International Journal of Smart Home
Vol. 9, No. 4 (2015), pp. 195-202
http://dx.doi.org/10.14257/ijsh.2015.9.4.20

# Development of the Rule-Based Inference Engine for the Advanced Context-Awareness

Jun-Hwan Jang[1] and *Sung Hyun Yang*[2]

[1]*Smart H & B Laboratory, Kwangwoon University, Seoul, Korea*
[2]*Smart H & B Laboratory, Kwangwoon University, Seoul, Korea*
[1]*gganjang@kw.ac.kr*, [2]*shyang@kw.ac.kr*

***Abstract***

*Smart home services for users require precise and reliable context information to guarantee credibility. To do this, system should have ability of context-awareness. Most of the context-aware systems have adopted rule-based inference but it did not fulfill users' demand. To support valuable services in smart home environment, we present the rule-based inference engine which uses user profiles to let the system be aware of users' context much accurately. Since the physical and logical size of the domain is relatively small, we also introduce inference algorithm optimized for limited size of memory space of the system.*

***Keywords:*** *Ubiquitous, Inference, Smart Home, Context-Awareness, User Preference*

## 1. Introduction

To provide various services to users in smart home environment, several steps such as gathering right sensing value, reasoning precise contexts of entities, and deploying the most appropriate services must proceed. Most of the works which have been done focuses on the context-awareness and services, but they were not able to recognize personalized user contexts unless rules are defined for each individual. Furthermore, previous inference algorithms which have high-speed reasoning performance require relatively plenty of memory spaces. In this paper, we introduce the inference algorithm which is optimized for the embedded system, user profile that contains user's preference for each context, and their convergence for the advanced context-awareness.

## 2. User profile

Previous systems for smart home services dealt with how the services are provided. For example, if the conditions like temperature, user location, and time are satisfied, the system used to fire service directly, and most of the smart home systems calibrated those services to satisfy user. As each user prefers different environment conditions, it looks reasonable. However, the system must recognize user's context before deploying services as mentioned above, or it loses credibility. It is because no matter how the service is great, there is a chance that user does not want the service at that time.

To obtain more precise context, we created user profile which contains user's preference for each context and service according to the profile management methods from [1].

ISSN: 1975-4094 IJSH

```
<?xml version="1.0" encoding="UTF-8" ?>
  <ca:user_profile xmlns:ca="http://www.example.org/userprofileSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.example.org/userprofileSchema userprofileSchema.xsd">
        <ca:user_identity>junhwanjang</ca:user_identity>
        <ca:user_context>
                <ca:sleeping>
                        <ca:illuminance>dim</ca:illuminance>
                </ca:sleeping>
                <ca:mealtime>
                        <ca:breakfast>breakfast</ca:breakfast>
                        <ca:lunch>lunch</ca:lunch>
                        <ca:dinner>dinner</ca:dinner>
                </ca:mealtime>
         </ca:user_context>
         <ca:user_service>
                <ca:watchingTV>
                        <ca:dawn>1</ca:dawn>
                        <ca:morning>2</ca:morning>
                        <ca:breakfast>3</ca:breakfast>
                        <ca:lunch>4</ca:lunch>
                        <ca:afternoon>5</ca:afternoon>
                        <ca:evening>6</ca:evening>
                        <ca:dinner>7</ca:dinner>
                        <ca:night>8</ca:night>
                </ca:watchingTV>
                <ca:washing>
                        <ca:watertemp>warm</ca:watertemp>
                </ca:washing>
         </ca:user_service>
  </ca:user_profile>
```

**Figure 1. A Partial Structure of User Profile Schema**

The user profile is built up by using profile schema and written in XML for the purpose of easy modification. Each user who resides in the domain should have whose own profile for the better context-awareness. They are loaded while the inference engine performs reasoning and affect the result of inference.

### 2.1. Context Preference

Context preference is a key feature of the user profile. It stores the information of preferences that users like for each context. For example, some users might want to sleep in dark environment and the others could prefer the lighter room illuminance to sleep well. Context preference is made to catch these kinds of different favors of individuals.

The elements or conditions where users can reflect their preference are determined by their unique characteristic. They must have a spectrum in its range so that users have more than two options to choose which invokes necessity of recognizing user's favors. For instance, environment information like illuminance has a wide range of spectrum to be selected as preference. On the other hands, binary information like *ON* and *OFF* is not helpful for the most of time because for example, the users must turn on the TV to watch something.

Once the conditions are defined, it is necessary to determine how the system is going to understand user's preference. Generally, the users do not express their preference in numeric value like 26.5 degree Celsius. They describe what they feel in linguistic expression such as cold, mild, and warm. To let the users and the system communicate well each other on ambiguous expressions, we adopted membership functions from Fuzzy Theory [2].

 

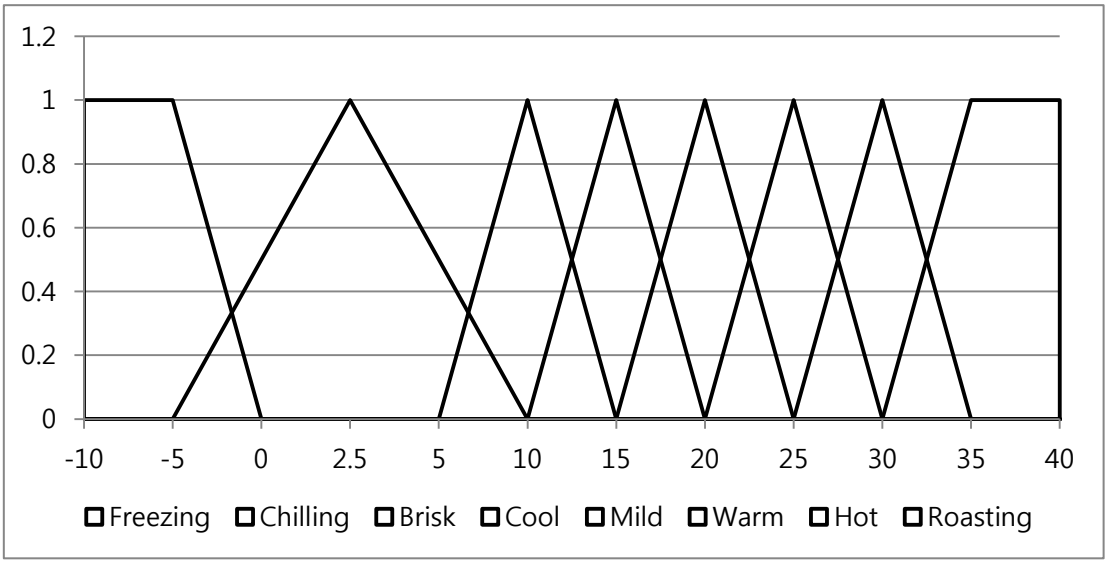


**Figure 2. Membership Functions for Temperature**

The membership functions shown above are fixed as defined [3]. If the user's preference is changed, then it switches to other function rather than re-defining functions because the system needs a static reference to compare with other users' preference.

### 2.2. Service Preference

Service preference contains the information that how the user wants to be served. It is used by service application layer rather than middleware or context provider. If a user is trying to take a shower, the system must control the water temperature which the user prefers. For now, only services related to the contexts such as *watchingTV* and *washing* are defined yet since the service application layer is still being developed.

## 3. Inference engine

Our inference engine is called Context Acquisition and located in Context Provider which contains Knowledge Base and interfaces to other layers. Context Acquisition consists of sub-modules such as Working Memory, Inference Module, Profile Manager, Context Inspector, Fuzzyfier, Low-level Context Composer, and Ontology Provider.

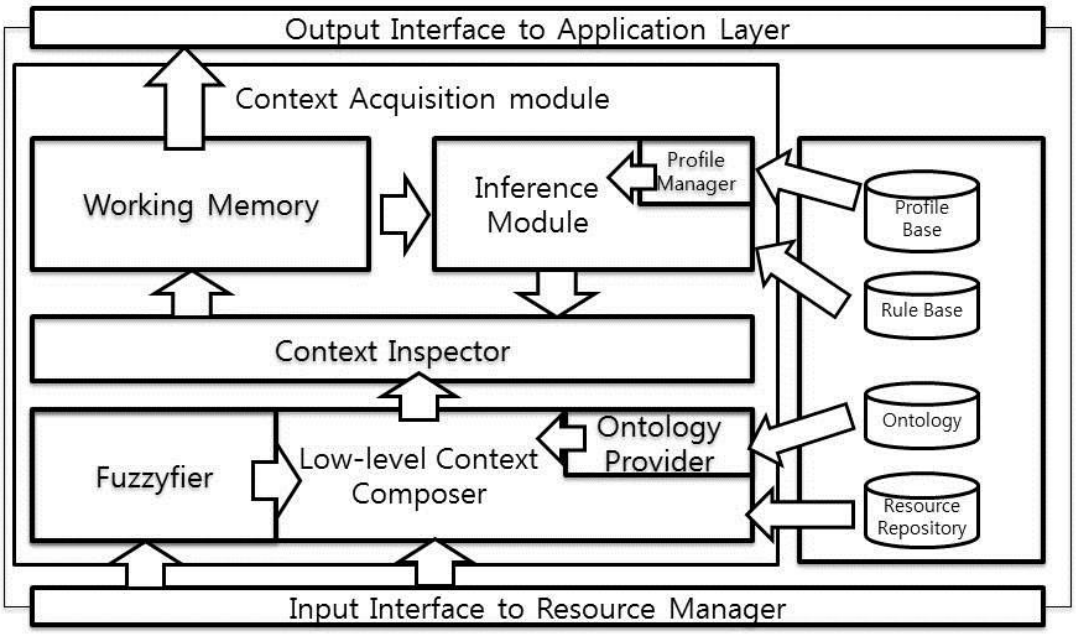


**Figure 3. Block Diagram of Context Acquisition in Context Provider**

Working Memory is a memory space to hold current contexts in the domain. Inference Module is where the actual reasoning takes place with rule-matching, profile adjustment, and verification. Profile Manager is to load user profiles and provide the information of user preference. Context Inspector literally inspects contexts to check if they are duplicate or invalid before adding contexts to Working Memory. Fuzzyfier performs fuzzification on sensor value, and Low-level Context Composer creates low-level contexts. Ontology provider supports ontological information like relations between entities.

### 3.1. Inference Algorithm

Traditional inference engines played with working memory to find all valid matches since the rule-based inference engines should deal with the known facts. It inevitably caused desperate running time as the size of working memory increases. RETE is one of the famous rule-matching algorithms to solve this problem [4, 5]. It reduced the inference time as creating extra networks for any change in working memory, but it also has a disadvantage that requiring lots of memory space to keep networks.

Our inference engine adopted forward chaining algorithm to infer context and backward chaining to verify the contexts inferred. In a small domain like home, it was theoretically obvious that running time for inference won't take so long, and the platform where the inference engine is installed has a limited hardware specification.

Therefore, we decided to user Brute Force Algorithm and sacrifice running time rather than memory space.

Not like other traditional inference engine, the inference process does not occur in working memory. It accesses to the Working Memory only when finding matches linearly. The inference proceeds by substituting variables in the current rule being matched with instances in Working Memory. If there is a variable with no instance, the engine immediately returns false so that the running time can be optimized. If any instance is found for the current variable, then it moves to next variable for substitution. Once all variables are substituted, the engine starts to look up the Working Memory to see if all

 

conditions are satisfied. If no matches are found for the current combination, then the inference engine will try another combination.

These processes are run recursively so that all possible outcomes can be examined, and the memory space can be saved. In addition, we cannot simply use a lot of nested loops like poor Brute Force Algorithm because it is not flexible enough to respond to dynamic number of variables for each rule and instances of each variable. The concept of making possible combinations of contexts is pretty similar with traveling binary search tree. It starts from the bottom left and ends at the bottom right [6].

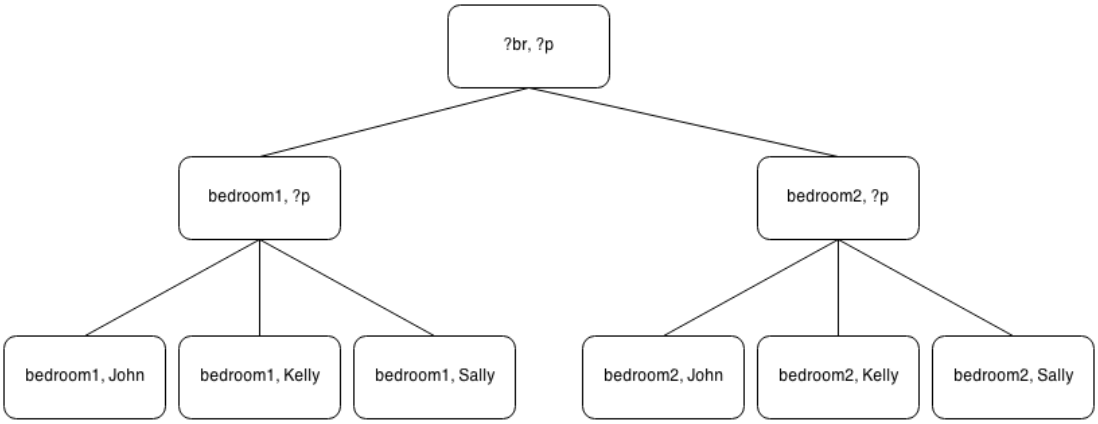


**Figure 4. Possible Outcomes of Instance Combinations for 'Sleep' Context in Case there are 2 Bedrooms and 3 Persons in the Smart Home Domain**

### 3.2. Profile-applied Inference

The inference engine which has been demonstrated so far does not have a critical improvement compared to the previous inference engines as the user profile is not applied yet.

The user profile is applied when all variables are substituted, and if there is any user who is related to the context, a corresponding profile is loaded and substitutes the value of context. For example, if someone cannot sleep in completely dark room and prefer the lighter illuminance, his/her lighting preference will be like dim or gloomy. Since the profile has that information, it replaces the value in a rule which is dark to his/her preference and finds matches.

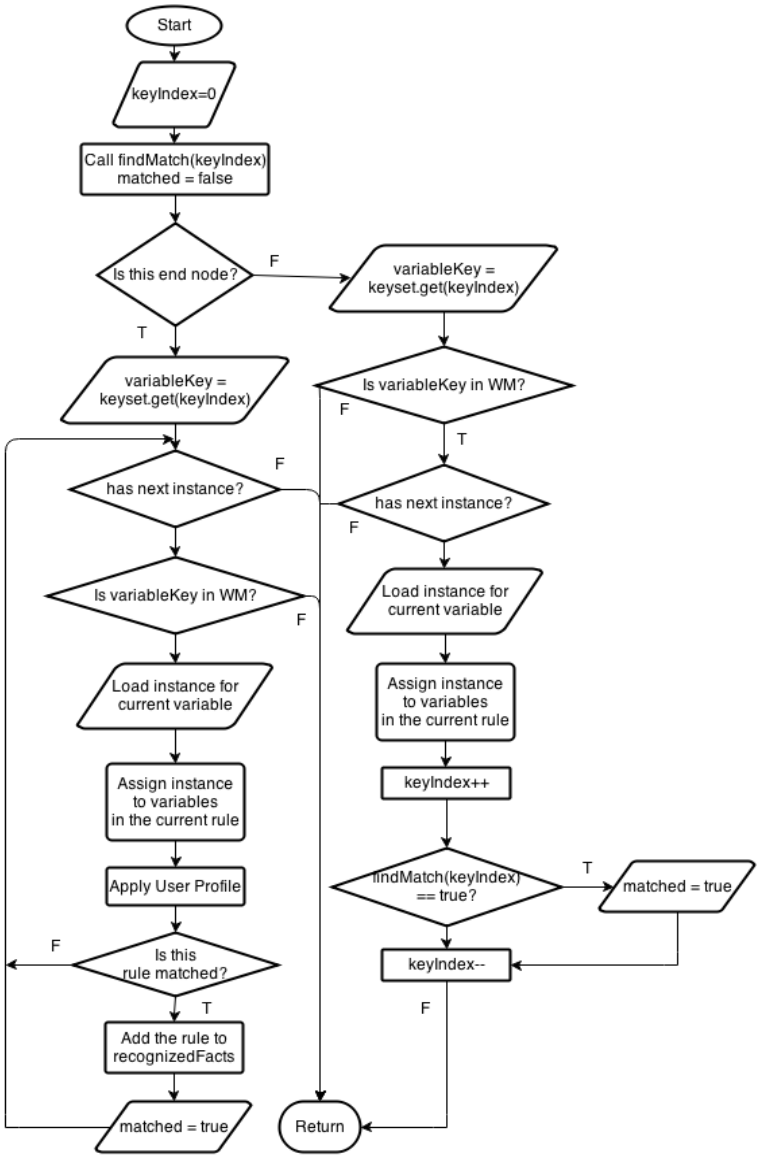


**Figure 5. A Flowchart of the Entire Inference Procedures**

After the entire matching process is done, a function named Status Validator is called to check if user's state is valid. If any condition for user's current context is failed, the user context will be changed to 'none' state to describe no context is applied for user.

## 4. Conclusion

In this paper, we categorized steps to provide smart home services in smart home environment and introduced how more precise contexts can be obtained. To gain user's credibility on the smart home system, it is essential that correct services must be given on the right time regardless of the quality of service.

We could save more memory space with the newly developed inference algorithm because it is only proportional to the number of types of variables in a rule. The user profile made it possible to infer personalized contexts applying user's preference without writing many lines of rules for each individual. It also gave the flexibility in building and modifying rules to not even system developers but users.

 

As more features that user's preference can apply are found, recognizing user's context will be easier and more reliable than now. To apply our inference engine to the bigger domains like buildings, it is necessary that the inference algorithm should be improved in the aspect of running time. Furthermore, it is true that the user profile is fixed as created. To reflect user's preferences which change over time much correctly, the user profile must be updated periodically. We consider machine learning as a promising methodology for it.

## Acknowledgement

This work was supported by Industrial Strategic technology development program, 10041788, Development of Smart Home Service based on Advanced Context-Awareness funded by the Ministry of Trade, industry & Energy (MI, Korea)

## References

[1] J. Groppe, Paderborn Univ., Mueller, W. Profile Management Technology for Smart Customizations in Private Home Applications. Database and Export Systems Applications, 2005. Proceedings. Sixteenth International Workshop on, pp. 226 – 230, **(2005)** August 26-26; Copenhagen, Denmark.
[2] L. A. Zadeh, Fuzzy Sets. Inform. Control, 8, pp.338 – 353 **(1965)**.
[3] T. P. Hong, C. Y. Lee, Induction of fuzzy rules and membership functions from training examples. Fuzzy sets and Systems 84, 1, pp. 33 – 47 **(1996)**.
[4] C. L. Forgy, Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial intelligence, 19, 1, pp. 17-37 **(1982)**.
[5] D. P. Miranker, Treat: A better match algorithm for ai production systems; long version. University of Texas at Austin **(1987)**.
[6] R. Gilberg, B. Forouzan, Data Structures: A pseudocode Approach With C++, Pacific Grove, CA **(2001)**

## Authors

**Jun-Hwan Jang**, received his B.S. in Computer Science from Northwest Missouri State University, Missouri, USA, in 2012 and M.S. degree in Electronic Engineering from Kwangwoon University, Seoul, Republic of Korea, in 2015. He works in Wisenut, Republic of Korea as a researcher. His main research interests are Context-Awareness, Natural Language Processing and Big Data.

**Sung Hyun Yang**, received his B.S. and M.S. degree in Electrical Engineering from Kwangwoon University, Seoul, Republic of Korea, in 1983 and 1987 respectively. He completed his Ph.D. from Kwangwoon University in 1993. He is a Professor in Electronic Engineering at Kwangwoon University, Seoul, Republic of Korea. He is a Director of the Ubiquitous Home Network Center, Kwangwoon University. He was a Research Scientist at Boston University from 1996 to 1998. He was Chairman of the Home Network Market Activation Section, Korean Association for Smart Home from 2007 to 2008. His main research interests are Digital logic, Embedded Systems, M2M, Next Generation Ubiquitous Home Networks, and Context-aware systems.