OPSitu: A Semantic-Web Based Situation Inference Tool Under Opportunistic Sensing Paradigm

Jiangtao Wang[1,2], Yasha Wang[1,3(✉)], and Yuanduo He[1,2]

[1] Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871, China
wangys@sei.pku.edu.cn
[2] School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[3] National Engineering Research Center of Software Engineering, Peking University, Beijing, China

Abstract. Opportunistic sensing becomes a competitive sensing paradigm nowadays. Instead of pre-deploying application-specific sensors, it makes use of sensors that just happen to be available to accomplish its sensing goal. In the opportunistic sensing paradigm, the sensors that can be utilized by a given application in a given time are unpredictable. This brings the Semantic-Web based situation inference approach, which is widely adopted in situation-aware applications, a major challenge, i.e., how to handle uncertainty of the availability and confidence of the sensing data. Although extending standard semantic-web languages may enable the situation inference to be compatible with the uncertainty, it also brings extra complexity to the languages and makes them hard to be learned. Unlike the existing works, this paper developed a situation inference tool, named *OPSitu*, which enables the situation inference rules to be written in the well accepted standard languages such as OWL and SWRL even under opportunistic sensing paradigm. An experiment is also described to demonstrate the validity of *OPSitu*.

Keywords: Semantic web · Situation inference · Opportunistic sensing

1 Introdution

In the research of situation-aware systems, situation inference is considered to be an important technique, which focuses on how to infer the situation of an entity (i.e. a person, a thing or a place) based on sensing data collected from the physical space or the cyberspace [1]. Among multiple approaches for situation inference, the Semantic-Web based approach is widely adopted [2–6]. In this approach, standard Semantic Web languages, such as OWL (Web Ontology Language) and SWRL (Semantic Web Rule Language), are used to model the related concepts and inference rules at design time. After obtaining the sensing data, situation inference process is conducted by a semantic inference engine

I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 3–16, 2014.
DOI: 10.1007/978-3-319-11569-6_1

at runtime. In these works, there is a common assumption that the sensing data are certain and complete during the inference process [1].

In recent years, with the technological advance and popularity of IOT (Internet of Things) and mobile computing, sensing infrastructures have been established in our daily surroundings. The massively existing sensing devices include static sensors spreading across buildings, streets, public parks and rivers, and mobilizable sensors carried by people and vehicles, such as built-in sensors in smartphones, tablets, wearable devices, vehicles borne radars, GPS, cameras, etc. Together with the sensors, wireless communication infrastructures, such as WSN, Wi-Fi and 3G/4G mobile network, are also available almost everywhere to deliver sensing data. With these abundant sensors and sensing data delivery infrastructures, a new sensing paradigm emerges, which is referred to as *Opportunistic Sensing* [7–11]. Instead of pre-deploying application-specific sensors, opportunistic sensing applications make use of sensors that just happen to be available to accomplish its sensing goal [11].

Due to the sensor sharing mechanism, the opportunistic sensing paradigm is less costly and more environmental friendly. However, it leads to new technical challenges to those applications that adopt the Semantic-Web based situation inference approach. Firstly, opportunistic sensing attempts to discover and utilize sensors available by chance. Therefore, when there are no sensors to acquire sensing data that is necessary during situation inference process, the situation of an entity cannot be deduced. Secondly, even if all needed sensors are available, the confidence of sensing data is unpredictable. There are two reasons for the unpredictability. On the one hand, the sensors to fulfill a sensing goal are by products of other sensing systems rather than application-dedicated, and the accuracy of the same type of sensors vary dramatically from one sensing system to another. On the other hand, it is hard to predict what sensor will be selected to accomplish a sensing goal at runtime.

The above stated problems may be abstracted as how to do semantic reasoning with uncertainty. To solve this problem, various extensions of OWL and SWRL have been proposed with different mathematical theories [12]. These works have proved their validity to varying degrees, but they also have a common deficiency, i.e., the extended languages are often very complicated and hard to be learned, even for those people who are familiar with standard Semantic Web languages.

Therefore, this paper developed a situation inference tool, named *OPSitu*. Instead of extending languages, *OPSitu* provides the developers of situation-aware applications with standard OWL and SWRL to write the situation inference rules, no matter the application will run under opportunistic sensing paradigm or not. The uncertainty of sensing data in opportunistic sensing is handled at runtime by the situation inference engine of *OPSitu* with the help of a pre-built knowledge base.

The rest of this paper is divided into 5 sections. Section 2 presents an example for opportunistic sensing. Section 3 gives a system overview of the *OPSitu*; Sect. 4 introduces the implementation of the situation inference engine in detail.

Section 5 describes the experiment. Section 6 reviews related works. Finally, directions of future works are concluded in Sect. 7.

2 Running Example

A situation-aware application, named *MyClassroom*, is to provide different services for students in classrooms according to their different situations. Thus *MyClassroom* has to identify current situation of a student in the classroom out of a set of possible situations. There are only five possible situations for a student user in the classroom that *MyClassroom* focuses, and they are class attendance, open lecture, student meeting, class exam and self-study. To infer the users situation, there are also five contexts to be exploited and the relevant sensing modules to acquire these contexts are described in Table 1. *MyClassroom* is running under opportunistic sensing paradigm, because there are two contexts whose availability is uncertain, i.e., status of the projector and existence of human voice.

Moreover, to infer the student's situation in classrooms, five rules are given in Table 2, and one row for each possible situation. For example, if a student is in a large classroom, the projector in that room is on, human voice exists in that room, and the acquaintance proportion of Tom in that room is low, then Tom

Table 1. Context and Relevant Sensing Modules

Context	Relevant Sensing Module	Availability
Classroom Capacity	Observed by human and stored in the database	Available
Projector Status	Based on the light sensor on the screen of projector	Uncertain
People Speaks	Based on the microphone on the rostrum	Uncertain
Location	Based on Wi-Fi fingerprint	Available
Acquaintance proportion	Based on the Bluetooth in persons smartphone	Available

Table 2. Situation Inference Rules

Context / ***Situation***	***Location***	***Classroom Capacity***	***Projector Status***	***Human Voice Existence***	***Acquaintance Proportion***
Class Attendance	In Classroom	Small/Mid/Large	On	Yes	High
Open Lecture	In Classroom	Large	On	Yes	Low
Student Meeting	In Classroom	Small/Mid	Off	Yes	High
Class Exam	In Classroom	Mid/Large	Off	No	High
Self-Study	In Classroom	Small/Mid/Large	Off	No	Low

is attending an open lecture. By adopting the Semantic-Web based approach, these inference rules are written in OWL and SWRL. In Fig. 1, the inference rule for specifying "Open Lecture" is written in SWRL in (a), and related concepts appearing in the rule are defined in the ontology model in (b).

3 System Overview

3.1 Key Concepts

For the convenience of description, some concepts are interpreted in the follow.

Situation & Context. In this paper, a situation is the semantic abstraction about the status of an entity and the adaptions of the situation-aware application are triggered with the change of situations. A context is the information for characterizing the situation of an entity, and a situation is specified by multiple contexts based on human knowledge. For the example in Sect. 2, the situation of a student in a classroom is specified by five contexts based on the inference rules in Table 2.

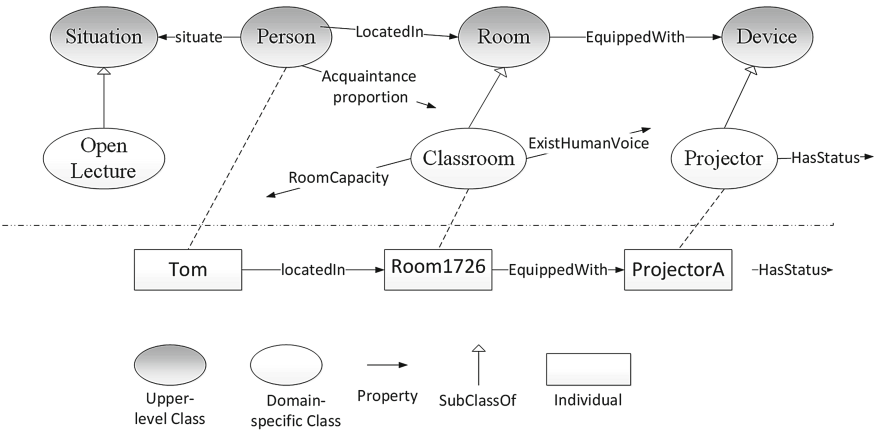
Situation Candidate Set (SCS). Generally speaking, although the possible situations of some entities (for example, a person) are infinite, the situations that an application focuses are limited. Therefore, situation inference can be considered as a classification problem. The candidate situations of an entity form a set, which is referred to as an *SCS (Situation Candidate Set)* in this paper. For the example in Sect. 2, the SCS for *MyClassroom* is S = {Attending Class, Attending Open Lecture, Having Meeting, Taking Exam, Self-Studying}.

Context Assertion (CA). In this paper, *Context Assertion (CA)* is defined as a logic expression describing the condition that a context should be satisfied. For the example in Fig. 1 there are five contexts. Correspondingly, there are five *Context Assertions (CAs)* denoted as $A(C_i)(i = 1, 2, \ldots, 5)$, and they are listed in Fig. 2.

Situation Inference Rule (SIR). The *Situation Inference Rule (SIR)* is a first-order logic expression defining the relationship between contexts and a situation. More specifically, an *SIR* consists of two parts, the antecedent and the consequent. The antecedent part is a set of *Context Assertions (CAs)* connected with each other using logic AND. Thus the antecedent part of an *SIR* for a candidate situation S_i can be represented as $R(S_i) = A(C_1) \wedge A(C_2) \wedge \ldots \wedge A(C_m)$, where $A(C_i)$ is the *ith CA* and the *SIR* is related to m contexts. The consequent part is the logic expressions for a candidate situation. The semantic inference rule in Fig. 1(a) is an example of an *SIR*. Its antecedent part is $A(C_1) \wedge A(C_2) \wedge A(C_3) \wedge A(C_4) \wedge A(C_5)$, where $A(C_i)$ are expressed in Fig. 2 respectively, and its consequent part is $Situate(?p, ?stu) \wedge OpenLecture(?stu)$, which means the person $?p$ is attending an open lecture.

Person(?p)∧ClassRoom(?r)∧LocatedIn(?p,?r)∧RoomCapacity(?r,?cap)∧ equalTo(?cap,'Large')∧Projector(?pro)∧EquippedWith(?r,?pro)∧HasStatus (?pro,?s)∧equalTo(?s,'on')∧ExistHumanVoice(?r,?x)∧ equalTo(?x,'yes') ∧ AcquaintanceProportion(?p, ?y)∧ equalTo(?y,'Low') → Situate(?p,?stu) ∧ OPenLecture(?stu)

(a) Situation inference Rule Written in SWRL: an Example



(b) An Ontology Model Defining Concepts of the Rule in (a)

Fig. 1. Situation Inference Rule: An Example

$A(C_1) = LocatedIn(?p,?r) \wedge ClassRoom(?r) \wedge Person(?p)$ // the person is in a classroom

$A(C_2) = RoomCapacity(?r,?cap) \wedge ClassRoom(?r) \wedge equalTo(?cap,'Large')$ // the room is large

$A(C_3) = HasStatus(?pro,?s) \wedge Projector(?pro) \wedge EquippedWith(?r,?pro) \wedge$

$ClassRoom(?r) \wedge equalTo(?s,'on')$ // the status of projector is on

$A(C_4) = ExistHumanVoice(?r,?x) \wedge Classroom(?r) \wedge equalTo(?x,'yes')$ // human voice exists

$A(C_5) = AcquaintanceProportion(?p,?y) \wedge Person(?p) \wedge equalTo(?y,'Low')$

// acquaintance proportion of the person in the room is low

Fig. 2. Example of context assertions

3.2 Architecture

Figure 3 demonstrates the architecture of *OPSitu* system and some other components that cooperate closely with *OPSitu*, and they are described in the follow.

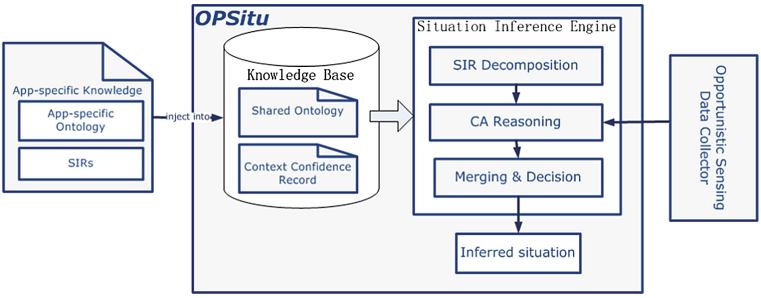


Fig. 3. System architecture of *OPSitu*

Opportunistic Sensing Data Collector. It consists of sensing modules for different contexts, including location, temperature, light, sound, etc. Those modules obtain sensing data from the physical space or the cyberspace and process them into meaningful context information. This part has been done by many existing works [7–9,11], thus we will not discuss it in detail.

Knowledge Base. Situation-aware applications perform the situation inference based on two types of knowledge. One is shared by all applications, and the other is application-specific. *OPSitu* is designed according to this classification.

The knowledge shared by applications is stored and managed in a pre-built *Knowledge Base*. It consists of the *Shared Ontology* and the *Context Confidence Record*. The *Shared Ontology* defines commonly used concepts for all applications as *Class* and *Property* in OWL (Web Ontology Language). To address the unpredictability of sensing data's confidence pointed out in Sect. 1 the *Context Confidence Record* pre-stores the confidence of contexts, which is measured by the accuracy of the sensing data collector. At runtime, the situation inference engine can query the *Context Confidence Record* and utilize them in the inference process.

Application-specific knowledge is injected into the *Knowledge Base* by application developers. It is comprised of the *App-specific Ontology* and the *SIRs*. The *App-specific Ontology* is derived from the *Shared Ontology*. Therefore, it not only contains all concepts in the *Shared Ontology* but includes some additional concepts just for a specific application. *SIRs* are logic expressions defining the relationships between contexts and situations, and they are also application-specific.

Since the management of knowledge base is a mature technology and there are many existing tools [13], *OPSitu* directly adopts Protégé [14], a free open-source Java tool, to support the creation and management of knowledge in OWL and SWRL.

Situation Inference Engine. The *Situation Inference Engine* is to conduct situation inference with uncertainty at runtime, and it is on the basis of the knowledge and opportunistic sensing data. It consists of three modules, *SIR*

Decomposition, *CA Reasoning* and *Merging & Decision*. Compared with the sensing data collector and knowledge base, the design and implementation of *Situation Inference Engine* is more challenging due to its complexity. Thus it is the main contribution of this paper, and we will describe it in detail in Sect. 4.

4 Situation Inference Engine Implementation

For a semantic reasoner that only supports the certain reasoning, two conditions must be satisfied in order to infer the situation of an entity. Firstly, an inference rule is considered as a whole. Secondly, before the inference process is activated, some variables in the rule must be assigned with specific value. However, this is not compatible with opportunistic sensing, because the value of some variable may not be determined when corresponding sensors are not available. To address this problem, the *Situation Inference Engine* adopts an inference process including the following three steps. Firstly, it decomposes the *SIR* into several *CAs* at first. Secondly, it performs the reasoning for the *CAs* whose context can be determined at runtime. Thirdly, it merges the reasoning results of all *CAs* and makes a decision about which candidate situation is the most possible.

4.1 Semi-automatic SIR Decomposition

Although it is easy for human to recognize what is a *CA* in an *SIR*, it is difficult to make *OPSitu* smart enough to decompose an *SIR* into *CAs* in a fully-automatic way. Thus, we come up with a semi-automatic strategy, and it consists of following two steps.

Step 1: Sensible Atomic Formula Selection. After finish writing an *SIR* at design time, the developer is required by the system to select the atomic formula that are directly related to sensing data (either from physical sensor or cyberspace), which are referred to as *Sensible Atomic Formula* in this paper. For the *SIR* in Fig. 1 (a), five atomic formulas, $LocatedIn(?p, ?r)$, $RoomCapacity(?r, ?cap)$, $HasStatus(?pro, ?s)$, $ExistHuman\ Voice(?r, ?x)$, and $Acquaintance\text{-}Proportion(?p, ?y)$ should be selected by the developer as *Sensible Atomic Formula* in this step.

Step 2: Runtime Decomposition. At runtime, the *SIR Decomposition* module will decompose an *SIR* into several *CAs* based on the *Sensible Atomic Formula* that developer has selected. For each Sensible Atomic Formula, its related atomic formulas including itself are combined together with logic AND as a CA. Take $LocatedIn(?p, ?r)$ as an example. $?p$ relates to $Person(?p)$, and $?r$ relates to $ClassRoom(?r)$. Therefore, three atomic formulas, $LocatedIn(?p, ?r)$, $Person(?p)$ and $ClassRoom(?r)$, are connected together with logic AND as a *CA* $A(C_1)$. Similarly, $A(C_2)$, $A(C_3)$, $A(C_4)$ and $A(C_5)$ becomes another four *CAs* after the decomposition phase, and they are listed in Fig. 2.

4.2 Topological-Ordering Based CA Reasoning

After the decomposition, *OPSitu* directly exploits Pellet to conduct the reasoning for each *CA* whose context can be acquired. However, the reasoning of each *CA* is not independent, and this gives *OPSitu* an opportunity to improve its reasoning performance. Let us take $A(C_1)$, $A(C_3)$ and $A(C_4)$ in Fig. 2 as an example to illustrate the dependency issue.

Before runtime reasoning, some variables in a *CA* have to be assigned with a specific value. For instance, the value of variable $?r$ must be assigned before the reasoning of $A(C_3)$ and $A(C_4)$. This is because only when the room is specified, whether human voice exists and the projector's status in that room can be determined. Moreover, if one wants to determine where Tom is located in, a query must be issued by using the OWL API [15] *getObjectPropertyValues(Tom, LocatedIn)*. Therefore, the reasoning of $A(C_3)$ and $A(C_4)$ depends on $LocatedIn(?p, ?r)$. Here we define the dependency between *CAs* in Definition 1. According to this definition, $A(C_3)$ and $A(C_4)$ depends on $A(C_1)$.

Definition 1. *If the reasoning of $A(C_i)$ depends on the Sensible Atomic Formula of $A(C_j)$, then $A(C_i)$ depends on $A(C_j)$.*

In fact, after the reasoning of $A(C_1)$, the value of $?r$ (a specific room) has already been determined. Consequently, if the following two conditions are satisfied, the *OPSitu* does not need to query the value of $?r$ when reasoning $A(C_3)$ and $A(C_4)$, thus improving its reasoning performance.

Condition 1: OPSitu performs the reasoning of $A(C_1)$ before $A(C_3)$ and $A(C_4)$.

Condition 2: OPSitu records the value of $?r$ as an intermediate result after the reasoning of $A(C_1)$.

Based on the analysis above, we propose a method for arranging a reasonable reasoning order so as to enhance the reasoning performance. It consists of two steps, the dependency analysis and the Topological-Ordering based reasoning.

Step 1: Dependency Analysis. In this step, OPSitu will analyze the dependency among all *CAs* of an *SIR*. In this process, the dependency analysis is designed as the generation of a directed graph, in which a *CA* is a vertex, and the dependency between two *CAs* is a directed edge linking two vertexes.

Step 2: Topological-Ordering Based Reasoning. After the dependency analysis, all *CAs* of an *SIR* are to be arranged in a topological order based on the Topological Ordering algorithm. Then the *CAs*, whose context can be acquired, will be reasoned one by one according to the topological order. Since the Topological Ordering is a well-known algorithm and the reasoning of *CAs* is based on the open-source semantic reasoner (the Pellet), we will not describe the ordering and reasoning in detail.

4.3 Similarity-Based Merging and Decision

After the reasoning of all *CAs*, the reasoning results would be merged together to compute the possibility of each candidate situation based on a similarity function, and to make a decision about which situation is the most possible one.

To describe the merging and decision phase, some concepts should be defined at first.

Firstly, the truth-value of a CA is extended from the conventional $0/1(false/true)$ to the interval $[-1, 1]$. The absolute value of the truth-value indicates the confidence of the context, and the positive/negative symbol represents the assertions tendency of being true or false. The semantic interpretation of this extension is represented in Formula 1, in which $k \in (0, 1]$ is the confidence of context C_i obtained from the *Context Confidence Record* in the knowledge base.

$$TruthValue((A(C_i)) = \begin{cases} k & \text{if } \textit{acquired } C_i \text{ indicates that}(A(C_i)) \text{ is true} \\ 0 & \text{if the } C_i \text{ can not be acquired} \\ -k & \text{if } \textit{acquired } C_i \text{ indicates that}(A(C_i)) \text{ is false} \end{cases} \tag{1}$$

Secondly, *CTV (Contexts Truth Vector)* and *BV (Benchmark Vector)* are defined in Definition. In this paper, we assume that all situations in an *SCS* are based on a common set of *CAs*.

Definition 2. *An* SCS *is denoted as* $S = S_1, S_2, \ldots, S_n$ *the objective of situation inference is to find the situation that most likely to be from* S*. At a given time* t*, the antecedent of an* SIR *for* $S_i \in S$ *is denoted as* $R(S_i) = A(C_1) \wedge A(C_2) \wedge \cdots \wedge A(C_m)$*, where* $A(C_i)$ *is a CA and the* $R(S_i)$ *is related to* m *contexts.* CTV (Contexts Truth Vector) *and* BV (Benchmark Vector) *are defined in (a) and (b)*

(a) Denote $TV_t(S_i) = (T_1, T_2, \ldots, T_m)$ *as* CTV (Contexts Truth Vector) *of* $R(S_i)$*, where* $T_i = TruethValue(A(C_i))$*, and* m *is the number of contexts.*

(b) Denote $bV = (1, 1, \ldots, 1)$*, a* m*-dimension vector, as the* BV (Benchmark Vector).

Thirdly, we define a similarity $functionSim(S(t), S_i)$ to represent the similarity between S_i and $S(t)$ in Formula 2, where $S_i \in S$ and $S(t)$is the actual situation at time t. It is measured by the cosine similarity between *Contexts Truth Vector* TV_t and *Benchmark Vector* bV.

$$Sim(S(t), S_i) = \cos(TV_t(S_i), bV) = \frac{TV_t(S_i) \cdot bV}{|TV_t(S_i)||bV|} \tag{2}$$

Finally, our merging and decision phase is described in Fig. 4. The main idea of this process is to compute the degree of possibility of each candidate situation, and the possibility is measured by the a similarity function $Sim(S(t), S_i)$. Then the candidate situation with maximum possibility is considered to be the inferred situation.

Denote a *SCS* as $S = \{S_1, S_2, \ldots, S_n\}$, for a specific *SIR*'s antecedent $R(S_i) = A(C_1) \wedge A(C_2) \wedge \ldots \wedge A(C_m)$, where $A(C_i)$ is a *CA* and the *SIR*'s antecedent $R(S_i)$ is related to m contexts $C_1, C_2, \ldots, C_m$.

Step 1: For $i = 1, 2, \ldots, n$, computes the value of $Sim(S(t), S_i)$ using formula (2).

Step 2: Make a decision about which situation is most likely to be at time t(denoted as $S_{inferred}(t)$). $S_{inferred}(t) = S_k$, if S_k let $Sim(S(t), S_i)$ to be maximum. $i = 1, 2, \ldots, n$.

Step 3: If there is more than one S_i whose $Sim(S(t), S_i)$ reach to the maximum value, randomly choose one of them as the inference result.

Fig. 4. The merging and decision phase description

5 Experimental Evaluation

5.1 Experimental Methodology

To evaluate the validity of *OPSitu*, an experiment has been conducted based on the example in Sect. 2. Firstly, we construct and store *SIRs* in Table 2 into SWRL format in the *Knowledge Base*. Secondly, the sensing modules in Table 1 serve as the *Opportunistic Sensing Data Collectors*.

After establishing the *SIRs* and sensing data collectors, our experiment comprises two steps:

Step 1: Context Confidence Generation. We generate the confidence of contexts in two ways. One way is through experiment. For example, the confidence of projector status is generated by experiment. As the length of the paper is limited, we put the details of four experiments on the website [16]. The other way is set by experience. For example, as the classroom capacity is observed by human and stored in a database, its confidence is set to be a constant 100 % without experiment. The generated confidence of each context is listed in Table 3, and we store them in the Context Confidence Record of *OPSitu's Knowledge Base*.

Table 3. Context Confidence

Context	Confidence	Generation Method
Classroom Capacity	100 %	Experience
Projector Status	100 %	Experiment
People Speaks	82.5 %	Experiment
Location	93.2 %	Experiment
Acquaintance Proportion	94 %	Experiment

Step 2: Simulative Situation Inference. The parameters of this simulative inference are the confidence of each context, which are generated by the real-world experiment above. The process of the simulative situation inference is described in Fig. 5. In Step A and B, we simulate an opportunistic sensing environment by programming, and then adopt the *Situation Inference Engine* of *OPSitu* to infer the situation in Step C. For the simulation of the opportunistic sensing environment, there are two points need to be explained.

(1) Based on a survey in university P, in the step B (2) 90 % of all virtual classrooms are randomly selected to be equipped with light sensors, and 95 % to be equipped with microphone.

(2) In the Step B (3) we assign the value of contexts based on the corresponding *SIR*. This is because the objective of this experiment is to evaluate the validity of *Situation Inference Engine of OPSitu* rather than the reliability of *SIR*, thus we assume that all *SIRs* are reliable in this simulative inference.

Step A: Create N virtual Classrooms: R_1, R_2, …, R_N.
Step B: for (i = 1 to N) {
(1) Randomly assign a situation for R_i;
(2) Assign the availability of sensors opportunistically for R_i;
(3) Assign value to context based on *SIR*s. }
Step C: for (i = 1 to N) {
(1) Adopt the *OPSitu*'s *Situation Inference Engine* to infer situation;
(2) Compare the inferred situation with the assigned one. }

Fig. 5. Situation inference in a simulative opportunistic sensing environment

5.2 Experimental Result

We set the number of virtual classroom N to be 10000. The experimental result is demonstrated as the confusion matrix in Table 4. By analyzing the confusion matrix in Table 4, the overall situation inference accuracy by *OPSitu* reaches to 94.9 % in such a simulative opportunistic sensing environment.

The misclassification is caused when the key sensing data to classify similar situations is missing. For example, when the light sensor is not available, class attendance and student meeting are easy to be misclassified. Therefore, the limitation of *OPSitu* is that the fewer contexts are determined, the lower inference accuracy would be. However, the opportunistic sensing paradigm has a basic assumption that the sensors are abundant enough in the environment where the application is expected to be used [9]. Thus under this assumption, *OPSitu* can conduct Semantic-Web based situation inference with a satisfactory accuracy.

Table 4. Situation Inference Confusion Matrix

Context / *Situation*	*Class Attendance*	*Open Lecture*	*Student Meeting*	*Class Exam*	*Self-Study*
Class Attendance	1830	0	162	0	0
Open Lecture	0	1990	0	0	10
Student Meeting	155	0	1762	83	0
Class Exam	0	0	89	1911	0
Self-Study	0	9	0	0	1991

6 Related Work

To deal with uncertainty in the Semantic Web and its applications, many researchers have proposed extension of Semantic Web language with special mathematical theories. [17,18] extended OWL based on the probability theory. Reference [19] proposed an extension for terminological logics with the possibility theory. Reference [20–23] extended either OWL or SWRL based on fuzzy logic, etc. In addition to language extension, some of these work developed corresponding semantic inference engines. In terms of the expressiveness, those extensions for semantic web languages are capable of dealing with the uncertainty brought by opportunistic sensing. However, these extended languages are often complicated and hard to learn, even for those who are familiar with the standard semantic web languages. Besides, there are already many situation inference rules written in OWL and SWRL on the Semantic Web. If we want to share and reuse this existing knowledge in opportunistic sensing applications, they have to be transformed into the format of those extended languages. However, since the extended languages are quite complex, the transformation process is very time-consuming.

To avoid the shortcomings of the language extension approaches above, [24] provided a guidance to use OWL and SWRL to express fuzzy semantic rules. This approach models the binary predicate with uncertainty as a 3-ary predicate. Since SWRL is a rule language only supporting unary and binary predicates, this paper adopts a procedure called the reification to express a 3-ary relation via unary and binary relations. Therefore, under this guidance, developers can express fuzzy rules without the modification of OWL and SWRL. However, the reification process is very complicated and it is conducted by developers. Besides, the rules after the reification process, although expressed by SWRL, are too complex to be understood.

7 Conclusion

In order to solve the problem of uncertainty during situation inference brought by the opportunistic sensing paradigm, this paper proposed *OPSitu*, a Semantic-Web based situation inference tool. *OPSitu* enables the developers of opportunistic sensing applications to write the situation inference rules with standard OWL

and SWRL, and utilizes a pre-built knowledge base to handle the uncertainty at runtime.

The future work about *OPSitu* include two parts. Firstly, taking the weights of contexts into consideration. In some cases, different contexts contribute to the inference of a situation to different degrees. However, the current version of *OPSitu* does not consider this aspect. Hence we plan to take the weight of context into consideration in the next version of *OPSitu*. Secondly, this paper assumes that all candidate situations in an *SCS* are based on the same set of contexts. However, in some circumstances, this may not be true. Thus we plan to revise the inference engine of *OPSitu* to make it able to handle more complex conditions.

Acknowledgments. This work is funded by the National High Technology Research and Development Program of China (863) under Grant No. 2013AA01A605, the National Basic Research Program of China (973) under Grant No. 2011CB302604 and the National Natural Science Foundation of China under Grant No.61121063.

References

1. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: a review. Pervasive Mob. Comput. **8**(1), 36–66 (2012)
2. Goix, L.-W., Valla, M., Cerami, L., Falcarin, P.: Situation inference for mobile users: a rule based approach. In: 2007 International Conference on Mobile Data Management, pp. 299–303. IEEE (2007)
3. Matheus, C.J., Baclawski, K., Kokar, M.M., Letkowski, J.J.: Using SWRL and OWL to capture domain knowledge for a situation awareness application applied to a supply logistics scenario. In: Adi, A., Stoutenburg, S., Tabet, S. (eds.) RuleML 2005. LNCS, vol. 3791, pp. 130–144. Springer, Heidelberg (2005)
4. Oberhauser, R.: Leveraging semantic web computing for context-aware software engineering environments. In: Wu, G. (ed.) Semantic Web. In-Tech, Vienna (2010)
5. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.Q.: Ontology based context modeling and reasoning using owl. In: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004, pp. 18–22. IEEE (2004)
6. Yau, S.S., Wang, Y., Karim, F.: Development of situation-aware application software for ubiquitous computing environments. In: Proceedings of 26th Annual International Computer Software and Applications Conference, COMPSAC 2002, pp. 233–238. IEEE (2002)
7. Hoseini-Tabatabaei, S.A., Gluhak, A., Tafazolli, R.: A survey on smartphone-based systems for opportunistic user context recognition. ACM Comput. Surv. (CSUR) **45**(3), 1–27 (2013)
8. Roggen, D., Lukowicz, P., Ferscha, L., del Mill, R., Tröster, G., Chavarriaga, R., et al.: Opportunistic human activity and context recognition. Computer **46**, 36–45 (2013)
9. Conti, M., Kumar, M.: Opportunities in opportunistic computing. Computer **43**(1), 42–50 (2010)
10. Ferscha, A.: 20 years past weiser: what's next? IEEE Pervasive Comput. **11**(1), 52–61 (2012)

11. Kurz, M., Hölzl, G., Ferscha, A., Calatroni, A., Roggen, D., Tröster, D., Sagha, H., Chavarriaga, R., Millán, J.D.R., Bannach, D., et al.: The opportunity framework and data processing ecosystem for opportunistic activity and context recognition. Int. J. Sens. Wireless Commun. Control, Special Issue on Autonomic and Opportunistic Communications **1**, 102–125 (2011)
12. Stoilos, G., Simou, N., Stamou, G., Kollias, S.: Uncertainty and the semantic web. IEEE Intell. Syst. **21**(5), 84–87 (2006)
13. Schmidt, J.W., Thanos, C.: Foundations of knowledge base management: Contributions from logic, databases, and artificial intelligence applications (2012)
14. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The evolution of protégé: an environment for knowledge-based systems development. Int. J. Hum Comput Stud. **58**(1), 89–123 (2003)
15. Bechhofer, S., Volz, R., Lord, P.: Cooking the semantic web with the OWL API. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 659–675. Springer, Heidelberg (2003)
16. http://219.143.213.95/experiments/
17. Ding, Z., Peng, Y.: A probabilistic extension to ontology language owl. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004, pp. 1–10. IEEE (2004)
18. Ding, Z., Peng, Y., Pan, R.: A Bayesian approach to uncertainty modelling in owl ontology. Technical report, DTIC Document (2006)
19. Hollunder, B.: An alternative proof method for possibilistic logic and its application to terminological logics. Int. J. Approximate Reasoning **12**(2), 85–109 (1995)
20. Pan, J.Z., Stoilos, G., Stamou, G., Tzouvaras, V., Horrocks, I.: f-SWRL: A Fuzzy Extension of SWRL. In: Spaccapietra, S., Aberer, K., Cudré-Mauroux, P. (eds.) Journal on Data Semantics VI. LNCS, vol. 4090, pp. 28–46. Springer, Heidelberg (2006)
21. Stoilos, G., Stamou, G.B., Tzouvaras, V., Pan, J.Z., Horrocks, I.: Uncertainty and the semantic web. In: OWLED, Fuzzy owl (2005)
22. Wang, X., Ma, Z.M., Yan, L., Meng, X.: Vague-SWRL: a fuzzy extension of SWRL. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 232–233. Springer, Heidelberg (2008)
23. Wlodarczyk, T.W., Rong, C., O'Connor, M., Musen M.: Swrl-f: a fuzzy logic extension of the semantic web rule language. In: Proceedings of the International Conference on Web Intelligence, Mining and Semantics, pp. 1–39. ACM (2011)
24. Ciaramella, A., Cimino, M.G.C.A., Marcelloni, F., Straccia, U.: Combining fuzzy logic and semantic web to enable situation-awareness in service recommendation. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 31–45. Springer, Heidelberg (2010)