



Graphs and Patterns for Context-Awareness*

Andrei Olaru and Adina Magda Florea and Amal El Fallah Seghrouchni

Abstract A central issue in the domain of Ambient Intelligence is context - awareness. While previous research in the field presents complex context-aware infrastructures, but with little flexibility and fixed context representations, this paper presents a simple, flexible and decentralized representation of context, for the detection of appropriate context-aware action. This representation is inspired from notions like concept maps and conceptual graphs. A formalism for context patterns, that allows the detection and solution of problems, based on the user's context, is also proposed.

Key words: Multi-agent system, context-awareness, ambient intelligence.

1 Introduction

A true Ambient Intelligence [5] system must be non-intrusive, but also proactive. Appropriate proactive action must fit the context of the user or the user will not have an optimal experience with the system [12]. Additionally, in order to be useful, many times proactive action must result from the anticipation of future situations,

* Original publication at <http://www.springerlink.com/content/8256381213847273/>

Andrei Olaru
Computer Science Department, University Politehnica of Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania, and University Pierre et Marie Curie, 4 Place Jussieu, 75005 Paris, France e-mail: cs@andreiolaru.ro

Adina Magda Florea
Computer Science Department, University Politehnica of Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania e-mail: adina@cs.pub.ro

Amal El Fallah Seghrouchni
Laboratoire d'Informatique de Paris 6, University Pierre et Marie Curie, 4 Place Jussieu, 75005 Paris, France e-mail: amal.elfallah@lip6.fr

based on the current context. Context-aware action and anticipation are key features that will make an AmI system seem "intelligent".

Context has been defined as: "Any information that can be used to characterize the situation of entities (i.e. a person, a place or an object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves" [3]. Most works deal with context as location, location and time or other physical conditions (e.g. temperature) [1, 11], and some also with activity [6]. Situation may be described by means of associations [6], ontologies or rules [11], that are, however, predefined aspects.

In our approach towards an implementation of Ambient Intelligence [9, 10], we are trying to build mechanisms and representations that facilitate a more flexible approach to AmI and context-awareness, while in the same time are easy to implement and can work on resource-constrained devices.

In this paper we present a formalism that allows agents in a multi-agent system, that have only local knowledge, to share and process context-related information and to solve problems by using *context matching* and *context patterns*. Each agent has a representation of the context of its assigned user, including models of other users' context. The focus of this paper is more on defining a manner of representation and problem solving, and less on the algorithms used for context matching or the protocol for the communication between agents.

Context matching is based on representing information about the context of a user as a conceptual graph [13] and on the existence of context patterns, or, in short, *patterns*. Patterns can describe (in variable detail) situations that the user has experienced or common sense knowledge. Two matching context graphs (for two different users) mean a shared context, which is an occasion for further sharing of information between the users' agents. A pattern that matches the user's context means the user has been in the situation before (or it is a well-known, common sense, situation) and this allows the agent to anticipate possible problems, as well as to find solutions to problems already detected.

The next section presents related work in the field of context-awareness. The proposed context representation and scenario are presented in Section 3. Section 4 defines context matching and problem solving. The last section draws the conclusions.

2 Related Work

In previous work in the field of context-awareness there are usually two points of focus: one is the architecture for capturing context information; the other is the modeling of context information and how to reason about it.

Ever since the first works on context-awareness for pervasive computing [4], certain infrastructures for the processing of context information have been proposed [1, 6, 7]. There are several layers that are usually proposed, going from sensors to the preprocessing of the perceived information, the layer for its storage and man-

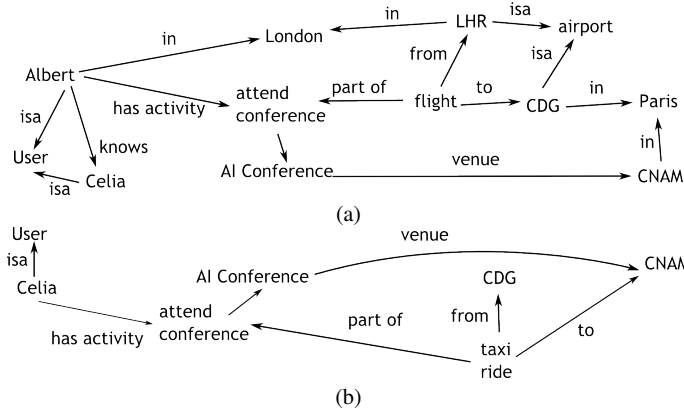


Fig. 1 The knowledge of Albert’s and Celia’s agents, respectively: (a) Albert will attend a conference in Paris; he will reach Paris by plane; (b) Celia will be attending the same conference, and will get from the CDG airport to the conference venue by taxi (her flight is omitted in this picture).

agement, and finally to the application that uses the context information [1]. This is useful when the context information comes from the environment. However, physical context is only one aspect of context [2].

Infrastructures are usually centralized, using context servers that are queried to obtain relevant or useful context information [4, 7]. In our approach [9], we attempt to build an agent-based infrastructure that is decentralized, in which each agent has knowledge about the context of its user, and the main aspect of context-awareness is based on associations between different pieces of context information.

Modeling of context information uses representations that range from tuples to logical, case-based and ontological representations [11]. Henricksen et al use several types of associations as well as rule-based reasoning to take context-aware decisions [6]. While ontologies make an excellent tool of representing known concepts, context is many times just a set of associations that changes incessantly, so it is very hard to dynamically maintain an ontology that describes the user’s context by means of concepts. In this paper we propose a more simple, but flexible and easy-to-adapt dynamical representation of context information, based on the notions of concept map and conceptual graph [8, 13]. While there has been a significant body of work in the domain of ontology alignment, this is not the subject of this paper. We assume ontologies have already been aligned.

3 Context Modeling

This goal of this paper is to present a formalism for the representation of context information, that can be used for assisting the user in solving problems. This section

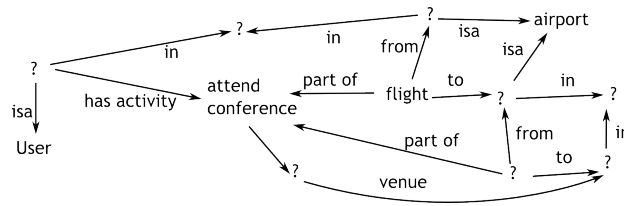


Fig. 2 A pattern that says that attending to a conference requires taking a plane to an airport located in the same city as the venue, as well as going from the airport to the venue of the conference.

covers the representation. We will work on the following example, that is very simple and only meant to illustrate the notions of context graphs and context patterns:

Example. Albert is a researcher in the field of Artificial Intelligence. His professional agenda contains, among other activities, attending the AI Conference, which is held in Paris, at the CNAM. Albert has already booked a flight from Heathrow airport to Charles de Gaulle airport, but has not yet thought about how to get from the airport to CNAM. Celia, another researcher in the same field, will also attend the conference. Besides booking a flight, she has also noted in her agenda that she will be taking a taxi from the airport to the venue of the conference (this reminder will help her later be prepared with money to pay for the taxi). Albert and Celia know each other, and the communication between their respective agents will help solve Albert's problem.

Albert and Celia are both users of the AmIciTy Ambient Intelligence system. What we want is that the system (1) detects the need for a means of transportation for Albert and (2) based on information on Celia's agenda, suggest that a taxi may be an appropriate solution for Albert as well.

Each user of AmIciTy has an associated agent. Figure 1 shows the concept graph for the knowledge of the two agents (agent *A* for Albert and agent *B* for Celia) that is relevant to the scenario (the information about Celia's flight has been omitted in the figure). Formally, the knowledge of each agent can be represented as a graph $G = (V, E)$ in which the values of vertices and edges can be either strings or, better, URI identifiers that designate concepts, relations, people, etc. The value of an edge may be null.

The graph that an agent has contains the knowledge that the agent has about the user and about the user's context. The graph represents the context of the user, in the measure in which the agent has perceived it (or been informed of by the user or by another agent). The manner in which the context information has been gathered is not the focus of this paper.

For the detection of possible problems in the user's context, we define the notion of *context patterns*. A pattern represents a set of associations that has been observed to occur many times and that is likely to occur again. Patterns may come from past perceptions of the agent on the user's context or be extracted by means of data mining techniques from the user's history of contexts. Commonsense patterns may come from public databases or be exchanged between agents. However, the creation

or mining of patterns is not the subject of this paper. An example of a context pattern is presented in Figure 2.

Example. Albert is a researcher for some time now, so he has attended many conferences. His AmIciTy agent (agent *A*) has formed the following pattern: attending a conference located in a certain city usually implies a flight to that city. The flight is between two airports, so a means of transportation between the airport and the venue of the conference is also required.

A pattern is also a graph, but there are several additional features that makes it match a wider range of situations. The graph for a pattern *s* is defined as:

$$\begin{aligned} G_s^P &= (V_s^P, E_s^P) \\ V_s^P &= \{v_i\}, v_i = ? \mid \text{string} \mid \text{URI}, i = \overline{1, n} \\ E_s^P &= \{e_k\}, e_k = (v_i, v_j, E_RegExp), v_i, v_j \in V_s^P, k = \overline{1, m}, \text{ where } E_RegExp \text{ is a} \\ &\text{regular expression formed of strings or URIs.} \end{aligned}$$

4 Context Matching

An AmIciTy agent has a set of patterns that it matches against the current context (graph *G*). A pattern G_s^P (we will mark with " P " graphs that contain special features like ? nodes) *matches* a subgraph G' of *G*, with $G' = (V', E')$ and $G_s^P = (V_s^P, E_s^P)$, iff there exists an injective function $f: V_s^P \rightarrow V'$, so that

$$(1) \forall v_i^P \in V_s^P, v_i^P = ? \text{ or } v_i^P = f(v_i^P) \text{ (same value)}$$

and

$$(2) \forall e^P \in E_s^P, e^P = (v_i^P, v_j^P, \text{value}) \text{ we have:}$$

-if *value* is a string or an URI, then the edge $(f(v_i^P), f(v_j^P), \text{value}) \in E'$

-if *value* is a regular expression, then it matches the values $\text{value}_0, \text{value}_1, \dots, \text{value}_p$ of a series of edges $e_0, e_1, \dots, e_p \in E'$, where $e_0 = (f(v_i^P), v_{a_0}, \text{value}_0)$, $e_k = (v_{a_{k-1}}, v_{a_k}, \text{value}_k)$ $k = \overline{1, p-1}$, $e_p = (v_{a_{p-1}}, f(v_j^P), \text{value}_p)$, $v_{a_l} \in V'$.

That is, every non-? vertex from the pattern must match a different vertex from G' ; every non-RegExp edge from the pattern must match an edge from G' ; and every RegExp edge from the pattern must match a series of edges from G' . Subgraph G' should be minimal.

A pattern G_s^P *k-matches* (matches except for *k* edges) a subgraph G' of *G*, if condition (2) above is fulfilled for $m - k$ edges in E_s^P , $k \in [1, m - 1]$, $m = ||E_s^P||$ and G' remains connected and minimal.

Example. For the pattern in Figure 2 and the example in Figure 1 (a), the pattern 2-matches the knowledge about Albert that his agent has: Albert is attending a conference and he has booked a flight to the city in which the conference takes place. The missing edges relate to the trip from the airport to the venue: the *part of* edge, the *from* edge and the *to* edge, and there is also the means of transportation that is missing. Since something is missing, the agent should ask Albert about that piece of information or to try to find it itself.

This is defined as a *problem*. A *Problem* is a graph G^P that is a partial instantiation of a pattern G_s^P , according to the current context. It is the union between (1) the

```

pattern  $G_s^P$   $k$ -matches  $G \rightarrow$ 
  if  $k > 0$ 
    put problem in the list
  if there is a problem and this can be a solution
    is the solution is certain / complete
    let user know
    user confirms solution
    increase confidence in pattern
  otherwise
    decrease confidence in pattern
  otherwise
    link possible solution to problem

```

Fig. 3 Pseudo-code of the agent's behaviour related to context matching.

subgraph G' of G that k -matches pattern G_s^P and (2) the part of G_s^P that is not matched by G' (the *unsolved* part of the problem). The problem remains associated with the pattern: for the pattern $G_s^P = (V_s^P, E_s^P)$ and the k -matching subgraph $G' = (V', E')$, the problem p is a tuple (G_s^P, G_p^P) , where G_p^P is the problem's graph:

$$G_p^P = G' \cup G_x^P, G_x^P = (V_x^P, E_x^P) \text{ (the unsolved part), where}$$

$$V_x^P = \{v \in V_s^P, v \notin \text{dom}(f)\}$$

$$E_x^P = \{e \in E_s^P \text{ for which condition (2) is not fulfilled}\}$$

$$G_x^P \text{ is a subgraph of } G_s^P.$$

Example. In our scenario, the matching part of the problem contains the nodes *Albert, User, London, attends conference, AI Conference, flight, LHR, CDG, airport, from, to, Paris, CNAM*, and their connecting edges. The unsolved part of the problem contains one ? node and the edges *part of, from* and *to*.

But the agents are part of the AmIciTy multi-agent system. Agents A and B communicate because Albert and Celia know each other (they share common social context). They also share the same field of research. Based on the fact that the node *AI Conference* relates to Albert and to Artificial Intelligence as a domain (the same nodes to which Celia relates to), agent A will send to Celia the association between Albert, the domain, and the conference. This is common context, so Celia will respond with the data connected to the conference: venue and the means of transportation that she will use to get there (see Figure 1 (b)).

All the data that agent A has about other agents (here, agent B) is stored in the agent's knowledge base as its model of the other users. The model for the other users is not necessarily separate though: if the same concept appears in both models (provided the concept has the same URI, or the agent is able to detect by means of common sense knowledge that it is the same concept), both subgraphs will contain the corresponding node. The model for Celia's agenda contains the same *AI Conference* node that is contained in the graph for Albert. When matching patterns from its pattern set, A detects that the pattern mentioned above fully matches the model for Celia. Agent A also has a *problem* that is linked to this pattern. Since Celia's context fully matches the pattern, it means it may be a *solution* to Albert's problem: Albert may also use a taxi to reach the conference.

In this particular case, with the given knowledge and patterns, there is only one solution to the problem that arose. But in a more realistic case, where context is more complex and there are more patterns, more solutions to the same problem may be found. In case they fit equally well in the current context, then the agent must prompt the user with all of them and the user must be given the choice.

It can be argued that context-matching is a very difficult problem in the case of large graphs and complex situations. However, resource-constrained devices will work only with smaller pieces of context information (i.e. smaller graphs), that are relevant to their function. Second, algorithms inspired from data-mining allow for incremental matching, starting from common nodes and growing the matching sub-graph.

Another problem that may appear in realistic situations (as opposed to our simple example) is the abundance of simultaneous matching context patterns, possibly describing contradictory situations. This is where more refined measures must be found that will allow calculating the relevance of each match. This too will be part of our future work.

5 Conclusion

So far, work in context-awareness for pervasive environments has been based predominantly on location-awareness and physical conditions. The use of ontologies or rules does not bring much dynamical flexibility and they are not easy to modify automatically, at runtime.

This paper proposes a more simple and more flexible manner of representing context and context patterns, that allows the agents to take decisions without the need for a centralized structure, by means of their knowledge, their history and by local communication alone.

At this point, we are in the process of identifying an efficient matching algorithm, as well as deploying the described formalism into a previously implemented multi-agent system. There is a great potential in detecting incompatible contexts – contexts that the user should not be in. Also, uncertainty and temporal relations have yet to be included in our work.

Acknowledgment

This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 1315/2008 and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/16..

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* **2**(4), 263–277 (2007)
2. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College (2000)
3. Dey, A.: Understanding and using context. *Personal and ubiquitous computing* **5**(1), 4–7 (2001)
4. Dey, A., Abowd, G., Salber, D.: A context-based infrastructure for smart environments. Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99) pp. 114–128 (1999)
5. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Scenarios for ambient intelligence in 2010. Tech. rep., Office for Official Publications of the European Communities (2001)
6. Henriksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing* **2**(1), 37–64 (2006)
7. Lech, T.C., Wienhofen, L.W.M.: AmbieAgents: a scalable infrastructure for mobile and context-aware information services. Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands pp. 625–631 (2005)
8. Novak, J.D., Cañas, A.J.: The origins of the concept mapping tool and the continuing evolution of the tool. *Information Visualization* **5**(3), 175–184 (2006)
9. Olaru, A., El Fallah Seghrouchni, A., Florea, A.M.: Ambient intelligence: From scenario analysis towards a bottom-up design. In: M. Essaïdi, M. Malgeri, C. Badica (eds.) *Intelligent Distributed Computing IV*, Proceedings of the 4th International Symposium on Intelligent Distributed Computing - IDC 2010, Tangier, Morocco, September 16-18 2010, *Studies in Computational Intelligence*, vol. 315, pp. 165–170. Springer Berlin / Heidelberg (2010). DOI 10.1007/978-3-642-15211-5_17. URL http://dx.doi.org/10.1007/978-3-642-15211-5_17
10. Olaru, A., Gratie, C.: Agent-based information sharing for ambient intelligence. In: M. Essaïdi, M. Malgeri, C. Badica (eds.) *Intelligent Distributed Computing IV*, Proceedings of the 4th International Symposium on Intelligent Distributed Computing - IDC 2010, MASTS Workshop, Tangier, Morocco, September 16-18 2010, *Studies in Computational Intelligence*, vol. 315, pp. 285–294. Springer Berlin / Heidelberg (2010). DOI 10.1007/978-3-642-15211-5_30. URL http://dx.doi.org/10.1007/978-3-642-15211-5_30
11. Perttunen, M., Riekkilä, J., Lassila, O.: Context representation and reasoning in pervasive computing: a review. *International Journal of Multimedia and Ubiquitous Engineering* **4**(4), 1–28 (2009)
12. Riva, G., Vatalaro, F., Davide, F., Alcañiz, M. (eds.): *Ambient Intelligence*. IOS Press Amsterdam (2005)
13. Sowa, J.: *Knowledge representation: logical, philosophical, and computational foundations*. MIT Press (2000)