Incomplete and Uncertain Data Handling in Context-Aware Rule-Based Systems with Modified Certainty Factors Algebra*

Szymon Bobek and Grzegorz J. Nalepa

AGH University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
{szymon.bobek,gjn}@agh.edu.pl

Abstract. Context-aware systems make use of contextual information to adapt their functionality to current environment state, or user needs and habits. One of the major problems concerning them is the fact, that there is no warranty that the contextual information will be available, nor certain at the time when the reasoning should be performed. This may be due to measurement errors, sensor inaccuracy, or semantic ambiguities of modeled concepts. Several approaches were developed to solve uncertainty in context knowledge bases, including probabilistic reasoning, fuzzy logic, or certainty factors. However, handling uncertainties in highly dynamic, mobile environments still requires more consideration. In this paper we perform comparison of application of different uncertainty modeling approaches to mobile context-aware environments. We also present an exemplary solution based on modified certainty factors algebra and logic-based knowledge representation for solving uncertainties caused by the imprecision of context-providers.

Keywords: context-awareness, mobile devices, knowledge management, uncertainty.

1 Introduction

Context-aware systems aim to make use of context information to allow devices or applications to behave in a context-aware, thus "intelligent" way. The variety of sensors available on mobile devices, and almost unbounded access to the Internet, allows for building more advanced reliable context-aware systems. However, many context-aware systems are based on the assumption that the information they require is always available and certain. In mobile environments these assumption almost never hold.

Contextual data can be delivered to the mobile context-aware system in several different ways: directly from the device sensors [13], from other devices sensors, over peer-to-peer communication channels [2,11], from external data sources like contextual servers [6], from reasoning engines that based on the low-level context and a contextual-model, provide higher-lever context [17]. In each of this cases, the system may experience problems caused by the uncertain contextual information.

* The paper is supported by the AGH UST Grant.

A. Bikakis et al. (Eds.): RuleML 2014, LNCS 8620, pp. 157–167, 2014.

Although there are many solutions for uncertainty handling in knowledge bases, there is still little research in the field of mobile context-aware systems. The mobile environment is highly dynamic which requires from the uncertainty handling mechanism to adjust to rapidly changing condition. Probabilistic and machine learning approaches cope very well with most common uncertainties types, but they need time to learn an re-learn. What is more, they use a model that is not understandable for the user, and therefore it cannot be modified by him or her. Fuzzy logic approaches can be used to model uncertainty in an understandable form, but they mainly cope with uncertainty caused by the lack of human precision which is not the primary focus in mobile context-aware system. The aforementioned facts were the main factors why we decided to use rule-based solution for context-based modeling and reasoning. Therefore, the primary objective of the research presented in this paper was to find the best uncertainty handling mechanism that will support rule-based knowledge representation and solve most common uncertainties that are present in mobile context-aware systems.

The rest of the paper is organized as follows. Section 2 presents current state of the art and discusses main drawbacks of available solutions with respect to mobile context-aware systems and presents the motivation for our work. Section 3 describes our approach of applying certainty factor algebra to ALSV(FD) logic. It also tackles the issue of modeling dynamics of certainty factors. A simple use case scenario is presented in Section 4 and summary and possible future work was included in Section 5.

2 Related Work and Motivation

Uncertainty of data may be defined in different ways and can be caused by various different factors. However, we can distinguish three general types of uncertainties [19]:

1. Uncertainty due to lack of knowledge – that comes from incomplete information both at the model level or if the information is not provided by the sensors,
2. Uncertainty due to lack of semantic precision – that may appear due to semantic mismatch in the notion of the information,
3. Uncertainty due to or lack of machine precision – which covers machine sensors imprecision and ambiguity. Although the lack of machine precision may also be caused by erroneous sensors readings, this type of uncertainty is beyond the scope of this classification.

Among many proposals of uncertainty handling mechanisms [21] like Hartley Theory, Shannon Theory, Dempster-Shafer Theory, the following have been found the most successful in the area of context-awareness:

- Probabilistic approaches, mostly based on Bayes theorem, that allows for describing uncertainty caused by the lack of machine precision and lack of knowledge [12,5].
- Fuzzy logic, that provides mechanism for handling uncertainty caused by the lack of human precision [8,22]. It ignores law of excluded middle allowing for imprecise, ambiguous and vague descriptions of knowledge.
- Certainty factors (CF), that describe both uncertainties due to lack of knowledge and lack of precision [9,1]. They are mostly used in expert systems that rely on the rule-based knowledge representation.

– Machine learning approaches, that use data driven rather than model driven approach for reasoning [14]. They allow for handling both uncertainties due to lack of knowledge and lack of precision.

Methods presented above provide different capabilities of representing and handling diverse types of uncertainties listed at the begining of the section. They also require different implementation effort. The comparison of uncertainty handling mechanisms with respect to these criteria was presented in Table 1.

Machine learning approaches deal very well with uncertainties caused by the lack of knowledge and imprecise measurements, as they provide high generalization features, which allows them to make correct decisions on previously unseen data. Probabilistic methods provide handling mechanisms that best fits uncertainties caused by the lack of precision and ambiguity, as they can express vague information in terms of probability. It allows to project the uncertainty to the output and value it with respect to the probability. However, implementation effort of both probabilistic and machine learning approaches is rather high. What is more, the model cannot be directly modified by the user, as it requires expert knowledge in probability theory and machine learning.

Fuzzy logic allows for imprecise, ambiguous and vague descriptions of knowledge. This is very often source of uncertainties caused by the lack of human precision as human operates on concepts that semantic notion is vague as ”tall”, ”small”, etc. Although this type of uncertainty is also present in context-aware systems, it is beyond the scope of this paper.

Certainty factors are able to describe both uncertainties related to lack of knowledge, and related to lack of machine precision, which are the most common uncertainties in context-aware systems. One of the main advantages of certainty factors over other uncertainty handling mechanisms is that they can be easily incorporated into existing rule-based system without the necessity of redesigning or remodeling knowledge base. They also require a very low implementation effort.

Table 1. Comparison of uncertainty handling mechanisms. Full circles represent full support, whereas empty circles represent low or no support.

	Uncertainty source			
	Lack of knowledge	Semantic imprecision	Machine imprecision	Implementation effort
Probabilistic	◗	❍	●	High
Fuzzy Logic	❍	◗	◗	Medium
Certainty Factors	◗	❍	●	Low
Machine learning	●	❍	●	High

From the comparison presented in the Table 1 we choose certainty factors as the best method for modeling most common uncertainty in mobile context-aware systems that are: uncertainty due to lack of knowledge and lack of precision. Certainty factors cope well with these uncertainties and are easy to design and implement. What is more, together with rules they can be easily understood and modified by the user, which is one

of the most important features in nowadays user-centric intelligible systems. Therefore the primary motivation for this work was to incorporate modified certainty factor algebra into a logic-based knowledge representation called XTT2 in a way that fits best the mobile environment requirements. These are defined as: ability to adapt to dynamically changing context and ability for handling uncertainties caused by the lack of knowledge and lack of precision. We decided to use XTT2 rule-based knowledge representation [15], as it is used by the HeaRTDroid – a prototype of a lightweight rule inference engine dedicated for mobile devices [16]. These required us to 1) incorporate certainty factors handling in ALSV(FD) logic which is the foundation of rule representation used in XTT2, 2) provide dynamic adaptation of certainty factors, both at the ALSV(FD) formulae level and on the rule representation level, 3) propose inference strategy that will allow for making decisions under uncertainty. The following sections describe in details the results of our research.

3 Applying Certainty Factors to ALSV(FD) Logic

Certainty factors (CF) are one of the most popular methods for handling uncertainty in rule-based expert systems. However, for a long time they were under strong criticism regarding lack of theoretical background and the assumption of independence of conditions for rules of the same conclusion which not always hold [10]. As a response to these, the Stanford Modified Certainty Factors Algebra was proposed [20]. It accommodated two types of rules with the same conclusion: cumulative rules (with independent list of conditions) and disjunctive rules (with dependent list of conditions). As it will be shown in this section, this makes the certainty factors fit ALSV(FD) logic *generalised* and *simple* attributes.

The basic elements of the language of Attribute Logic with Set Values over Finite Domains (ALSV(FD) for short) are attribute names and attribute values. There are two attributes types: *simple* which allows the attribute to take a single value at a time, and *generalized* that allows the attribute to take set values. The values that every attribute can take are limited by their domains. For the purpose of further discussion let's assume that: A_i represents some arbitrarily chosen attribute, D_i is a domain of this attribute, and V_i represents a subset of values from domain D_i, where $d_i \in V_i$. Therefore we can define a valid ALSV(FD) formula as $A_i \propto d_i$ for simple attributes, where $\propto$ is one of the operators from set $=, \neq, \in, \notin$ and $A_i \propto V_i$ for generalized attributes, where $\propto$ is one of the operators from set $=, \neq, \sim, \not\sim, \subset, \supset$.

3.1 Certainty Factors Algebra

Rule in CF algebra is represented according to formula:

$$condition_1 \wedge condition_2 \wedge \ldots \wedge condition_k \rightarrow conclusion \tag{1}$$

Each of the elements of the formulae from equation (1) can have assigned a certainty factor $cf(element) \in [-1; 1]$ where 1 means that the element is absolutely true; 0 denotes element about which nothing can be said with any degree of certainty; -1

denotes an element, which is absolutely false. The CF of the conditional part of a rule is determined by the formulae:

$$cf(condition_1 \wedge \ldots \wedge condition_k) = \min_{i \in 1 \ldots k} cf(condition_i)$$

The CF of conclusion C of a single *i-th* rule is calculated according to a formula:

$$cf_i(C) = cf(condition_1 \wedge \ldots \wedge condition_k) * cf(rule) \tag{2}$$

The $cf(rule)$ defines a certainty of a rule which is a measure of the extent, to which the rule is considered to be true. It is instantiated by the rule designer, or it comes from a learning algorithm (like for instance an association rule mining algorithms). Major departure from the traditional Stanford Certainty Factor Algebra [4] is an attempt to remove the major objection raised against it concerning conditional dependency of rules with the same conclusions. To address this issue, rules with the same conclusions were divided into two groups: *cumulative* ans *disjunctive*. Cumulative rules have the same conclusions and have independent conditions (i.e. value of any of the conditions does not determine values of other rules conditions). The formula for calculating the certainty factor of the combination of two cumulative rules is given in (3).

$$cf(C) = \begin{cases} cf_i(C) + cf_j(C) - cf_i(C) * cf_j(C) & \text{if } cf_i(C) \geq 0, cf_j(C) \geq 0 \\ cf_i(C) + cf_j(C) + cf_i(C) * cf_j(C) & \text{if } cf_i(C) \leq 0, cf_j(C) \leq 0 \\ \frac{cf_i(C) + cf_j(C))}{1 - \min\{|cf_i(C)|, |cf_j(C)|\}} & \text{if } cf_i(C) cf_j(C) \notin \{-1, 0\} \end{cases} \tag{3}$$

Disjunctive rules have the same conclusions but are conditionally dependent (i.e. value of any of the conditions determine values of other rules conditions).

The equation for calculating certainty factor of a disjunctive rule is presented in (4).

$$cf(C) = \max_{i \in 1 \ldots k} \{cf_i(C)\} \tag{4}$$

The calculation of the CF for the rules are performed incrementally. This means that for instance for a pair of rules $i - th$ and $i - th + 1$, there is calculated certainty factor $cf_k(C)$ that later is taken into the equation (3) or (4) together with rule $i - th + 2$ to calculate $cf_{k+1}(C)$.

3.2 Certainty Factors in ALSV(FD) Formulae

Every ALSV(FD) formula is a logical expression that can be either true or false according to a value of an attribute in consideration. We can therefore translate every ALSV(FD) formula as a conjunction or alternative of equality formulae. In particular the formula $A_i \in V_i$ can be translated into a form:

$$(A_i = V_i^0) \vee (A_i = V_i^1) \vee \ldots \vee (A_i = V_i^k) \tag{5}$$

where the V_i^k is a *k-th* element from a subset V_i of domain D_i, and A_i is a simple attribute. On the other hand, for the general attributes A_i, the formulae of a form $A_i \sim V_i$ can be translated into:

$$(A_i^0 \in V_i) \vee (A_i^1 \in V_i) \vee \ldots \vee (A_i^k \in V_i) \tag{6}$$

where A_i^k is a *k-th* element of a set representing by the general attribute A_i. This formula can be further recursively rewritten as a conjunction of formulaes from equation (5).

Similarly we can continue for every formula in the ALSV(FD) logic. Such a notation allows us to use certainty factors algebra for evaluating the formulae for uncertain attributes values, treating these formulae as a set of cumulative or disjunctive rules. In particular, we can represent the alternative of equality formulae from equation (5), as a set of logical rules of a form:

$$\begin{aligned} (A_i = V_i^0) &\rightarrow Satisfied \\ (A_i = V_i^1) &\rightarrow Satisfied \\ &\ldots \\ (A_i = V_i^k) &\rightarrow Satisfied \end{aligned} \tag{7}$$

Every rule CF is assigned a value 1 for simplicity, so the certainty of a formula is determined by the certainty of conditional expressions on the left hand side. The rules are disjunctive, as the value of A_i can be only one (as it is simple attribute), hence the equation (4) applies to this. On the other hand, rule interpretation of formulae (6) generates a set of cumulative rules, as the attribute A_i can take multiple values that do not depend on each other, and hence the equation (3) applies to this case. Real-life examples of these transformations were given in Section 4.

What is more, when dealing with logic that operates finite domains, the negative certainty factors may be as valuable as the positive ones. Let us consider the example from equation (7). Assuming that $V_i^{'} = D_i \setminus V_i$, we can add additional rule to the equation, that will cover the *false* cases of the ALSV(FD) fromula $A_i \in V_i$:

$$(A_i \neq V_i^{'0}) \wedge (A_i \neq V_i^{'1}) \wedge \ldots \wedge (A_i \neq V_i^{'l}) \rightarrow Satisfied \tag{8}$$

Supposing that we have no positive certainty on the value of attribute A_i, but we know which of the values the attribute does not take for sure, we can notice the dependence below:

$$\Big(cf(A_i = V_i^{'l}) = -1\Big) \Rightarrow \Big(cf(A_i \neq V_i^{'l}) = 1\Big)$$

The formula above can now be applied together with rule from equation (8) to infer the certainty factor of the ALSV(FD) formulae in consideration.

3.3 Modeling the Dynamics of Certainty Factors

In many context-aware systems that operate in highly dynamic environments, once observed data cannot be treated as certain for unlimited period of time. For instance user activity observed five minutes ago, may not be certain right now. Therefore, adding expiration time to attributes values can improve uncertainty handling in case of lack of

data. The expiration time may be assigned to the attribute in a form of a function over time that decreases certainty factor of an attribute value. Let us consider the expiration time for a value of attribute A to be defined as $expiration(A)$. The simplest expiration time function may be defined as a linear function, that decreases certainty factor of an attribute value over time down to a zero:

$$cf(V, \Delta t) = \begin{cases} cf(V) * \frac{expiration(A) - \Delta t}{expiration(A)} & \text{if } \Delta t \leq expiration(A) \\ 0 & \text{otherwise} \end{cases}$$

where Δt is a difference between the current time and the time when the value V of the attribute A was observed. Different attribute types may have different expiration time functions assigned. Expiration time may be assigned arbitrarily by the system designer in cases when the attribute expiration time is not influenced by the environment dynamics. In other cases, the expiration time function should be dynamically adjusted with respect to the environment dynamics.

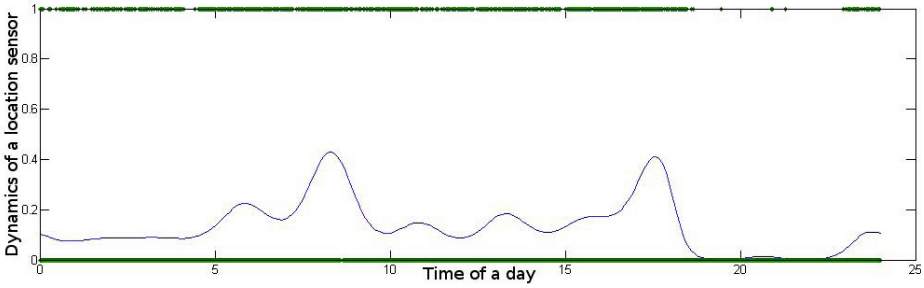

Fig. 1. Dynamics of a location sensor over time [3]

Such a functionality can be achieved with learning middleware approach [3]. Learning middleware is a system that uses linear regression to learn sensors usage patterns from historical data. It automatically generates a model of sensor activity which can be used to dynamically modify the expiration time for the attributes values.

Figure 1 shows the exemplary GPS sensor usage pattern obtained by the learning middleware. The curve describes the probability of the sensor change its reading in particular point of time. We can define this probability as a dynamics of the attribute. Let us consider expiration time for a value of an attribute A to be defined as $expiration(A)$, and the dynamics of an attribute value over time obtained from the learning middleware to be defined as $dynamic(A, t)$, where $dynamic(A, t) \in [0; 1]$. We can now define the dynamic expiration time of an attribute A as

$$expiration(A, t) = expiration(A) * (1 - dynamic(A, t))$$

This will allow to shorten the expiration time in cases where there is a high probability, that the value of the sensor will change, and leave the long expiration time in cases where there is a very low probability that the value of the sensor will change (i.e. location at night).

One of the main disadvantages of the learning middleware is that it needs time and data to learn sensor dynamics. The other approach that allows to discover sensor dynamics is based on the entropy of previous n readings. The entropy is a measure of amount of uncertainty in the data. It can be calculated according to the equation below:

$$entropy(A, n) = -\sum_{x \in X} \frac{x}{n} \log_2 \frac{x}{n}$$

where X is a set of all different readings, and $\frac{x}{n}$ is a proportion of the number of readings such that $A = x$ to total number of readings taken into consideration. Assuming that $n = 4$ and we have following readings from GPS sensor: *still, still, moving, moving*, the entropy of this data equals 1, because we have equal number of *still* and *moving* readings. This is equivalent for high dynamics of data. However, if the readings from the sensor looks as follows *still, still, still, still*, the entropy equals 0, which is an equivalent for low dynamics of data. The expiration dime can be therefore determined according to the equation:

$$expiration(A, n) = expiration(A) * (-log_2 \frac{1}{n} - entropy(A, n))$$

4 Applying Certainty Factors to XTT2 Tables

The certainty factor handling mechanism described in Section 3 operates on the level of ALSV(FD) formulaes, which are foundation of the rule-based knowledge representation called eXtended Tabular Trees [18] version 2 (XTT2 for short). An XTT2 rule is of the form:

$$(condition_1) \wedge (condition_2) \wedge \ldots (condition_n) \longrightarrow RHS$$

where $condition_i$ is one of the admissible ALSV(FD) logic formulaes, and RHS is the right-hand side of the rule covering conclusions. In practice the conclusions are restricted to assigning new attribute values, thus changing the system state. Similar rules are grouped within separated tables, and the system is split into such tables linked by arrows representing the control strategy. An example of XTT2 table is presented in Figure 2. It describes a fragment of a context-aware recommendation system, that based on the user activity, weather and user profile suggests nearby points of interests.

(?) weather	(?) {user_profile}	(?) activity	(->) poi
∈ {sunny,cloudy}	~ {eating}	= any	:= outdor-eating
∈ {rainy}	~ {eating}	∈ {walking,running}	:= indoor-eating
∈ {rainy}	~ {eating}	∈ {driving}	:= drivethrough-eating
∈ {rainy,cloudy}	~ {culture,entertainment}	∈ {walking,driving}	:= theatre-cinema
∈ {rainy,cloudy}	= {culture,sighseeing}	∈ {walking,driving}	:= museum
∈ {sunny}	= {sighseeing,culture}	∈ {any}	:= monuments

Table id: tab_2 - Recommendations

Fig. 2. Example of XTT2 table with uncertain data

The system consists of three simple attributes: *weather, activity* and *poi*, and one generalized attribute: *user_profile*. Let consider, that we know that there is going to be *sunny weather* with certainty 0.3, cloudy with 0.1, and rainy with 0.6. The user selected that he is interested in suggestions about places for eating in 60%, culture in 20%, entertainment in 80% and sightseeing in 20%. User may be independently interested in different recommendations, hence the values are trated as disjoint and the sum does not have to be equal 100%. We also have an information from the activity recognition sensor that the user have been recently walking with certainty 0.8, running with 0.1 certainty and driving with certainty 0.1. Having this information we can now calculate certainty factors for every rule conditions. We use equation (4) (disjunctive rules) to simple attributes, and equation (3) (cumulative rules) to generalized attributes.

After calculation we should get the results presented in Table 2. The last column shows the certainty of a conclusion of a rule calculated according to the equation (2). From the calculations we see that we should suggest user either *indoor-eating* places or *theaters and cinemas*, because both have the highest certainty factors. We have no knowledge on which of these two suggestions should have a greater priority, because the certainty factors for all the rules were assigned 1 for simplicity. This however can be changed in the future by taking into consideration user feedback. If the user decides that a better suggestion would be the *theaters and cinema*, the certainty factor of rule producing this conclusion should be increased (if possible) and the certainty factors of remaining rules can be decreased. This will allow the to make better decisions in the future, when the system faces the same or similar situation.

Table 2. The certainty factors for rules presented in figure 2

(?) weather	(?) user_profile	(?) activity	cf(conditions)	*cf(rule)*	*cf(conclusion)*
0.3	0.6	0.8	0.3	1	0.3
0.6	0.6	0.8	0.6	1	0.6
0.6	0.6	0.1	0.1	1	0.1
0.6	0.84	0.8	0.6	1	0.6
0.6	0.36	0.8	0.36	1	0.36
0.3	0.36	0.8	0.3	1	0.3

5 Summary and Future Work

In this paper we presented an approach for the uncertainty handling in mobile context-aware environments. We provided comparison of application of different uncertainty modeling approaches to this class of systems and chose one that best fits requirements of such environment. These requirements were defined as ability to adapt to dynamically changing context and ability for handling uncertainties caused by the lack of knowledge and lack of precision. Based on the comparison of capabilities of different uncertainty handling mechanisms we decided to use certainty factors. We provided a solution that allows to bind this formalism with ALSV(FD) logic and XTT2 rule-based representation that are used by the HeaRTDroid inference engine dedicated for mobile

platforms. We also described two approaches that allow for automatic adaptation of certainty factors values with respect to dynamically changing context.

As a future work we plan to implement and evaluate the certainty factor based approach described in this paper in HeaRTDroid [1] inference engine. We also plan to use mediation techniques [7] to collect feedback from users and modify certainty factors of XTT2 rules, so they can better fit user preferences. What is more, we would like to compare the approach presented in this paper with a solution that is based on Bayesian networks. Although certainty factors algebra copes very well with uncertainties on the ALSV(FD) level, we believe that it can be successfully replaced by the probabilistic approach on he XTT2 tables level. The XTT2 tables can be interpreted as a tabular conditional probability distributions (CPDs), where the CPDs are learned from statistical analysis of the system performance. The XTT tablesh can be connected to form a graph, which also can be easily translated into Bayesian network without the necessity of redesigning the knowledge base.

References

1. Almeida, A., Lopez-de Ipina, D.: Assessing ambiguity of context data in intelligent environments: Towards a more reliable context managing systems. Sensors 12(4), 4934–4951 (2012)
2. Benerecetti, M., Bouquet, P., Bonifacio, M., Italia, A.A.: Distributed context-aware systems (2001)
3. Bobek, S., Porzycki, K., Nalepa, G.J.: Learning sensors usage patterns in mobile context-aware systems. In: Proceedings of the FedCSIS 2013 Conference, pp. 993–998. IEEE, Krakow (2013)
4. Buchanan, B.G., Shortliffe, E.H.: Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project. The Addison-Wesley Series in Artificial Intelligence. Addison-Wesley Longman Publishing Co., Inc, Boston (1984)
5. Bui, H.H., Venkatesh, S., West, G.: Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. Intl. J. of Pattern Rec. and AI 15 (2001)
6. Chen, H., Finin, T.W., Joshi, A.: Semantic web in the context broker architecture. In: PerCom, pp. 277–286. IEEE Computer Society (2004)
7. Dey, A.K., Mankoff, J.: Designing mediation for context-aware applications. ACM Trans. Comput.-Hum. Interact. 12(1), 53–80 (2005)
8. Fenza, G., Furno, D., Loia, V.: Hybrid approach for context-aware service discovery in healthcare domain. J. Comput. Syst. Sci. 78(4), 1232–1247 (2012)
9. Hao, Q., Lu, T.: Context modeling and reasoning based on certainty factor. In: Asia-Pacific Conference on Computational Intelligence and Industrial Applications, PACIIA 2009, vol. 2, pp. 38–41 (November 2009)
10. Heckerman, D.: Probabilistic interpretations for mycin's certainty factors. In: Proceedings of the First Conference Annual Conference on Uncertainty in Artificial Intelligence, UAI 1985, pp. 9–20. AUAI Press, Corvallis (1985)
11. Hu, H.: ContextTorrent: A Context Provisioning Framewrok for Pervasive Applications. University of Hong Kong (2011)
12. van Kasteren, T., Kröse, B.: Bayesian activity recognition in residence for elders. In: 3rd IET International Conference on Intelligent Environments, IE 2007, pp. 209–212 (2007)

[1] See `http://bitbucket.org/sbobek/heartdroid/overview`

13. Kjaer, K.E.: A survey of context-aware middleware. In: Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering, SE 2007, pp. 148–155. ACTA Press (2007)
14. Krause, A., Smailagic, A., Siewiorek, D.P.: Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array. IEEE Transactions on Mobile Computing 5(2), 113–127 (2006)
15. Ligęza, A., Nalepa, G.J.: A study of methodological issues in design and development of rule-based systems: Proposal of a new approach. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(2), 117–137 (2011)
16. Nalepa, G.J., Bobek, S., Ligęza, A., Kaczor, K.: Algorithms for rule inference in modularized rule bases. In: Bassiliades, N., Governatori, G., Paschke, A. (eds.) RuleML 2011 - Europe. LNCS, vol. 6826, pp. 305–312. Springer, Heidelberg (2011)
17. Nalepa, G.J., Bobek, S.: Rule-based solution for context-aware reasoning on mobile devices. Computer Science and Information Systems 11(1), 171–193 (2014)
18. Nalepa, G.J., Ligęza, A., Kaczor, K.: Formalization and modeling of rules using the XTT2 method. International Journal on Artificial Intelligence Tools 20(6), 1107–1125 (2011)
19. Niederliński, A.: rmes, Rule- and Model-Based Expert Systems. Jacek Skalmierski Computer Studio (2008)
20. Parsaye, K., Chignell, M.: Expert systems for experts / Kamran Parsaye, Mark Chignell. Wiley, New York (1988)
21. Parsons, S., Hunter, A.: A review of uncertainty handling formalisms. In: Hunter, A., Parsons, S. (eds.) Applications of Uncertainty Formalisms. LNCS (LNAI), vol. 1455, pp. 8–37. Springer, Heidelberg (1998)
22. Yuan, B., Herbert, J.: Fuzzy cara - a fuzzy-based context reasoning system for pervasive healthcare. Procedia CS 10, 357–365 (2012)