Abstract—A context-aware system generally integrates many different kinds of information techniques to build a computing environment for serving users anytime, anywhere. For a complicated context-aware system with large number of users, the interference among devices and users usually decreases the correctness and reliability of context. The incorrect context may cause context inconsistency and lead a context-aware system to trigger abnormal reactions and wrong decisions. In this paper, the prevention strategy is proposed for resolving context inconsistency. The context consistency constraints for different sensor data types are proposed and developed to prevent the context inconsistency caused by imprecise sensing devices and environmental noises. The experimental results show that the proposed consistency constraints and resolution rules can eliminate incorrect contexts effectively and improve the quality of a context-aware system in the application of smart living space.

Keywords—context aware; context inconsistency; inconsistency prevention; smart home;

I. INTRODUCTION


Though context-aware computing has been studied over a decade, most of the research issues focused on system frameworks, context representations, context reasoning and security of application services. Many prototypes and tools were proposed and developed. Some practical systems have also been built in many applications recently. However, the performance and reliability of context-aware systems are getting low while system components and operating environments are getting complicated.

To improve the effectiveness of a context-aware system, recognizing context correctly is an important task. Since context-aware systems may cause improper reactions due to the incorrect sensing context, the low qualified context will decrease the effectiveness of a context-aware system directly. Generally, incorrect contexts may cause two main problems: context inconsistency and context conflict. If a contradiction among the contexts exists while computing services in a context-aware system, the situation is called context inconsistency. If the resources can not satisfy different context services at the same time, it is called context conflict. In the previous researches, most of the solutions used the strategy of inconsistency detection and resolution for the context inconsistency problem. However, it is difficult to identify all problematic contexts completely using consistency constraints. The inconsistency detection is also an extra overhead for the system.

In this paper, the prevention strategy is proposed and discussed for the context inconsistency problem. Instead of detecting and resolving inconsistency cases after context inconsistency occurring, the prevention strategy involves specific constraints and rules for each sensor context. The context consistency constraints for different sensor data types are proposed and developed to prevent the possible context inconsistency caused by imprecise sensing devices and environmental noises. The proposed approaches are tested on two ADL (Activities of daily living) datasets. The experimental results show that the proposed consistency constraints and resolution rules can eliminate incorrect contexts effectively and improve the quality of a context-aware system in the application of smart living space.

The remainder of this paper is organized as follows. Section 2 reviews the previous technologies of context inconsistency detection and resolution. Section 3 depicts the idea of prevention strategy and proposes the consistency constraints and prevention rules for different sensor data types. The experimental results are shown in Section 4. Finally, we give a conclusion for this work in Section 5.

II. RELATED WORK

The problem of context inconsistency was firstly mentioned by Henricksen et al in 2004. In [8], the authors analyzed and summarized several reasons of mismatching between sensed contexts and real-world activities. The representation method of describing imprecise context is proposed in their article, but the resolution for imprecise context was not discussed.

Context inconsistency is possibly caused by either imprecise, incorrect context, or the contradiction of context definitions. The related research and solution strategy for
context inconsistency generally contains two steps: inconsistency detection and resolution. The system has to resolve the problem of context inconsistency, Xu et al. [12] defined consistency constraints to detect context inconsistency existing in the system. Bu et al. [7] used ontology to describe the context inconsistency detection rules for representing and detecting context inconsistency. This method applies the relationships between two classes including transitivity, inclusion, and exclusion to construct detection rules. The detection rules are then used to detect if the instances of context in ontology violate the rules of relationships. The merit of this approach is that the rules are able to be inferred easily. However, it is hard to represent complicated consistency constraints by relationships of ontology.

In 2008, Xu [11] proposed a declarative constraint language to represent consistency constraints and detect context inconsistency. The proposed constraint language is based on first-order logic. Further, it supports user-defined functions. This approach can represent complicated rules easily, but it lacks of unified description and representation for used-defined functions.

For the resolution of context inconsistency, Xu et al. [12] proposed a solution based on database systems. The traditional resolution of inconsistency in a database system considers two operations to perform: Accept or Reject. The operation Accept is to accept the last context and delete the previous old context causing inconsistency. The operation Reject is to reject the last context causing inconsistency and preserve the previous context for maintaining the consistency.

Using the heuristic approaches to resolve context inconsistency has been proposed and studied by Bu et al. [7] and Xu et al. [13]. The difference between the heuristic and the database approaches is the information used for resolution. Before determining which context should be deleted while inconsistency occurs, the heuristic resolution considers more data, e.g. sampling time, frequency of context, and context data types. However, the heuristic approach cannot guarantee to delete the inconsistent context all the time.

Mayrhofer [9] proposed the division of reactions for a context-aware system: Reactive and Proactive. The Reactive reaction will be triggered according to the sensing context and the situation of environments. The Proactive reaction will respond proper context actively to avoid inconsistency by predicting the attention and changing of users.

III. CONTEXT INCONSISTENCY PREVENTION

In this paper, we propose the prevention strategies for context inconsistency instead of inconsistency detection and resolution. We propose five different prevention approaches for context inconsistency according to different kinds of sensor data types: binary data, numerical data and nominal data.

A. Binary data type

Three context inconsistency prevention constraints are proposed for binary sensor data including the range constraint (Ĉₘ), the least persistence constraint (Ĉₚ), and the change-point constraint (Ĉₕ).

1) The range constraint Ĉₘ(tₘᵢₙ, tₘₐₓ):

This constraint contains two parameters: tₘᵢₙ and tₘₐₓ. tₘᵢₙ is the minimal sensing time and tₘₐₓ is the maximal sensing time of the context. The consistency prevention rule for the constraint Ĉₘ(tₘᵢₙ, tₘₐₓ) is described as

\[
\begin{align*}
\text{IF} \quad & [t - \text{start time}(x_{\text{signal}} = 1)] \leq t_{\text{min}} \\
\text{THEN} \quad & x_{\text{context}}(t) = \text{ON} \\
\text{IF} \quad & [t - \text{start time}(x_{\text{signal}} = 1)] > t_{\text{max}} \\
\text{THEN} \quad & x_{\text{context}}(t) = \text{OFF}
\end{align*}
\]

where x is a sensor, x_{signal} and x_{context}(t) represent the raw signal’s value and the context for the sensor x at current sampling time t, respectively. The function start time() returns the start sampling time for signals or context of the sensor x.

The prevention rule of the Ĉₘ constraint can be used in fixing the context of sensors with regular responding time. While such a sensor’s signal is triggered, the context should be turned “ON” for the time tₘᵢₙ minimally and tₘₐₓ maximally. An instance of applying the consistency prevention rule for the range constraint Ĉₘ is shown as Example 1.

Example 1. For the range constraint Ĉₘ(tₘᵢₙ, tₘₐₓ), assume that the parameters are tₘᵢₙ = 5 and tₘₐₓ = 6. The example in Fig. 1 shows that the raw sensor signal is triggered between the durations of the 5th second and the 14th second, and the 18th second and the 20th second. The first duration of “ON” signal keeps 9 seconds. It violates the maximal time of constraint Ĉₘ(5, 6). The prevention rule will fix the sensed context to be “ON” for maximally 6 seconds. On the other hand, the second duration of “ON” signal keeps only two seconds. It also violates the minimal time of constraint Ĉₘ(5, 6). The fixed context will be kept “ON” by the prevention rule for 5 seconds minimally from the time of the context being triggered.

![Figure 1. Example for the range constraint Ĉₘ.](image)

(2) The least persistence constraint Ĉₚ(tₚₜₜₚₜ): The least persistence constraint Ĉₚ contains only one parameter tₚₜₜₚₜ which is the least persistent time for the signals having to be sensed as “1” before the sensor context...
“ON” is triggered. The consistency prevention rule for the least persistence constraint $\hat{C}_{LP}(t_{\text{least}})$ is described as

$$\begin{align*}
&\text{IF } [ t - \text{start time}(x_{\text{signal}} = 1) ] \geq t_{\text{least}} \\
&\text{THEN } x_{\text{context}}(t) = \text{ON} ; \\
&\text{ELSE } x_{\text{context}}(t) = \text{OFF} ;
\end{align*}$$

The context is kept “OFF” if the raw signal was not sensed as “1” for $t_{\text{least}}$ time continuously. Example 2 shows an instance of executing the consistency prevention rule for the least persistence constraint $\hat{C}_{LP}$.

**Example 2.** Let the parameter of constraint $\hat{C}_{LP}(t_{\text{least}})$, $t_{\text{least}}$ be set to be 5 seconds. As shown in Fig. 2, the raw sensor signal is triggered between the durations of the 5th and the 14th seconds, and the 18th and the 20th seconds. The least persistence constraint $\hat{C}_{LP}(5)$ will first check if the sensor signal is persisted 5 seconds at least before the sensor context being triggered. The first duration of “ON” signal keeps 9 seconds. It satisfies the constraint $\hat{C}_{LP}(5)$, the prevention rule thus triggers the context to be “ON” from the 10th second. On the other hand, the second duration of “ON” signal keeps only two seconds. It does not satisfy the constraint $\hat{C}_{LP}(5)$, the context will not be triggered.

![Figure 2. Example for the least persistence constraint $\hat{C}_{LP}$.](image)

(3) The change-point constraint $\hat{C}_{CP}$:

The change-point constraint $\hat{C}_{CP}$ referred to the state correction method proposed by Kasteren et al. in 2008 [16]. Here, we modified the method to be the consistency prevention rule for the change-point constraint $\hat{C}_{CP}$. The rule is described as follows.

$$\begin{align*}
&\text{IF } [ x_{\text{signal}}(t) \text{ XOR } x_{\text{signal}}(t-1) ] = 1 \\
&\text{THEN } x_{\text{context}}(t) = \text{ON} ; \\
&\text{ELSE } x_{\text{context}}(t) = \text{OFF} ;
\end{align*}$$

where $x_{\text{signal}}(t)$ is the raw signal’s value at current sampling time $t$ for the sensor $x$. The context of sensor $x$ is only set to “ON” while the signal is changing at the sampling point. Example 3 shows the result of executing the consistency prevention rule for the change-point constraint $\hat{C}_{CP}$.

**Example 3.** As shown in Fig. 3, the raw signal changes its state at the time point of the 5th, 14th, 18th, and the 20th second. The change-point constraint $\hat{C}_{CP}$ will trigger the context to be “ON” until the next sampling time point. The sampling points without value change keeping “OFF.”

![Figure 3. Example of Change-Point constraint.](image)

**B. Binary data type**

For numerical sensor data, we propose a context inconsistency prevention constraint called the bound constraint and denoted as $\hat{C}_{B}$. The bound constraint $\hat{C}_{B}(v_{\text{max}})$ contains one parameter $v_{\text{max}}$ which is the maximal variation of signal values. The rule of consistency prevention for the constraint $\hat{C}_{B}(v_{\text{max}})$ is described as

$$\begin{align*}
&\text{IF } [ x_{\text{signal}}(t) - x_{\text{context}}(t-1) ] > v_{\text{max}} \\
&\text{THEN } x_{\text{context}}(t) = x_{\text{signal}}(t-1) + v_{\text{max}} ; \\
&\text{ELSE } x_{\text{context}}(t) = x_{\text{signal}}(t) - v_{\text{max}} ;
\end{align*}$$

Since the context returns a value with numerical data type, the sensor’s signal exceeding the maximal variation $v_{\text{max}}$ is not allowed. The continuous sampling signals violates the bound constraint $\hat{C}_{B}(v_{\text{max}})$ will be restricted by the rule of consistency prevention. An example in Example 4 shows the correcting of the consistency prevention rule for the bound constraint $\hat{C}_{B}$.

![Figure 4. Example of the bound constraint.](image)

**Example 4.** Let the parameter of constraint $\hat{C}_{B}(v_{\text{max}})$, $v_{\text{max}}$ be set to be 10. As shown in Fig. 4, the raw sensor signal value sequence is <29, 38, 13, 20, 45, 42>. The values at the second sampling point and the third sampling point are 38 and 13, respectively. The difference between the two neighbor values is larger than $v_{\text{max}}=10$. The value of the third sampling point will be fixed into 28. The fifth sampling point is another value in this example needed to be fixed. Since the value at the fourth sampling point is 20, the value at the fifth sampling point should be corrected to be
30 instead of 45. However, after the fifth sampling point being fixed, the value at sixth sampling point needs to be adjusted, too. The consequence of the value sequence after applying the bound constraint is \(<29, 38, 28, 20, 30, 40>\).

C. Binary data type

For nominal sensor data, we propose a context inconsistency prevention constraint named the Class Constraint, \(\hat{C}_C\). Before applying the class constraint \(\hat{C}_C\), a weighted transition graph \(G\) has to be built by training the historical signal sequence of an activity. As the example of sequence “ababbbbcdedea”, “a” is the first signal and “b” is the second signal. We thus get an edge of \((a, b)\) in the transition graph. Since two pairs of \((a, b)\) of the transition graph is the second signal. We thus get an edge of \((a, b)\) in the transition graph. Since two pairs of \((a, b)\) can be found in the sequence, the weight of the edge \((a, b)\) is 2. The weights of the transition graph \(G\) for the sequence is shown as Fig. 5. The rule of consistency prevention for the class constraint \(\hat{C}_C\) is described as

\[
\text{IF } \{(x_{\text{context}}(t-1), x_{\text{signal}}(t)) \notin G\} \\
\text{THEN } x_{\text{context}}(t) = C \text{ with max } \{(x_{\text{signal}}(t-1), C) \in G\}; \text{ ELSE } x_{\text{context}}(t) = x_{\text{signal}}(t);
\]

where \(\text{max}\{x, y\} \in G\) represents the edge \((x, y)\) with the maximal weight in the weighted transition graph \(G\). \(x\) and \(y\) are the context of signals. Example 5 shows an example of executing the consistency prevention rule for the class constraint \(\hat{C}_C\).

![Weighted transition graph for the sequence “ababbbbcdedea.”](image)

**Example 5.** We assume that the weighted transition graph in Fig. 5 is constructed by the sequence “ababbbbcdedea.” Let the sequential data “ababbbbcdedea” be sensed by the sensor. The correction is shown as Table I. For the raw signal sequence, the first edge pair \((a, b)\) exists in the graph. The second edge pair \((b, d)\) is not in the transition graph. Hence, the third signal “d” is corrected and replaced as “b.” After the correcting, the next edge pair starts from the next raw signal, \((d, c)\), after the fixed context. Since the edge \((d, c)\) does not exist in the transition graph, the “e” in the edge \((d, e)\) is used to replace “c.” Then, we are going to check the next edge pair \((c, d)\). The context sequence after the correcting of the rule becomes “abbdced.”

<table>
<thead>
<tr>
<th>Raw</th>
<th>a</th>
<th>b</th>
<th>d</th>
<th>d</th>
<th>c</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>a</td>
<td>b</td>
<td>b</td>
<td>d</td>
<td>e</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

**IV. EVALUATIONS**

The evaluations used a synthesized dataset to show the performance of the proposed strategy. The synthesized dataset is generated by the Synthetic Data Generator [13]. The generator simulates a living space equipped with 10 sensors. The data record the values of sensors and annotate the corresponding activities. Totally 7 activities are annotated. The sampling rate is one record per 60 seconds. There are 28 days recorded data in the dataset. The original sampling time for the dataset SD is 60 seconds. In this experiment, we set the sampling time to one second and randomly generated noises for the new sampling data with 0% to 95% of possibility in interval of 5%. Thus, totally 20 test datasets were generated.

To evaluate the performance of the proposed prevention methods, the dataset was split into 27 days data for training and 1 day data for testing by cross-validation. The evaluation model is Hidden Markov model (HMM) [10]. After the model was built by training data, the testing data were evaluated by two measures: the time slice accuracy \((Acc_T)\) and the class accuracy \((Acc_C)\). The two measures are defined as

\[
Acc_T = \frac{\sum_{t=1}^{N} |\text{HMM}(t) = L(t)|}{N}, \text{ and } Acc_C = \frac{1}{C} \sum_{c=1}^{C} \frac{\sum_{t=1}^{N} |\text{HMM}(n) = L(n)|}{N_c},
\]

where \(N\) is the total number of sampling time slices. \(N_c\) is the number of sampling time slices belonging to the labeled activity \(c\) and \(C\) is the number of possible activities. HMM(t) is the inferred activity and \(L(t)\) is the labeled real activity in the dataset at sampling time \(t\). \(|\text{HMM}(t) = L(t)|\) is the number of activities which HMM(t) and \(L(t)\) is identical at sampling time \(t\).

For the parameters of the range constraint \(\hat{C}_R(t_{\text{min}}, t_{\text{max}})\), \(t_{\text{min}}\) is set to one second, and \(t_{\text{max}}\) is tested from one second to five seconds. The parameter of the least persistence constraint \(\hat{C}_{LP}(t_{\text{least}})\), \(t_{\text{least}}\) is tested from one second to five seconds, too. The results of testing are from Fig. 6 to Fig. 11. Fig. 6 and Fig. 7 are the time slice accuracy and the class accuracy of \(\hat{C}_R\), respectively. Since the original sampling time is much larger than noises, the time intervals of constrain \((t_{\text{min}}, t_{\text{max}})\) obviously have no effect on the accuracy. The constraint even causes noise-like result. Fig. 8 and Fig. 9 show the time slice accuracy and the class accuracy of \(\hat{C}_{LP}\), respectively. Since the \(\hat{C}_{LP}\) constraint returns context if signals maintain for \(t_{\text{least}}\) time, it can filter...
out the noise effectively when the noise rate are high and the time $t_{\text{data}}$ is larger. Fig. 10 and Fig. 11 show the time slice accuracy and the class accuracy of $\hat{C}_{CP}$, respectively. This constraint is not suitable for the case of large sampling time and unstable signals.

V. CONCLUSION

In this paper, we initiate the prevention strategy for solving the context inconsistency problem. The context consistency constraints for different sensor data types include binary, numerical, and nominal data types. Five rules of consistency constraints, $\hat{C}_R$, $\hat{C}_LP$, $\hat{C}_CP$, $\hat{C}_B$, and $\hat{C}_C$, are proposed and developed to prevent the possible context inconsistency caused by imprecise sensing devices and environmental noises. The experimental results of the proposed approaches are show that the proposed consistency constraints and resolution rules can eliminate incorrect contexts effectively if a proper consistency constraint is applied to the corresponding sensor.

ACKNOWLEDGMENT

This research was supported in part by National Science Council of Taiwan, R. O. C. under contract NSC 99-2221-E-024-015.

REFERENCES


![Figure 6](image_url)  
Figure 6. Time-slice accuracy for $\hat{C}_R(1,1)$ to $\hat{C}_R(1,5)$.

![Figure 7](image_url)  
Figure 7. Class accuracy for $\hat{C}_C(1,1)$ to $\hat{C}_C(1,5)$. 
Figure 8. Time-slice accuracy for $\hat{C}_{LP}(1)$ to $\hat{C}_{LP}(5)$.

Figure 9. Class accuracy for $\hat{C}_{LP}(1)$ to $\hat{C}_{LP}(5)$.

Figure 10. Time-slice accuracy for $\hat{C}_{CP}$.

Figure 11. Class accuracy for $\hat{C}_{CP}$. 