An Interval-Based Context Reasoning Approach

Tao Lu, Xiaoling Liu, Xiaogang Liu, Shaokun Zhang
System Engineering Institute
Dalian University of Technology
Dalian, China

Abstract— **Context-aware computing is an emerging computing paradigm that provides intelligent context-aware application. Context reasoning is an important aspect in context awareness, by which high level context can be derived from low-level context data. In this paper, we focus on the situation in mobile workspace, where a worker performs a set of activities to archive defined goals. The main part of being aware is to be able to answer the question of "what is going on". Therefore high level context we need to derive is current activity and its state. The approach we propose is knowledge-driven technique. Temporal relations as well as semantic relations are integrated into the context model of activity, and the recognition is performed based on the model. We first define the context model of activity, and then we analyze the characteristics of context change and propose a method of context reasoning.**

Keywords- context-aware; context reasoning; interval-based; activity recognition.

I. Introduction

Context-aware computing is an emerging computing paradigm that provides intelligent context-aware application. According to well-known definition proposed by Dey, context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves [1]. Context-awareness means exploiting context information to provide adaptive information or service, or reducing the interaction between user and application.

Context-awareness is related to the manipulation of context information pertaining to certain entities. Information from physical sensors are called low-level context. High level context, also called situation sometimes, can be derived from low-level context by proper interpreting. This process is context reasoning, or context interpretation. Situations are semantic abstractions from low-level contexts cues. So the relationship between low level context and situation must be integrated into a context model, which represents human knowledge about the world [2].

This can either be done by specification, i.e. human defines situations and their relationship based on his /her knowledge, or the model are learned automatically using machine learning techniques [2]. The two approaches are also called knowledge-driven approach and data-driven approach [3].

Data-driven techniques are based on the machine learning methods and are well suited for recognizing simple activities and gestures from raw sensor data or video data [4]. A wide range of algorithms and models include Hidden Markov Models [5], dynamic and naïve Bayes networks [6], and decision trees [7] and so on. Knowledge-driven techniques, concerning knowledge representation as well as reasoning with them, is closely related with classical topics in artificial intelligence [8]. Ontology-based method is one of frequently used technique. For example, Chen has proposed a technique to recognize activities through ontological reasoning [9]. Knowledge-driven techniques can also be used to recognizing complex situations based on the recognized simple context [10].

In this paper, we focus on the situation in mobile workspace, where a worker performs a set of activities to archive defined goals. The main part of being aware is to be able to answer the question of "what is going on". Therefore the situation, or high level context, we need to derive is activity and its state.

Determining the activity and its state cannot only be completed by observing the context but also need to draw conclusion from the observations. Knowledge about the relation of context and activity should be integrated into the model. Compared with most of knowledge-driven techniques, the approach we propose considers the temporal relation between activities and contexts.

Activity is seemed as process with duration, and context may change in the process. Each activity context model is a set of semantic relations and temporal constraints with respect to the activity and the context. If some observed contexts match the defined patterns, or if their times of occurrence meet the specified constraints, then an instance of this situation occurs.

We have developed an activity recognition approach taking the temporal relation into account in previous work [11]. This paper considers more temporal relations. Moreover semantic relations are also integrated into the model. By this means, more types of situation can be recognized.

The rest of the paper is organized as follows. Section 2 provides the context model of activity. Section 3 proposes the activity recognition approach. Case study is described in section 4 and section 5 concludes the paper.

II. Context Model Of Activity

Suppose the application is related to a set of contexts which are denoted as $c_1,c_2,...c_n$, and the domain of c_i is D_i. At given time t, the value of context c_i is denoted as $c_i(t)(c_i(t)\in D_i)$, $i=1,2,\ldots n$. The values of context $c_1,c_2,...c_n$ at time t is denoted as $C(t)$, $C(t)=(c_1(t),c_2(t),\ldots c_n(t))$.

Suppose $d_j^{(i)} \subseteq D_i$, $(c_i, d_j^{(i)})$ is called a context pattern. The context pattern$(c_i, d_j^{(i)})$ holds at time t, if and only if $c_i(t) \in d_j^{(i)}$, denoted as $hold\_at((c_i, d_j^{(i)}), t)$, i.e. $hold\_at((c_i, d_j^{(i)}), t) \Leftrightarrow c_i(t) \in d_j^{(i)}$.

Apparently $hold\_at((c_i, d_j^{(i)}), t) \Leftrightarrow \neg hold\_at((c_i, D_i\text{-}d_j^{(i)}), t)$, $(c_i, D_i\text{-}d_j^{(i)})$ is called the negative pattern of $(c_i, d_j^{(i)})$, denoted as $\neg(c_i, d_j^{(i)})$.

We define a special context pattern time-elapse(t_s,T), $hold\_at(time-elapse(t_s, T), t) \Leftrightarrow t = t_s + T$. t_s is the time to start timing. The context pattern holds after T. For simplicity, we use time-elapse instead of time-elapse(t_s,T) . Initializing time-elapse in this paper means setting the time length and start timing.

A. Temporal relations of context pattern and activity

Context pattern may hold over time interval. We consider time as a linearly ordered discrete set of instants, a time interval is represented as an ordered pair of time points representing starting and ending time. Context pattern may hold in one or more time intervals.

We use $I(p)$ to denote the interval set in which the context pattern p holds, i.e.

$$I(p)=\{[t_s,t_e] \mid \forall t \in [t_s,t_e],\ hold\_at(p,t) \text{ and } \neg\exists\ [t_m,t_n],\ t_m \le t_s, t_n \ge t_e, [t_m,t_n] \ne [t_s,t_e],\ \forall t \in [t_m,t_n],\ hold\_at(p,t)\} \quad (1)$$

In same way, we can use an ordered pair to represent the time interval in which an activity is conducted. The starting time and ending time of activity a are denoted as $a.start$ and $a.end$.

Allen has suggested a well-known temporal model based on relationships among intervals [12]. Here we use these predicates to model the temporal relations between context patterns and activities.

Let P be a context pattern set and A an activity set. *Equal*, *During*, *Start*, *Finish*, *Before*, *Overlap* are predicates representing relations between context patterns and activities. The meaning of the predicate is defined in TABLE I.

In TABLE I, we only list the relations usually used in our model. Other relations are not difficult to deduce from the meaning.

TABLE I. TEMPORAL RELATION OF CONTEXT PATTERN AND ACTIVITY

Relation	Definition
$Equal(p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}= a.start,\ t_{pe}= a.end$
$During(p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps} \ge a.start,\ t_{pe} \le a.end$
$During(a,p)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps} \le a.start,\ t_{pe} \ge a.end$
$Start(p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}= a.start,\ t_{pe}< a.end$
$Start(a,p)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}= a.start,\ t_{pe}> a.end$
$Finish(p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}> a.start,\ t_{pe}=a.end$
$Finish(a,p)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}< a.start,\ t_{pe}=a.end$
$Overlap(p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}< a.start,\ t_{pe}< a.end$
$Overlap(a,p)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}> a.start,\ t_{pe}>a.end$
$Before\ (p,a)$	$\exists\ [t_{ps},t_{pe}] \in I(p),\ t_{ps}< a.start$

B. Semantic relations of context pattern and activity

Besides temporal relations, there are also semantic relations existing between context patterns and activities. COND, CONSEQ, PREM, ACCOMP are semantic relations defined to illustrate logic connection in context patterns and activities. The meaning and corresponding temporal relations are listed in TABLE II.

TABLE II. SEMANTIC RELATION CONTEXT PATTERN AND ACTIVITY

Relation	Meaning	Corresponding temporal realtion
COND(p,a)	Context pattern p is a condition of conducting activity a. If it is not satisfied as its corresponding temporal pattern, the activity is abnormally conducted.	$Before(p,a)$, $During(a,p)$, $Overlap(p,a)$
PREM(p,a)	Context pattern p is a premise of conducting activity a. If it is not satisfied as its corresponding temporal pattern, the activity is not started or interrupted.	$Overlap(p,a)$, $During(a,p)$, $Finish(a,p)$
ACCOMP(p,a)	Context pattern p will occur as its corresponding temporal pattern if the activity a is conducted normally	$Start(p,a)$, $Equal(p,a)$, $During(p,a)$, $Finish(p,a)$, $Overlap(a,p)$
CONSEQ(p,a)	Context pattern p is a consequence of activity a.	$Overlap(a,p)$

If COND(p,a), p is called a condition context pattern of a. In same way, for PREM(p,a), ACCOMP(p,a), CONSEQ(p,a), p are called premise context pattern, accompanying context pattern and consequence context pattern respectively.

C. *Modeling activity with semantic relation and temporal relation*

We can model the relation of activity and context pattern with semantic relation and temporal relation. These relations show the context changing rules when the activity is performed normally without interruption.

For real world problem, some semantic relations are not easily to be distinguished. For example, condition and premise are very similar. Here the difference between condition context pattern and premise context pattern is that the condition context pattern can be controlled by human, while the premise is objective. That is if a condition context pattern does not hold in proper time as defined, the activity can be performed but not normally. However, if a premise context pattern does not hold, it can be deduced that the activity has not been started.

The difference between consequence and accompanying context pattern is that the consequence is related with the goal of the activity, and the pattern holding means the activity will end or has ended, while the accompanying context pattern holding may means the activity has been started and if it does not occurred as defined, there may be interruption or abnormal situation.

TABLE II shows the possible temporal relation of every type semantic relation. The temporal relation should be determined when defining an activity. The different temporal relation is used differently.

When context interpretation is performed, the occurring context is compared with the model, and current high level context, i.e. the activity and its state is deduced.

III. Recognizing Activity By Context Reason

Generally in context-aware system, sensors (either virtual sensors or environmental sensors) are used to acquire the raw context. The acquired low-level context data are dealt with by many context-aware middleware or infrastructure and higher level context information are available and represented in formal format. Our work is to derive context in much higher level based on these context information, i.e. recognize the activity and its current state.

The input of our work is sequence of context values in different time, $c(t_1),c(t_2),\ldots c(t_n)$. Assume that every meaningful context change can be detected, and the values are omitted if there is no change happening. So if we have $c(t_i)$ and $c(t_{i+1})$, for any t, $t_i \le t < t_{i+1}$, $c(t)=c(t_i)$. We need to identify the activity and its state according to these context values and activity model.

There are two opposite strategies of processing context data. The first one analyzes previously received context, matching them with related temporal pattern on the basis of their semantic meanings and give the result. This strategy requires less memory but may introduce a higher processing delay. The second approach processes data incrementally, recognizing and stores partial consequence in the form of automata as soon as they are detected. This may need more memory (memorizing different state) but speed up the processing. We adopt the second strategy.

A. *State and internal state*

In activity models, temporal relations define the rules of context pattern occurring. Combined with their semantic meanings, these rules can be used to interpret "what is going on".

Definition 1 Let $a \in A$, *s-pattern*(a) is called **start pattern** of activity a, if and only if:

$$s\text{-}pattern(a)=\{p \mid \mathrm{PREM}(p,a) \vee (\mathrm{ACCOMP}(p,a) \wedge (start(p,a) \vee equal(p,a) \vee start(a,p))\} \quad (2)$$

Starting pattern of an activity is a context pattern set, the pattern in which should hold when the activity is started. Therefore according to generally underlying assumption of context reason, it can also be used to determine the start of the activity.

When an activity a ends normally, its consequence context pattern should hold. Otherwise, if there is an accompanying or premise context pattern p, *equal*(p,a), or *finish*(p,a), p should cease holding, i.e. $\neg p$ holds, when the activity ends.

Definition 2 Let $a \in A$, *e-pattern*(a) is called **end pattern** of activity a, if and only if:

$$e\text{-}pattern(a)=\{p \mid \mathrm{CONSEQ}(p,a) \vee (\mathrm{PREM}(\neg p,a) \wedge finish(a, \neg p)) \vee (\mathrm{ACCOMP}(\neg p,a) \wedge (finish(\neg p,a) \vee equal(\neg p,a)))\} \quad (3)$$

Let $e\text{-}pattern(a,t)=\{p \mid p \in e\text{-}pattern(a) \wedge hold\_at(p,t)\}$, apparently $e\text{-}pattern(a,t)$ is a subset of $e\text{-}pattern(a)$, which can be used to measure how close to the end for the activity process. If $e\text{-}pattern(a,t_{i-1}) \subset e\text{-}pattern(a,t_i)$, the state at time t_i is closer to activity end than last moment. This is useful when the activity is judged to be ended, but meaningless when the activity is still proceeding normally.

When an activity is being conducted without interruption, its accompanying context pattern should always hold if its temporal relations with the activity is *equal*, so should the premise context pattern if the relation is *during* (i.e. activity during context pattern). If for an accompanying context pattern p, *finish*(p,a), or overlap(a,p), p should be hold once it occurs until or after the activity end. Therefore, when the activity is being conducted without interruption, the conditions that the context should satisfy are changing.

Definition 3 Let $a \in A$, *o-pattern*(a) is called an **on pattern** of a, if and only if :

$$o\text{-}pattern(a) \subseteq \{p \mid (\mathrm{PREM}(p,a) \wedge During(a,p)) \vee (\mathrm{ACCOMP}(p,a) \wedge (Equal(p,a) \vee Finish(p,a) \vee Overlap(a,p)))\}$$
$$\text{and } o\text{-}pattern(a) \supseteq \{p \mid (\mathrm{PREM}(p,a) \wedge During(a,p)) \vee (\mathrm{ACCOMP}(p,a) \wedge Equal(p,a))\} \quad (4)$$

An activity may have more than one on patterns. Among them, if:

$$o\text{-}pattern(a)=\{p \mid (\mathrm{PREM}(p,a) \wedge During(a,p)) \vee (\mathrm{ACCOMP}(p,a) \wedge (Equal(p,a) \vee Finish(p,a) \vee Overlap(a,p)))\} \quad (5)$$

Any on pattern of a is the subset of *o-pattern*(a), denoted as *max-on-pattern*(a).

Proposition 1 Let $o\text{-}pattern(a,t) = \{p \mid p \in max\text{-}on\text{-}pattern(a) \wedge hold\_at(p,t)\}$, if the activity a is performing at time *t*, and there is no context change until next moment *t*+1, then the activity *a* is performing without interruption if and only if: $o\text{-}pattern(a,t+1) \supseteq o\text{-}pattern(a,t)$.

Since the above proposition is easy to be proved according to the definitions, proof is omitted here.

However, context value at *t* satisfies the start pattern or end pattern, i.e. all patterns in *s-pattern* or *e-pattern* hold, does not mean the activity starts or ends normally. There are other condition should be checked. Similarly, if the context values over the interval satisfy the on patterns of an activity does not mean the activity is performed normally, even if no interruption happens. We will illustrate this afterwards.

There are 4 types of state to be recognized, called output state. The states are: *waiting* (for activity beginning), *on* (one activity is conducted), *suspend* (one activity is interrupted), *abnormal*. The recognizing result is denoted as 2-tuple:(activity name, state). The state transition is shown in Figure 1.

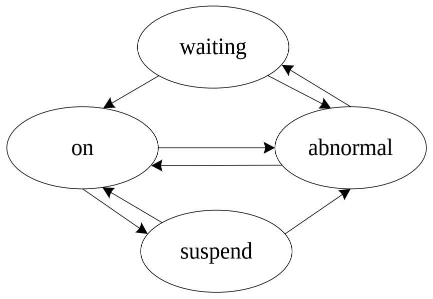

Figure 1. Output state transition.

In addition to above states, we define other 9 states which are used for system control, called internal states. The concept of internal state is actually state of automate, which has process memory itself when it is reached. The states are: *waiting*, *starting*, *on*, *suspend*, *ending*, *end*, *start abnormal*, *process abnormal*, *end abnormal*.

Among them, *waiting*, *on*, *suspend* have similar meanings with output states of same name. State *starting* means stage that all premise conditions are satisfied and part of accompanying context patterns have occurred but the start pattern is not satisfied. State *ending* means consequence context has occurred, or for the activity without consequence context pattern interruption lasts too long, hypothesis of activity ending need to be proved. If end pattern is satisfied, or time is longer enough, system converts to end state and check the condition to determine if it is normal. Therefore state *end* is actually a checking point without duration and there is no corresponding output state. Three abnormal states corresponds *abnormal* output state. Internal state *starting* is *start* when output. Since mostly there is no a time point in which all the start patterns are satisfied at same time, setting an internal state *starting* can avoid taking this situation as abnormal. Internal state transition is shown in Figure 2.

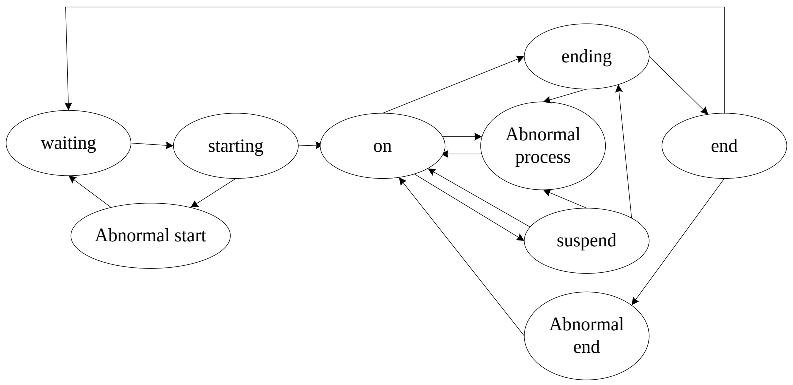

Figure 2. Output state transition.

Since internal state has more process information, analyzing previous context is avoided by this mean. This can be shown below. Internal state is related with the recognizing process, so we focus on the internal state in this paper and *waiting* (*on*, *suspend*) means internal state unless it is specifically explained.

B. Memorizing occurred context

For some of temporal relation, such as *Before* and *During*, determining whether they are satisfied depends on the previous context information. For example, if *During*(*p*,*a*), and *p* does not hold at time *t*, we can’t determine whether it is normal.

Only when the activity ends and p has never occurred, it is determined that activity is not performed normally. In order to avoid checking previous context which may introduces high processing delay, we use some flag to memorize some special occurred context.

We use a set b-COND(a) to record the context patterns which need to hold before activity a is started. The initial value of b-COND(a) is b-COND(a) $=\{p \mid p\in P$, $before(p,a)$ and COND(p,a)$\}$. From the very beginning, whenever a context value input, the patterns in b-COND(a) are checked and holding patterns are deleted from the set. Normally the set is empty when the activity is stated. So the when activity a is recognized to start, it is abnormal if b-COND(a) is not empty.

Similarly, we use a set d-ACCOMP(a) to record the context patterns which need to hold during the time that activity a is performed. Let d-ACCOMP(a)$=\{p \mid p\in P$, $during(p,a)$ and ACCOMPA(p,a)$\}$. This is the initial value of d-ACCOMP(a). When a is recognized to be started, the patterns in the set are checked and holding patterns are deleted. When activity a is recognized to end, it is determined to be an abnormal end if d-ACCOMP is not empty.

C. Recognizing activity and its state by state transition

The recognizing process depends on the state and state transition. The basic idea is as following.

If all the premise conditions are satisfied and accompanying context patterns start to hold, it is recognized that the activity has started. Considering there may be more than one accompanying context should hold at the beginning (*Equal* or *Start* relation) and there is no real time point that these context patterns change to hold, we use state *starting* to solve the problem. When all the premise context patterns hold and at least one accompanying pattern hold, the state is converted to *starting* from *waiting*. When the start pattern is satisfied, condition checking is performed and the state is converted to *on* or *abnormal start*.

As the activity is conducted, condition is checked when any context changes and the state is converted to *abnormal process* when any of them are not satisfied. Moreover, comparing current on pattern with last moment can determine whether the activity is interrupted. If interruption occurred, the state is converted to *suspend*.

Activity end can be judged by consequence context. If consequence context patterns occurred, the state is converted to *ending* and on pattern is no longer to be checked. If there is no consequence context pattern for an activity, activity end can be judged by premise and accompanying context pattern. If some of premise and accompanying context pattern are not satisfied, state will change to *suspend* and if the state lasts long enough, it will be converted to *ending* for the activity with no consequence context pattern, and to *abnormal process* for others.

In ending state, if *e-pattern* is satisfied, the state is converted to *end*, or else, $e\text{-}pattern(a,t)$is calculated according to the current context $c(t)$. $e\text{-}pattern(a,t)$ is compared with $e\text{-}pattern(a,t\text{-}1)$ to determine whether the activity is close to end. If more context patterns in end pattern occurred, i.e. $e\text{-}pattern(a,t\text{-}1)\subset e\text{-}pattern(a,t)$, initialize *time-elapse* context pattern to make the state lasting for longer time. If *time-elapse* is true, it means that the state lasts too long and state is converted to abnormal process.

In end state, d-ACCOMP(a) is checked and if the set is empty, the state is converted to *waiting*(for next activity). Otherwise there exists not happened context pattern that should hold during the activity process, the state is converted to *abnormal end.*

For three types of *abnormal*, the state transition rule is waiting for a certain time and converted to *waiting*, *on* respectively.

TABLE III shows the transition rules in every state.

IV. CASE STUDY

We use the approaches presented above in single-crystal X-ray diffraction experiment support. We focus on the context reason part, i.e. recognize the activity and its state. This is important in support environment, because the reaction of the support system, i.e. provide alerting and guiding information depends on the correctly recognized the situation.

The single-crystal X-ray diffraction procedure consists of three phases, in which activities are conducted in three places, such as Room 101, Room 102 and Room 103 in our example. The three phases are: selecting a crystal which is carried out in Room 101, analyzing the crystal which are conducted in Room 103 and structural determination which is accomplished in Room 103. There are eight activities in these phases and they should be performed as a certain sequence.

To perform every activity, different tools are needed. The location of user, location of tools as well as their states can be sensed and detected, which are used to determine the activity. There are 19 types of contexts related to the activities in experiment. TABLE IV is the context model of every activity.

Huang et al has designed same kind of system and analyze its effect in education [13]. However technology is not an important issue in that paper and activity recognition is simply realized by rule-based reason, only considering context at one time point.

Our work focuses on the technical aspects. We have developed an activity model taking the temporal logic into account, as in [11]. However, in [11], there are only equal and during relation, and some of abnormal situation cannot be recognized. These problems are partly solved in this paper.

V. CONCLUSION

Context interpretation in most of research is based on the assumption that high level context may result in different sensor readings or low level context, where the relation and knowledge is integrated into context model. Therefore the context interpretation is actually the hypothesis of “cause” from “consequence” which is regarded as true unless it is denied. The condition of deducing high level context is generally not sufficient. Our method is also based on this assumption and more complex because of composition of temporal relation with semantic relations. The underlying assumption of our approach is consistent with the research in the domain and consistent with intuition.

TABLE III. *STATE TRANSITION RULES*

State	**state transition rule**

Waiting
(for activity *a*)
	Input:*C*(*t*)
	IF $\forall p_i \in \{p \mid PREM(p,a)\}, hold\_at(p_i,t) \wedge \exists p_i \in \{p \mid ACCOMP(p,a)\}, hold\_at(p_i,t)$
	THEN current state is (*a*,*starting*)
	ELSE current state is (*a*,*waiting*)

starting	Input:*C*(*t*)
	IF $\forall p_i \in s-pattern(a), hold\_at(p_i,t)$
	THEN IF (*b*-COND(*a*)= ϕ) $\wedge(\forall p_i \in \{p \mid COND(p,a) \wedge \neg Before(p,a)\}, hold\_at(p_i,t))$
	THEN current state is (*a*,*on*)
	ELSE current state is (*a*, *abnormal start*)
	ELSE current state is (*a*,*waiting*)

on	Input:*C*(*t*), on pattern *o-pattern*(*a*) in last "on" moment
	IF $\exists p_j \in \{p \mid CONSEQ(p,a)\}, hold\_at(p_j,t)$
	THEN current state is (*a*,*ending*)
	ELSE IF $\exists p_j \in \{p \mid (During(a,p) \wedge COND(p,a))\}, \neg hold\_at(p_j,t)$
	THEN current state is (*a*, *abnormal process*)
	ELSE BEGIN set *o-pattern*(*a*,*t*)
	IF *o-pattern*(*a*,*t*)$\supseteq$ *o-pattern*(*a*)
	THEN current state is (*a*,*on*) *o-pattern*(*a*)= *o-pattern*(*a*,*t*)
	ELSE current state is (*a*,*suspend*)
	END

suspend	Input:*C*(*t*), on pattern *o-pattern*(*a*) in last "on" moment
	IF $\forall p_i \in o-pattern(a), hold\_at(p_i,t)$
	THEN current state is (*a*,*on*)
	ELSE current state is (*a*,*suspend*)
	
	Input: current time *t*
	IF *hold_at*(*time-elapse*,*t*)
	THEN IF $\{p \mid CONSEQ(p,a)\} = \varphi$
	current state is (*a*,*ending*)
	ELSE current state is (*a*, *abnormal process*)

ending	Input:*C*(*t*), *e-pattern*(*a*,t_{i-1})
	Set *e-pattern*(*a*,*t*)
	IF *e-pattern*(*a*,*t*)= *e-pattern*(*a*)
	THEN current state is (*a*,*end*)
	ELSE IF *e-pattern*(*a*,*t*)$\supseteq$ *e-pattern*(*a*,t_{i-1})
	Initialize *time-elapse*
	
	Input: current time *t*
	IF *hold_at*(*time-elapse*,*t*)
	THEN current state is (*a*, *abnormal process*)

end	Input:*C*(*t*)
	THEN IF (*d*-ACCOMP(*a*)= ϕ)
	THEN current state is ((next(*a*),*waiting*) // next(*a*) is next possible activity of *a*
	ELSE current state is (*a*, *abnormal end*)

abnormal start	Input: current time *t*
	IF *hold_at*(*time-elapse*,*t*)
	THEN current state is (*a*,*waiting*)

abnormal process	Input: current time *t*
	IF *hold_at*(*time-elapse*,*t*)
	THEN current state is (*a*,*on*)

abnormal end	Input: current time *t*
	IF *hold_at*(*time-elapse*,*t*)
	THEN current state is (*a*,*on*)

TABLE IV. CONTEXT MODEL OF ACTIVITY IN SINGLE-CRYSTAL X-RAY DIFFRACTION EXPERIMENT

symbol	Activity	Temporal relation	Semantic relation

a_1	Prepare Fiber	*during*(a_1, (**User_Loc**,*Room 101*))
		start((**Copperbar_Loc**,*Cuttingmat*),a_1)
		during((**Slicer**,*On*),a_1)
		overlap(a_1, (**Copperbar_Loc**,*Styrofoam*))
			PREM((**User_Loc**,*Room 101*),a_1)
			ACCOMP ((**Copperbar_Loc**,*Cuttingmat*),a_1)
			ACCOMP ((**Slicer**,*On*),a_1)
			CONSEQ(a_1, (**Copperbar_Loc**,*Styrofoam*))

a_2	Check Crystal Transparency	*during*(a_2, (**User_Loc**,*Room 101*))
		during(a_2, (**Microscope**,*On*))
		during(a_2,(**OcularlensI_Loc**,*Microscope*),)
		during((**Adjustmentscrew**,*On*), a_2)
		overlap((**OcularlensI_Loc**,*Ocularlensbox*), a_2)
			PREM((**User_Loc**,*Room 101*),a_2)
			PREM((**Microscope**,*On*),a_2)
			PREM((**OcularlensI_Loc**,*Microscope*), a_2)
			ACCOMP((**Adjustmentscrew**,*On*), a_2)
			CONSEQ((**OcularlensI_Loc**,*Ocularlensbox*) a_2)

a_3	Check Crystal Size	*during*(a_3, (**User_Loc**,*Room 101*))
		during(a_3, (**Microscope**,*On*))
		during(a_3 , (**OcularlensII_Loc**,*Microscope*))
		during((**Adjustmentscrew**,*On*), a_3)
		overlap((**OcularlensI_Loc**,*Ocularlensbox*) a_3)
			PREM((**User_Loc**,*Room101*),a_3)
			PREM((**OcularlensII_Loc**,*Microscope*) ,a_3)
			PREM((**Microscope**,*On*),a_2)
			ACCOMP((**Adjustmentscrew**,*On*), a_3)
			CONSEQ((**OcularlensII_Loc,** *Ocularlensbox*), a_3)

a_4	Place Crystal in Diffractometer	*during*(a_4,(**User_Loc**,*Room 102*))
		equal((**PCI**,*On*), a_4)
		during(a_4 , (**Temperature**,[*10*,*25*))
		during(a_4,(**Voltage**,*50KV*))
		equal((**Copperbar_Loc**,*Diffractometer*),a_4)
		during((**Controller**,*On*), a_4)
		overlap((**Crystalcenter**,*Screencenter*),a_4)
			PREM((**User_Loc**,*Room 102*),a_4)
			ACCOMP((**PCI**,*On*), a_4)
			COND((**Temperature**,[*10*,*25*) , a_4)
			COND((**Voltage**,*50KV*) , a_4)
			ACCOMP ((**Copperbar_Loc**,*Diffractometer*),a_4)
			ACCOMP((**Controller**,*On*), a_4)
			CONSEQ((**Crystalcenter**,*Screencenter*),a_4)

a_5	Diffractometer Spot Check	*during*(a_5,(**Diffractometer**,*On*))
		during(a_5,(**User_Loc**,*Room102*))
		during(a_5,(**Voltage**,*50KV*))
		equal((**Copperbar_Loc**,*Diffractometer*),a_5)
		overlap(a_5,**SpotResult_Bool**,*True*))
			PREM((**Diffractometer**,*On*),a_5)
			PREM((**User_Loc**,*Room102*),a_5)
			COND((**Voltage**,*50KV*) , a_5)
			ACCOMP ((**Copperbar_Loc**,*Diffractometer*),a_5)
			CONSEQ(**SpotResult_Bool**,*True*),a_5)

a_6	Lattice Constant Collection	*during*(a_6,(**Matrix**,*On*))
		during(a_6,(**User_Loc**,*Room102*))
		during(a_6,(***Voltage***,*20KV*))
		overlap (a_6,(**ConstantResult_Bool**,*True*))
			PREM((**Matrix**,*On*),a_6)
			PREM((**User_Loc**,*Room102*),a_6)
			COND((***Voltage***,*20KV*),a_6)
			CONSEQ((**ConstantResult_Bool**,*True*),a_6)

a_7	Crystal Lattice Regeneration and Revise	*during*(a_7,(**User_Loc**,*Room102*))
		equal((**Saint**,*On*), a_7)
		overlap(a_7,(**RevisedResult_Bool**,*True*),)
			PREM(**User_Loc**,*Room 102*),a_7)
			ACCOMP((**Saint**,*On*),a_7)
			CONSEQ((**RevisedResult_Bool**,*True*),a_7)

a_8	Crystal Structure Analyze	*during*(a_8,(**User_Loc**,*Room 103*))
		equal((**Shelxt**,*On*),a_8)
		overlap((**AnalyzeResult_Bool**,*True*),a_8)
			PREM((**User_Loc**,*Room 103*),a_8)
			ACCOMP((**Shelxt**,*On*),a_8)
			CONSEQ((**AnalyzeResult_Bool**,*True*),a_8)

The approach proposed in this paper is interval-based, considering the context change over a time period. This can recognized more situation than the method based on the context data in time point, but introduce high complexity in process. We define 6 temporal relations and 4 semantic relations; these are not enough to represent the relation in real world. The approach need to be further studied in future research work.

ACKNOWLEDGMENT

This research work was supported by the National Natural Science Foundation of China (Grant No. 70771017).

REFERENCES

[1] K. Dey Anind, D. Abowd Gregory and S. Daniel, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications", Human-Computer Interaction, vol.16, no.2, pp.97-166, 2001.

[2] W.Loke Seng, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective", The Knowledge Engineering Review vol.19, no.3, pp.213–233, 2005.

[3] D. Riboni, and C. Bettini, "OWL 2 modeling and reasoning with complex human activities," Pervasive and Mobile Computing, vol.7, no.3, pp.379-395, 2011.

[4] J. Lester, T. Choudhury, and N. Kern, "A hybrid discriminative /generative approach for modeling human activities," In Proceedings of the 19th International Joint Conference on Artificial Intelligence, Professional Book Center, Acapulco,Mexico, pp. 766-772, 2005.

[5] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational Markov networks," In Proceedings of the 19th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, pp.773-778, 2005.

[6] D.W Albrecht, and I. Zukerman, "Bayesian Models for Keyhole Plan Recognition in an Adventure Game," User Modeling and User-Adapted Interaction, vol.8, pp.5–47,1998.

[7] E.M Tapia, and S. Instille, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate

monitor," the 11th IEEE International Symposium on Wearable Computers, Boston,USA, pp. 37-40, October 2007.

[8] M.S. Ryoo, and J.K. Aggarwal, "Recognition of composite human activities through context-free grammar based representation," IEEE Conference on Computer Vision and Pattern Recognition, New York, USA, pp.1709-1718, June 2006.

[9] L. Chen, and C.D. Nugent, "Ontology-based activity recognition in intelligent pervasive environments," International Journal of Web Information Systems, vol.5, no.4, pp.410-430, 2009.

[10] L. Chen, C. Nugent, M. Mulvenna, D. Finaly, X. Hong, and M. Poland, "Using event calculus for behaviour reasoning and assistance in a smart home," in Proceedings of the 6th International Conference on Smart Homes and Health Telematics, IA, USA, vol. 5120, pp. 81–89, June 2008.

[11] T. Lu, S.K. Zhang and Q. Hao, "Activity Recognition in Ubiquitous Learning Environment," Journal of Advances in Information Technology, vol. 3,no.1, pp29-35, 2012.

[12] J.F. Allen, "Towards a general theory of action and time", Artificial Intelligence, vol.23, pp123-154,1984.

[13] G.J. Hwang, T.C. Yang, and C.C. Tsa, "A context-aware ubiquitous learning environment for conducting complex science experiments," Computers and Education, vol.53, no.2 , pp.402-413, 2009.

[14] M. Weiser, "The computer for 21st century," Scientific American, vol.261, no.30, pp.94-104, 1991.

[15] H.C. Chu, G.J. Hwang, S.X. Huang, and T.T. Wu, "A knowledge engineering approach to developing e-libraries for mobile learning," Electronic Library, vol.26, no.3,pp.303-317, 2008.

[16] C. Zhu, and W.H. Sheng, "Motion- and location-based online human daily activity recognition," Pervasive and Mobile Computing, vol.7, no.2, pp.256-269, 2011.

[17] H. Kautz, "A Formal Theory of Plan Recognition and its Implementation," Reasoning About Plans, San Francisco: Morgan Kaufmann, 1991, pp.69-125.

[18] R. Kowalski, and M. Sergot, "A logic-based calculus of events," New Generation Computing, vol.4, no.1, pp.67-95, 1986.

[19] O. Brdiczka, J.L. Crowley, and P. Reignier, "Learning situation models for providing context-aware services," Ambient Interaction 4th International Conference on Universal Access in Human-Computer Interaction, Beijing,China, pp.23-32, July 2007.

[20] T. Springer and A.Y. Turhan, "Employing Description Logics in Ambient Intelligence for Modeling and Reasoning about Complex Situations," Journal of Ambient Intelligence and Smart Environments, vol.1, no.3, pp.235-259, 2009.

[21] D.Q. Zhang, M.Y. Guo and J.Y. Zhou, "Context Reasoning Using Extended Evidence Theory in Pervasive Computing Environments," Future Generation Computer Systems, vol.26, no.2, pp.207-216,2010.

[22] A. Mannini and A.M. Sabatini, "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers," Sensors, vol.10, no.2, pp.1154-1175, 2010.

[23] T. Gu, Z. Wu, X.P. Tao and H.K Pung, "An Emerging Patterns based Approach to Sequential, Interleaved and Concurrent Activity Recognition," In Proceeding of the 7th Annual IEEE International Conference on Pervasive Computing and Communications, Galveston, TX, pp.1-9, March 2009.

AUTHORS PROFILE

Tao Lu

She is currently an associate professor in School of Management Science and Engineering, Dalian University of Technology, China. Her main research interest is information system, knowledge-based system and pervasive computing. Email: lutao@dlut.edu.cn

Xiaoling Liu

She is currently a graduate in School of Management Science and Engineering, Dalian University of Technology, China. Her main research interest is pervasive computing. Email: liuxiaoling@mail.dlut.edu.cn