sqsubseteq n_j \in N$ iff $n_i.l \vdash n_j.l$, where $\vdash$ is the logical entailment relationship.*

We specify a few assumptions on a situation lattice, which are given as follows: Among all the nodes,

- *Assumption 1:* there exists a unique top node $n_\top$ whose logical description is a tautology `TRUE`. $n_\top$ corresponds to all situations. Its semantics is that if no sensor reading is available, then any situation is possible to occur;
- *Assumption 2:* there exists a unique bottom node $n_\bot$ whose logical description is a contradiction `FALSE`. $n_\bot$ corresponds to an empty set of situations. Its semantics is that if sensor readings are conflicting, then no situation can be derived;
- *Assumption 3:* no two nodes share the same logical description.

A situation lattice is a join semi-lattice. For any two nodes, there exists a join node that contains the greatest logical description that is entailed by any logical description on these nodes. The situation set on their join is the least superset of the union of their situation sets. There does not necessarily exist a meet for any two nodes. If there exists a meet for a set of nodes, the logical description on the meet contains the least logical description that entails any logical description on these nodes. The situation set on their meet is the greatest subset of the intersection of their situation sets.

Within a situation lattice, we can express the semantic relationships between context predicates: various abstraction levels and conflict. According to Definition 1, the various abstraction levels between two context predicates are represented as a specialisation relationship between two preliminary nodes that host these two context predicates. The meet is used to express the conflict relationship. If two nodes host conflicting context predicates, then their conjunction is `FALSE`. According to *Assumption 2* and *3*, there exists a unique node $n_\bot$ whose logical predicate is `FALSE`, so the meet of conflicting context predicate is $n_\bot$. This conflicting relationship is formalised in the following corollary.

Corollary 2. *$n_i \sqcap n_j = n_\bot$, iff $n_i.l \wedge n_j.l = \mathit{FALSE}$.*

Once the conflicting relationship is defined between two preliminary nodes, it will be "inherited" by their more specific nodes. That is, if two nodes n_i and n_j are conflicting, then any of their more specific nodes n_i' and n_j' should conflict with each other (see Lemma 3). This will be used to avoid inconsistency (no compound nodes are created from conflicting nodes) during the process of automatically generating a situation lattice.

Lemma 3. *If $n_i \sqcap n_j = n_\bot$, then $\forall n_i' \sqsubseteq n_i$, $n_j' \sqsubseteq n_j$, $n_i' \sqcap n_j' = n_\bot$.*

4. Construction of Situation Lattices

This section will describe how a situation lattice is built from scratch. We will use the PlaceLab data set as an example to demonstrate the following processes: (1) *defining preliminary nodes* by abstracting sensor data into context predicates; (2) *defining semantic relationships* between preliminary nodes using domain knowledge; (3) *generating a situation lattice* based on the preliminary nodes; and (4) *training and refining* the lattice.

4.1. The PlaceLab Data Set

The PlaceLab is a living laboratory in Cambridge, Massachusetts, which is designed to be a highly flexible and multi-disciplinary observational research facility. It is used for the scientific study of people and their interaction patterns with new technologies and home environments. The PlaceLab contains over nine hundred sensors inputs, which covers the following types of sensors:

Wireless infra-red motion sensors detect motion in the regions of the laboratory, including the living room, the foyer, the dining room, the kitchen, the office, the bedroom, the bathroom, and the powder room. These sensors can be used to infer the subject's location, but they are not always accurate. The sensors are not person-specific, so the sensed motion can be caused accidentally or by the other subject. Different sensors often report the motions at the same time.

"Stick-on" object motion sensors are installed on doors, cabinets, couches, remote controls and measure their use by sensing motion in them. These sensors are not person-specific in that it is impossible to know which subject caused an object's motion. Since these sensors are installed in large numbers of objects, it is difficult to maintain a good sensor metadata file.

Switch sensors measure the state of objects like doors, cabinets, and wardrobes: whether they are open or closed. *RFID sensors* measure whether an object is being accessed or not. The RFID reader does not work well due to technical limitations, being inaccurate if the distance between the reader and the tag is outside the range of the reader [7]. In this data set, only the male subject wore the RFID reader on his right wrist.

Electrical current, water, and gas flow sensors measure the real-time usage of current, water, and gas in the home. The current flow sensors are installed on residential circuits including the living room, the office, and the kitchen. The water flow sensors are installed on the faucets in the kitchen, the bathroom, and the powder room. The gas flow sensor is installed on the stove.

Environmental sensors measure the temperature, humidity, and barometric pressure in different regions in the living space. *3-axis accelerometer sensors* measure a subject's body movement, which are only worn on the male subject's limbs and wrists.

During the data collection time, the subjects tried to live as normally as possible. The primary activities that were recorded in the diary are listed in Table 1.

Activity	**Description**
using phone	using a portable phone or the phone on the fax machine in the office
using a computer	using a desktop in the office or a laptop anywhere
reading	reading books, magazines, or printed paper
eating a meal	eating regular meals
eating a snack	eating a light meal
meal preparation	retrieving ingredients or cookware, combining or adding, stirring or mixing, warming food using microwave, or preparing a drink
watching TV	sitting in front of the television and actively watching TV or movies
dishwashing	washing hand or rinsing dishes using the faucet in the kitchen
hygiene	mainly about activities in the bathroom or powder room, like toileting, showering, or washing face
grooming	getting undressed or dressed, including clothes and sensors

Table 1: The primary activities in the PlaceLab data set

To demonstrate our work, we chose a subset of sensors that are straightforward to interpret and are potentially related to the above activities, which are the wireless infrared motion sensors, electrical current, water, and gas flow sensors, switch sensors, RFID, and object motion sensors. In our experiment, we chose a subset of the data set that included only the days that were covered by the diary data, which are 2006-08-23, 2006-08-25, 2006-09-05, 2006-09-06, 2006-09-11, 2006-09-12, 2006-09-14, and 2006-09-18. In total, we used almost 40 hours of the PlaceLab data out of 104 hours.

4.2. Defining Preliminary Nodes

To create a situation lattice, developers should define preliminary nodes for each single context predicate. A context predicate is a characteristic function on sensor data, which is associated with a set of sensor IDs and a constraint on readings from these sensors. A context predicate holds if its constraint is satisfied by a reading from any of these sensors. For the PlaceLab data set, we use the sensor metadata file. We classify the sensor IDs according to the types

of sensors, and defined context predicates for each sensor ID. The context predicates can be defined according to developers' knowledge on sensors, or sensors' technical specifications that indicate what a sensor reading means.

The PlaceLab researchers have published the technical specification of their sensors online[2]. For a switch sensor, the specification recorded that when its reading is either 0 or 200, then the state of an object that it attaches to is open; when a reading is 100 or 300, then the state of the object is close. One of the switch context predicates is defined as `doorInBedroomOpen`, which holds when the switch sensor on the bedroom door produces readings 0 or 200.

We could define context predicates by observing the sensor readings for each sensor ID in the sensor output files, since the technical specification does not record characteristic readings for all the sensors. For example of the infra-red motion sensors, their sensor output files records a list of time stamps, sensor IDs, and their corresponding values (which are either 5 or 10). Based on the output file, we define the location context predicates; for example, `inLivingRoom` is defined on a sensor ID 2088 when its reading is no less than 5. Using the same approach, we define context predicates for the electrical current, water, and gas flow sensors. We also define the time context predicates since the activities are potentially time-related such as "grooming". The time predicates are defined in an hour scale, e.g., `17-18` or `23-24`.

RFID and object motion sensors have a similar function of sensing whether an object is accessed or moved. If one object is attached by these two sensors, we define one object access context predicate on this object. For example, we define an object access predicate called `laptopAccessed` on the laptop that has two sensors attached: the RFID sensor `E00700001E226FEE` and the object motion sensor `956`. This predicate holds, if the RFID sensor is recorded, or the reading of the object motion sensor is greater than 0.

If sensors produce sophisticated readings (such as the body 3-axis motion accelerometers), machine learing techniques could be used to train context predicates. It would be simple if there existed a diary about fine-grained activities that recorded the movement of the subjects, such as walking, sitting, moving the arm, or lifting an object. Since the PlaceLab data set only provides a diary that records higher level activities as shown in Table 1, we cannot define movement context predicates from these accelerometers, so we do not use these sensors in our experiments.

[2]The technical specification of the PlaceLab sensors are available online. `http://architecture.mit.edu/house_n/data/PlaceLab/PLIA1.htm`

4.3. Defining Relationships between Preliminary Nodes

Developers can apply the domain knowledge to define the semantic relationships between these preliminary nodes as discussed in section 3.1. The domain knowledge can be the knowledge about an environment such as a location map, which defines the spatial containment, connectedness, and disjointness relationships between locations [21]. In the PlaceLab environment, the location context predicates conflict with each other, since each of them represents an individual room and the subject cannot be in two different rooms at the same time. Developers can also express their common-sense knowledge, for example, the predicates `lightInLivingRoomOn` and `lightInLivingRoomOff` cannot hold at the same time. The conflict between context predicates is represented by defining the bottom node as the meet of their host nodes (in Corollary 2). Another semantic relationship is varying abstraction level, which is expressed as the specialisation relationship between their host nodes. For example, a preliminary node hosting the time predicate `17-23` is defined more specific than another preliminary node hosting `17-24`.

4.4. Generating a Situation Lattice

The lattice generation algorithm will automatically generate a situation lattice by starting from the preliminary nodes and combining nodes level by level until all the non-conflicting nodes are combined. This top-down generation process guarantees the existence of the join for any pair of nodes in a situation lattice. The process of creating a new node is described as follows. Given two nodes n_i and n_j, their compound node (labelled as $n_i \otimes n_j$) can be

- the bottom node, if they are conflicting, which is evaluated by using Corollary 3;
- or a new node $n_{ij} = n_i \otimes n_j$ with
  - $n_{ij}.l = n_1.l \wedge n_2.l$;
  - $n_{ij}.S = n_1.S \cap n_2.S$;
  - $n_{ij} \sqsubseteq n_i$; and
  - $n_{ij} \sqsubseteq n_j$.

The generation process suffers a large problem of scalability. Given m preliminary nodes, the complexity of generating a lattice (with the conjunction) will be $O(2^m)$. With the scalability problem, it is unrealistic to list all the combinations of the context predicates. This is the reason that a situation lattice is not meet complete, since we do not combine all the non-conflicting preliminary nodes. A practical approach is to combine the nodes if

their predicates are related to each other. In our experiment, we combine the context predicates if they are related to the same location. For example, a node hosting `currentInLivingRoomOn` is combined with a node hosting `lightInLivingRoomOn`, while it is not combined with a node hosting `currentInOfficeOn`. The underlying assumption is that the context (or the sensor data) closely related to the surrounding of the subject will have a major effect on inferring his current situations.

We have defined 7 location predicates and 7 time predicates. We split the switch predicates, the current, water and gas flow predicates into these seven locations: 4 current predicates in the living room, 2 switch predicates and 4 current predicates in the office, 2 current predicates in the dining room, 6 switch predicates and 2 current predicates in the bedroom, 12 switch predicates, 4 current, 2 flow, and 2 gas predicates in the kitchen, 4 switch predicates, 2 current and 2 flow predicates in the bathroom and powder room respectively. For RFID and object motion predicates, we classify them into eight groups: the predicates in the first group that are defined on moveable objects and thus can be combined with a predicate in any location, and the predicates in the following seven groups that were defined on objects belonging to a particular location and can only be combined with predicates in the same location.

For each location, the generation algorithm generates the compound nodes by combining non-conflicting time, location, switch, RFID, and object motion predicates, and current/water/gas flow predicates. For example in the living room, given 4 current predicates with 2 pairs of conflicting predicates, 8 compound nodes are generated with these current predicates. 624 (=8×7+ 8×8+ 7×8+8×7×8) compound nodes are generated with the above 8 current compound nodes, 7 time predicates, and 8 object access predicates in the living room. With the same process, we generate a situation lattice with 27649 nodes in total: 1 top node, 1 bottom node, 113 preliminary nodes and 27534 compound nodes. The lattice of this size is easier for a system to manage, compared to a fully generated lattice with about 2^{113} nodes.

4.5. Training and Refining a Situation Lattice

So far, we have combined all the context predicates in a lattice. Now we will start to learn what these combined context predicates mean; that is, relationship between them and human-understanding situations. This is done by training a situation lattice with synchronised sensor and diary data. The training process will attach each node to a set of situations; that is, at each time instant when all context predicates on a node hold, the situations recorded occurring in the diary will be added to this node's situation set, which is presented in Figure 1. At the end of training, each node is associated with the times it is activated, and an array of situation occurrences (how many times a situation occurs when it is activated). For each node, the array of situation occurrences can be normalised, by being divided by the total occurrence of each situation.

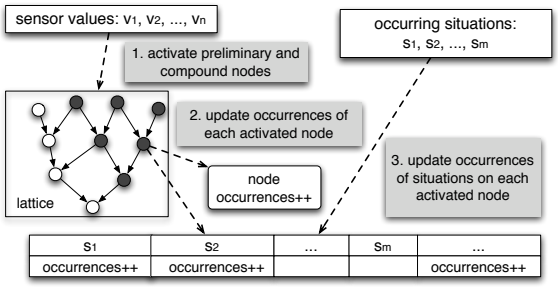

Figure 1: A process of training a situation lattice

After training, the lattice will be pruned by removing ineffective nodes. Nodes are ineffective if they have never been activated, or no situations occur when they are activated. By examining the removed nodes, we summarised the features of ineffective nodes. Nodes not being activated can be caused by ineffective sensors or bad combinations of predicates. For example, `snackAccessed` never holds, since RFID sensors on all the snack related objects (such as chips or peanuts) do not work well. An example of bad combinations of predicates is the node with `currentInLivingRoomOn` ∧ `lightInLivingRoomOff` that never holds in the training data, which uncovers the subject's activity pattern: whenever he used the electronics in the living room, he would switch on the light first. Another type of ineffective nodes is that they do not contribute to identifying any situation. For example, no situations are occurring when the node with `inOffice` ∧ `23-24` ∧ `currentInOfficeOff` ∧ `lightInOfficeOff` is activated.

We synchronise the sensor and diary data, and slice them into 10-second gaps, which produce 13870 pairs of sensor and diary instances. We use two third of them to automatically train the lattice and prune it using the above two rules. In the end, the number of the nodes is reduced to 13035, which is half of the size of the initial lattice.

5. Applications of a Situation Lattice

When a situation lattice is trained, it is ready to be used in inferring situations and evaluating the discernibility of sensors in detecting situations.

5.1. Inferring Situations

With respect to inferring situations, the structure of a situation lattice makes forward chaining algorithms more efficient by evaluating all the predicates just once. When fed with a set of sensor readings, the system starts by evaluating the predicates on all the preliminary nodes from the top. Situation inference is a procedure of applying the meet operator $\sqcap$ on all the activated preliminary nodes. Since the lattice is not meet-complete, the procedure may end up with a set of nodes that are most specific among all the activated nodes, instead of a unique node.

Instead of inferring a single situation, a situation lattice will return the following two sets of situations where each situation is provided with its occurrence ratio:

- a set of *possible-to-occur* situations that is the union of the situations in these resultant nodes, which indicates that the situations outside this set are impossible to occur; and
- a set of *most-likely-to-occur* situations that is the intersection of the situations in these resultant nodes, which indicates that these situations are more likely to occur compared to the other situations in the above set.

For example, if a situation lattice ends up with two most specific nodes whose situation sets are {"using computer", "reading"} and {"watching TV", "reading"} respectively, then the situations possible to occur are in {"watching TV", "using computer", "reading"}, and the situations most likely to occur are in {"reading"}.

The inference result of a situation lattice provides more accurate and detailed information about what is occurring in the real world. This will help application developers to design a more robust and customised system. Applications (or services) related to the most-likely-to-occur situations will be provided with a high confidence, while applications whose corresponding situations are not contained in the possible-to-occur situations should never be triggered. Applications related to the situations that are possible but not most likely to occur are not suggested to be triggered, but they can be triggered with the consideration of their associated occurrence probability and other design requirements.

5.1.1 Evaluation Methodology

We use the stratified 10-fold cross validation technique, and compare our result with the result using Naive Bayes and decision tree in WEKA software package [17]. These techniques were used in the PlaceLab data set paper [7], and they achieved good results.

The accuracy of inferring situations is evaluated using two parameters: *precision* and *recall*. Precision is the ratio of the times that a situation is correctly inferred to the times that it is inferred in most-likely-to-occur situation sets. Recall is the ratio of the times that a situation is correctly inferred in possible-to-occur sets to the times that it occurs.

When there exist conflicting nodes among the resultant nodes, the most-likely-to-occur situation set should be an empty set (see Lemma 3); that is, if the nodes conflict, then their situation sets should be mutually exclusive. In addition, the possible-to-occur situations will contain all the situations, when the conflicting sensor data activate most of preliminary nodes. Under this circumstance, all the situations are equally possible to occur, while none of them is most likely to occur. This makes both the possible-to-occur and the most-likely-to-occur situation sets insensitive to sensors. This insensitivity will produce 100% precision and 100% recall, which is a "fake victory" in our evaluation.

To avoid the insensitivity, we set thresholds on occurrence ratio of each situation to refine situations in the most-likely-to-occur and the possible-to-occur sets. When the conflict is detected in the resultant nodes, the most-likely-to-occur situation set will contain the situations with the highest probability, which excludes the possibility of 100% precision. The possible-to-occur situation set will filter out the situations whose occurrence ratio is below its corresponding threshold, which decreases the possibility of 100% recall but does not exclude it. Therefore, we will calculate the chance that a possible-to-occur situation set covers all the situations.

Since we will set thresholds on each situation to select most-likely-to-occur and possible-to-occur situations, now the question is how to set them. A threshold should balance the precision and recall. If a threshold is too low, it guarantees the best recall but reduces the precision greatly. If a threshold is too high, it guarantees better precision but decreases the recall. To balance the precision and recall, we apply the following F-measurement [25] that treats the precision and recall equally.

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

For each situation, we set the initial threshold to 1.0 and repeat reducing the threshold down to 0 with the scale 0.005. For each situation, we calculate the F-measurements for all these thresholds and choose the inference result that achieves its highest F-measurement, which best balances its precision and its recall.

5.1.2 Inference Experiment on PlaceLab Data Set

We use the synchronised 13870 sensor and diary instances in Section 4.5 to conduct stratified 10-fold cross validation on the situation lattice, Naive Bayes, and Decision Tree. There are 250 (1.8%) times that the possible-to-occur situation sets cover all the situations during these 10 folds'

validations. It shows that there is a small chance that the possible-to-occur situation sets are insensitive. Figure 2 and 3 present the comparison of recall and precision in inferring situations between situation lattice, decision tree, and Naive Bayes.

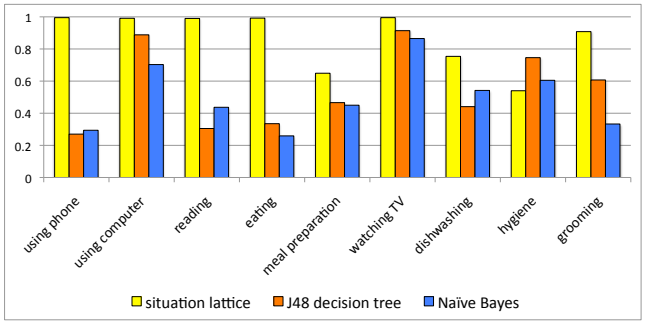

Figure 2: Recall for situation lattice, Naive Bayes, and Decision Tree

Figure 2 shows that the situation lattice has the highest recall on most of the situations (except for "hygiene"), since the possible-to-occur situations contain all the possible situations that are above the threshold. Figure 3 shows that the situation lattice has the relatively lower precision compared to the other two techniques.

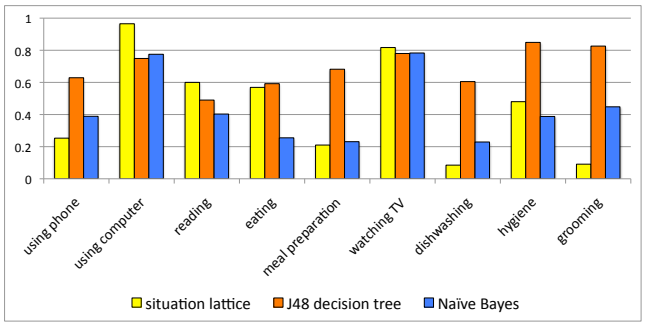

Figure 3: Precision for situation lattice, Naive Bayes, and Decision Tree

By examining the low-precision situations, we summarise the causes that a situation lattice has the low performance.

- The precision of inferring a situation will be low, if the sensors that are supposed to identify this situation are not accurate. For example, the situation "dishwashing" should be identifiable if the positioning sensor can accurately locate the male subject's location (i.e., in the kitchen), and the faucet in the kitchen can recognise that the male subject is accessing it.
- The precision of inferring a situation will be low, if the definition of this situation is imprecise; for example, "meal preparation" covers all the cooking activities (even including preparing a drink); "grooming" can be getting dressed or undressed (including clothes and worn sensors); "eating" can be eating a proper meal, or snack, or drinking. The imprecise definition of a situation makes it hard to discover its pattern.
- The precision of inferring a situation will be low, if this situation can randomly co-occur with any other situation and there are no effective sensors to identify it. For example, "using phone" can occur with any other situation at any time in any room. Since this situation cannot be identified or distinguished from other situations with characterised sensor data, this situation is frequently co-inferred with other situations.

5.2. Discernibility of Sensors on Situations

With even more sensors available and ready to be widely instrumented, developers in pervasive computing will need to find out what sensors satisfy application requirements best. If there exists a pair of situations that are required to be precisely identified but cannot be distinguished from each other with the existing sensors, developers might be interested in what new sensor or new type of sensors should be introduced. When the sensors are ready to be widely installed in other environments (for example, sensors for a smart home are ready to use in normal people's homes), developers might also need to find out what is the best configuration of sensors to achieve the best accuracy of inferring situations.

A situation lattice can facilitate detecting indistinguishable situations; that is, the situations are always identified at the same time. Indistinguishable situations can be caused by human activity patterns, for example, in the PlaceLab data set people often watch TV while they eat. They can also be caused by the lack of effective sensors, for example, only with positioning sensors it is difficult to determine whether people are reading or watching TV. Situation lattices cannot only detect when indistinguishable situations exist, but also they can point out what condition (or what sensor data) makes them indistinguishable. According to the requirements of applications, developers can decide whether the detected situations are necessary to be distinguished or not. If they are necessary, developers can use situation lattices to evaluate which type of sensors will have the best discernibility.

In a situation lattice, each node immediately above the bottom node should capture the smallest perceivable snapshot of reality, since it contains a conjunction of finest grained context predicates. Situations on each of these nodes should reflect the exact situations that are occurring.

If two situations share any of the same most specific nodes, then these two situations will be co-inferred whenever this node is activated. Algorithm 1 describes the process of detecting pairs of indistinguishable situations.

Algorithm 1: The algorithm of detecting indistinguishable situations

input : a set of nodes N immediately above the bottom node and a set of situation names S_s
output: pairs of indistinguishable situations

initialise a node array $nodes$ with the same size as S_s
foreach *Node* $n \in N$ **do**
  foreach *Situation* $s \in n.S$ **do**
    add n in $nodes[s.id]$
initialise a hashtable $pairs$ whose key is a pair of situation IDs and whose value is a set of nodes such that when these nodes are activated, this pair of situations are inferred together
for $i \leftarrow 0$ **to** $S_s.size$ **do**
  for $j \leftarrow i+1$ **to** $S_s.size$ **do**
    if $nodes[i] \cap nodes[j] \neq \emptyset$ **then**
      if (i,j) *not in* $pairs$ **then**
        initialise a set of nodes N_{ij}
      add all $nodes[i] \cap nodes[j]$ in N_{ij}
      put $((i,j), N_{ij})$ in $pairs$
return $pairs$

5.2.1 Sensor Evaluation Experiments on the PlaceLab Data Set

We will conduct an experiment to show how situations can be gradually distinguished from each other as new sensors are introduced. We start with an environment where no sensors are installed. We can build a simple situation lattice simply with the time context, since time is completely cost free [11].

Since the occurrences of most situations are overlapping, so they are indistinguishable from each other in this lattice only with the time predicates. We will examine each pair of indistinguishable situations to locate which situations that should not co-occur are indistinguishable from each other. We list a few groups of situations that should not co-occur as follows: (1) "hygiene" and "watching TV"; (2) "hygiene" and "meal preparation"; (3) "hygiene" and "dishwashing"; (4) "grooming", and "meal preparation"; and (5) "grooming", and "dishwashing".

We analyse the nature of these pairs of situations: each pair of situations occur in different rooms. We have a set of potential types of sensors to choose: infra-red motion sensors to infer the subject's location; electrical current, water, and gas flow sensors to detect the real-time usage of the flow; and the object motion sensors, switch sensors, and RFID to detect which objects are accessed or used by the subject. After examining all these sensors, we consider the infra-red motion sensors first, since their readings are directly related to the subject's location. Also, they are easy to install [19]: one sensor for each room. In contrast, the other sensors need a larger number of sensors and consume more installation effort. Therefore, the infra-red motion sensors come to our first choice.

A new situation lattice will be created with the location and time context predicates, and re-trained. Using Algorithm 1, we derive indistinguishable situations from the new lattice. Some of situations have been completely distinguished; for example, "dishwashing", "hygiene", "grooming" have been distinguished from the situations "using phone", "eating", and "meal preparation". "hygiene" has also been distinguished from "watching TV" and "dishwashing"; and "dishwashing" from "grooming". These distinguished pairs of situations have covered the above listed pairs of situations that should not co-occur.

Some of the situations are indiscernible at the more specific condition, even though they are sill indiscernible. For example,

(1) when `19-21` $\wedge$ `inKitchen` holds, "meal preparation", "using phone", "using computer", "reading", "eating", and "watching TV" are indiscernible from each other;

(2) when `23-24` $\wedge$ `inKitchen` holds, "dishwashing", "using computer", "reading", and "watching TV" are indiscernible;

(3) when `23-24` $\wedge$ `inBathroom` holds, "hygiene", "using computer", and "reading" are indiscernible; and

(4) when `23-24` $\wedge$ (`inBedroom` $\vee$ `inBathroom`) holds, "grooming", "using computer", "reading", "watching TV", and "hygiene" are indiscernible.

From the above listed conditions, it is easy to observe the involved noise of sensor data. For example, "watching TV" should not be attached to the location predicates `inKitchen`, `inBedroom`, and `inBathroom`. This can be caused by the fact that when the male subject was watching TV, the female subject appeared in any of these locations, which fired the infra-red motion sensors in these locations. This information discovered in the situation lattice is consistent with the discussion on the noise of sensor data in the PlaceLab publication [7].

The following task is to separate "meal preparation", "dishwashing", and "hygiene" from the co-occurring situations "using computer", "watching TV", "reading", and

"eating". There are two types of sensors left: the electrical current, water, and gas flow sensors, and the object motion, RFID, and switch sensors. For the condition in (1), we might consider the water flow sensors in the kitchen, and the object motion sensor on the faucet that might complement the noise on the location sensors to distinguish "dishwashing" from the other situations. For the condition in (2), we might consider the current flow sensor on the microwave, gas flow sensor on the stove, and any object motion sensor in the kitchen to distinguish "meal preparation" from the other sensors. For the condition in (3), we might consider the water flow sensors in the bath room, and the object motion sensor on the faucets to distinguish "hygiene" from the other situations. The situations in the condition (4) have been covered in the above discussions. We will use these sensors, and create a new situation lattice using the corresponding context predicates.

A new situation lattice is created with the time predicates, location predicates, and part of the current, water, gas flow predicates, and part of the object motion sensors. Using the same process, we find that the added sensors help to identify "meal preparation", "dishwashing", and "hygiene" from the other situations. However, "hygiene" is still indistinguishable from "using computer" and "eating" when `18-19 ∧ inPowderRoom ∧ waterInPowderRoomOff ∧ currentInPowderRoomOff ∧ NothingInPowderRoomAccessed` holds. We believe that this is caused by the noise on the infra-red motion sensors when the newly introduced sensors are not fired, since it would be impossible that the male subject would eat or use computer in the powder room.

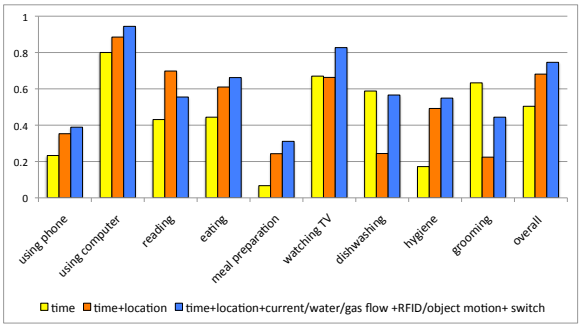

Figure 4: F-measurements on each situation with different situation lattices.

Figure 4 shows F-measurements on each situation with the three situation lattices built above. It is obvious that the F-measurements of most situations are gradually increased as more sensors are involved.

So far, we have demonstrated how to use situation lattices to find out what condition makes situations indistinguishable. Based on the given information, developers can use their knowledge of sensors to decide what new sensors might improve in distinguishing situations under a certain condition.

In another scenario where an environment has installed with different types of sensors, if these sensors are ready to be applied in other environments, developers can create multiple situation lattices with different combinations of sensors. These situation lattices will present the discernibility of the combinations of sensors, such as which combination of sensors makes which situations indiscernible and the corresponding accuracy of inference. With this requirement on a new environment, developers will be able to choose the best configuration of sensors.

6. Conclusion and Future Work

As the number of available sensors increases, it becomes difficult for developers to decipher and specify correlations between sensor data and situations. This paper has proposed a theoretical structure – the situation lattice – to make it easier to determine these correlations. This paper has demonstrated the feasibility of a situation lattice by constructing one from a real world data set, and has evaluated its performance in inferring situations and analysing sensors.

Situation lattices fit the needs of pervasive computing in that it is toward to manage rich semantics and complexity in context. Situation lattices provide a framework to allow developers to express the semantics of context and their domain knowledge and use them to uncover the correlation between context and situations. Situations will be automatically learned with the consideration of all the knowledge represented in situation lattices. In this way, situation lattices integrate the advantages of both the specification-related and learning-related approaches.

They can help to build a more robust pervasive computing system by providing richer inference result. The structure of a lattice will also make it possible to evaluate the performance of sensors in distinguishing situations.

Scalability is the major limitation of situation lattices. Ideally, developers define context predicates, and then a situation lattice will be generated by automatically combining all the non-conflicting context predicates. The complexity of the generation process is $O(2^n)$, which is impractical in a real-time pervasive computing system. One choice is to combine context predicates if they are related. Thus, a situation lattice would rule out potentially useful combinations of context. Besides, the scalability problem limits the expressivity of logical descriptions. Currently, a situation lattice only supports generating conjunction of context predicates. If disjunction and negation of context predicates are involved, then the complexity will grow much faster than exponential [13].

In the future, we will evaluate the performance of situation lattices in more complicated environments like an office setting, where richer semantics of context need to be captured, and more interactive activities and more subjects are involved. We have applied lattice theory in modelling a space map in a complicated environment (such as a building or an city plan) in [21]. This work proved the lattice structure has a powerful expressive ability in representing the semantics of location information. Another type of context that has rich semantics is social network, so we will use the lattice theory to model social network information as well. In the end, we will study how to organise each of the domain lattices in a situation lattice and deal with the scalability issue.

References

[1] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In *Second International Conference on Pervasive Computing*, pages 1–17, Vienna, Austria, Apr. 2004.

[2] G. Birkhoff. *Lattice Theory*. Providence, R.I. : American Mathematical Society, 3rd edition, 1967.

[3] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is Key. *Commun. ACM*, 48(3):49–53, 2005.

[4] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2004)*, pages 270–275, January 2004.

[5] M. K. Hasan, H. A. Rubaiyeat, Y.-K. Lee, and S. Lee. A hmm for activity recognition. In *10th International Conference on Advanced Communication Technology (ICACT 2008)*, volume 1, pages 843–846, Feb. 2008.

[6] K. Henricksen and J. Indulska. Developing contextaware pervasive computing applications: Models and approach. In *Pervasive and Mobile Computing, In Press, Elsevier*, 2005.

[7] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of Ubicomp 2007*, pages 483–500, Innsbruck, Austria, September 2007. Springer.

[8] S. W. Loke. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *Knowledge Engineering Review*, 19(3):213–233, 2004.

[9] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[10] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision Image Understanding*, 96(2):163–180, 2004.

[11] K. Partridge and P. Golle. On using existing time-use study data for ubiquitous computing applications. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 144–153, New York, NY, USA, 2008. ACM.

[12] C. S. Pinhanez and A. F. Bobick. Interval scripts: a programming paradigm for interactive environments and agents. *Personal Ubiquitous Comput.*, 7(1):1–21, 2003.

[13] U. Priss. Lattice-based information retrieval. *Knowledge Organisation*, 27(3):132–142, 2000.

[14] G. Thomson, S. Terzis, and P. Nixon. Situation determination with reusable situation specifications. In *Proceedings of PERCOMW '06*, pages 620–623. IEEE Computer Society, 2006.

[15] T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. In *Intelligent Environments, 2007. IE 07. 3rd IET International Conference on*, pages 209–212, September 2007.

[16] N. Weisenberg, R. Gartmann, and A. Voisard. An ontology-based approach to personalized situation-aware mobile service supply. *Geoinformatica*, 10(1):55–90, 2006.

[17] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[18] C. Wojek, K. Nickel, and R. Stiefelhagen. Activity recognition and room-level tracking in an office environment. In *2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 25–30, Sept. 2006.

[19] C. R. Wren and E. M. Tapia. Toward scalable activity recognition for sensor networks. In *LoCA'2006, Lecture Notes in Computer Science 3987*, pages 168–185. Springer, May 2006.

[20] S. S. Yau and J. Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *the Proceedings of SEUS-WCCIA'06*, volume 0, pages 5–10, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[21] J. Ye, L. Coyle, S. Dobson, and P. Nixon. A unified semantics space model. In J. Hightower, B. Schiele, and T. Strang, editors, *Location- and Context-Awareness*, volume 4718 of *LNCS*, pages 103–120. Springer, 2007.

[22] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Using situation lattices to model and reason about context. In *Proceedings of MRC 2007 (coexist with CONTEXT'07)*, pages 1–12, Roskilde, Denmark, August 2007.

[23] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Representing and manipulating situation hierarchies using situation lattices. *Revue d'Intelligence Artificielle*, 22(5):647–667, 2008.

[24] J. Ye, S. McKeever, L. Coyle, S. Neely, and S. Dobson. Resolving uncertainty in context integration and abstraction. In *Proceedings of the international conference on Pervasive Services*, pages 131–140, New York, NY, USA, July 2008. ACM.

[25] Y. Zhou, M. Würsch, E. Giger, H. C. Gall, and J. Lü. A bayesian network based approach for change coupling prediction. In *WCRE '08: Proceedings of the 2008 15th Working Conference on Reverse Engineering*, pages 27–36, Washington, DC, USA, 2008. IEEE Computer Society.