

User Modelling Using Conceptual Graphs for Intelligent Agents

James F. Baldwin, Trevor P. Martin, Aimilia Tzanavari

A.I. Group, Department of Engineering Mathematics,
University of Bristol,
Bristol BS8 1TR. UK

{Jim.Baldwin, Trevor.Martin, A.Tzanavari}@bristol.ac.uk

Abstract. We describe a novel inference method to help us deal with the problem of having sparse information about the user. The information we will have available for the user will form his/her model. That model will be represented in a conceptual graph format. We will also have gathered information about categories of users that share characteristics, preferences and interests. These will form the "prototypes", as we call them and will be represented in graphs as well. Because we will know significantly more about the prototypes, they will be the source where we will try to get the information we want for the user. This method has been implemented in FRIL [2, 3, 10].

1 Introduction

In the past software systems used to be relatively simple because technological capabilities were limited. They tended to be appropriate for performing very specific tasks only. Subsequently, a limited number of people were using them; the ones that had relevant background and needed to use them. However, nowadays things have changed: systems have become comparatively complex and the users that are meant to use them form very diverse user groups. They have different characteristics, needs, abilities, preferences and interests. As a result, software systems have to become more individualised and cater for those differences and not treat all the users in the same manner. Their scope is to make man's life as easy as possible.

User modelling [9, 12, 14] can be identified in general terms as the part of the software system's design that deals with the aforementioned problems. Since nowadays software systems structure is becoming more and more agent-based, we can safely assume that user modelling will be taken care of by either a dedicated agent, or by an agent's component.

Intelligent Agents can be defined as Personalised Assistants that "look over the user's shoulder" and learn the user's characteristics in order to act for his interest [13, 16]. Indeed, the user's characteristics, which of course will be related to the software system's purpose, form the core element of the individualisation. This information is called the user's model. The user's model, in combination with the relevant IA

knowledge base will be used by the Intelligent Agent to predict the human responses, the human needs and suggest to the user things to do, help the user utilise his computer more efficiently.

There are two problems related to this procedure. The first is concerned with the knowledge representation of the user's model, how we build the model, how we use the model with the other IA knowledge bases to make inferences and decisions. The second one is concerned with the fusion problem of individual inferences.

In this paper we will mainly concentrate on the first problem and discuss a form of knowledge representation and inference method, which is novel and new. We will then discuss the fusion problem and give a solution, which is suitable for our case but not unique. Examples will be given throughout the analysis of the method.

2 Knowledge Representation of the User Model and of the Prototypes

It is important to recognize that we will only have sparse information about a particular user and while this will build up over time, we should not expect to be able to use learning methods that require large amounts of data to provide the user model. Essentially, this means that we need to find a way to make inference when we only have sparse data available.

In human terms we do this all the time. We meet someone and make decisions about that person with reference to a set of prototypical persons, which we have built over time. For example, if we knew someone that worked in a software house, we could assume with a certain degree of confidence that he would know how to program. We should collect data for building up clusters of types of users, according to their behaviour, their abilities, needs, etc and use these clusters as prototypes. The way in which the system acquires relevant information to build them is a separate problem. It can produce them by extracting information from a database of individuals, or the user can provide it with his understanding of prototypes or refine existing ones. For our purpose, we will assume that we have a collection of prototypical people.

As for the user's model, the easiest way would be to ask him directly to pick a model (categorise himself) from an existing model database. That would leave everything to the user. Another way would be to ask the user for relevant information, possibly giving him a set of choices for each characteristic we are interested in. Finally, the most advanced method and at the same time the least demanding from the user's point of view, would be creating the model by learning the necessary information.

The next question that we need to answer is how can the system acquire relevant information about the user. By observing the user's real-time interaction with the

system, we can learn quite a lot about him, his preferences, his interests, his habits, etc. Additionally, all the systems that use user-profiling techniques, should give their user a certain degree of freedom as well as allow him to get involved with improving the agent's performance. Consequently, the user should be able to give feedback concerning the system's actions and should be able to improve his own profile at any time.

The representation of the clusters/ prototypes we mentioned earlier can be done in different ways. One approach would be to represent each person in the cluster as a vector point and to find the average vector for the cluster and this would represent our prototype for the cluster. Having a collection of prototypes represented in this way, when presented with a particular user, we would match him/her to the nearest prototype vector. However, we consider this to be a rather simplistic form of representation because first of all it does not allow us to consider relationships between concepts, but only basic attributes. In addition, there is a lack of flexibility as a result of having just a single point representing a prototype. Consequently, we decided to use Conceptual Graphs that satisfy both of the above requirements. In this way, we will be able to represent a cluster as a set of attributes and their relations. Conceptual graphs are described in greater detail in a following section.

Let us give an example to illustrate the difference between the two above forms of representation and identify the advantages of the second one. Supposedly, we had to represent a cluster of almost perfect circles. One way to achieve this would be to take a prototypical circle with radius r and centre c and create the vector (r, c) . We would then accept a circle as a member of this cluster, if its average radius and centre lie close to r and c . A second way of representing the same cluster would be by using fuzzy sets [11]. Specifically, by defining the radius as a fuzzy set f and the centre having co-ordinates (g_x, g_y) , where g_x and g_y are two fuzzy sets. The fuzzy sets f , g_x and g_y will thus define a family of circles with varying membership. We could describe this family as a fuzzy circle and this will constitute our cluster of acceptable circles. To decide whether a new almost perfect circle belongs to this cluster, we would have to check whether it lies in the aforementioned fuzzy circle. Relations between attributes can be defined as well. In our example, having the attributes minimum diameter and maximum diameter of a circle, we can define the difference between the two, which will have a maximum magnitude for accepted circles. We can then say that for any almost perfect circle, this difference must be g where g is a fuzzy number. Conceptual graphs are suitable for efficiently describing attributes as fuzzy sets and relations between them.

3 The Philosophy Behind our Approach

We will collect information on users based on their interaction with the computer and will divide them into clusters. The latter will be generic categories of people with similar behaviour and characteristics. Those characteristics will be captured in the cluster's definition. Each cluster will be defined by the relevant fuzzy set attributes

and relationships between them and will be represented by a conceptual graph. This will form an individual prototype. A new person will be checked against this prototype to decide how closely it matches it. He/she will also be checked against all the other prototypes that include information that is of interest. By doing this we assume that a user that partially matches a certain prototypical person's behaviour, will probably possess other features of that specific prototype as well. The computer will be able, when missing some information about its user, to deduce it from prototypical users with similar behaviour. This form of reasoning will be inductive or even analogical and will have no truth guarantee.

4 Conceptual Graphs

Conceptual graphs [15] are finite, connected, bipartite graphs with concept and relation nodes. Concept nodes have a label and a referent field, whereas relation nodes have only labels. The concept node's referent field is the concept's instantiation and can either be a value, a set of values or a fuzzy set [4]. The relation nodes' role is to connect the concept nodes and to represent the relationship between them. Conceptual graphs are related to semantic nets and overcome some of the difficulties such as the "isa" problem, which were found in early uses of the latter. Another major advantage that can be easily identified is that these graphs are closely related to natural language. Every English sentence can be represented with a conceptual graph and every graph corresponds to an English sentence. Here is an example graph that illustrates the aforementioned notions.

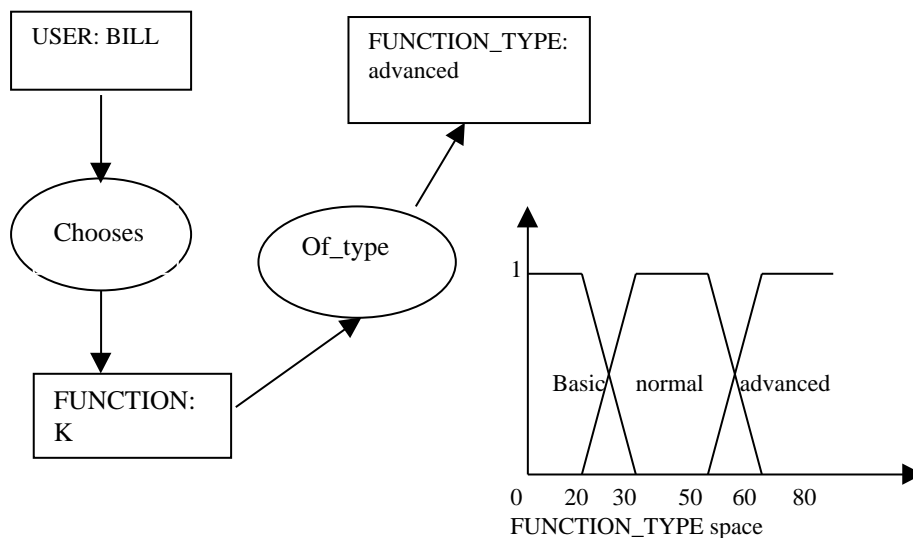


Fig. 1. Example conceptual graph

The graph above is a graph representing the sentence: " Bill chooses advanced functions". We can see that concepts USER and FUNCTION are instantiated to single

values “BILL” and “K” respectively, whereas “FUNCTION_TYPE” to the fuzzy set “advanced”, which is defined at the “Function type” space. The relation nodes “Chooses” and “Of_type” are there to connect the previously mentioned concepts.

5 Outline of the Inductive Inference Mechanism

In this section we will discuss the preliminary ideas for inference with respect to a certain user based on prototypes represented as conceptual graphs. At this point, we will not refer to the source of the information used. We will assume that after studying people in a certain context, we derived a set of prototype conceptual graphs. People with similar behaviour or characteristics, depending on the context, were clustered to the same graph. Furthermore, the user’s graph is developed after collecting relevant information about him. This information can be acquired in several ways: observation of the real-time interaction between him and the system, some direct user-feedback concerning the system’s actions, or by other means which will not be dealt with in this report. In this report we will refer to the prototype graphs as P1, P2, ... Pn and to the user’s graph as S. Examples of these can be seen in figures 2 and 3:

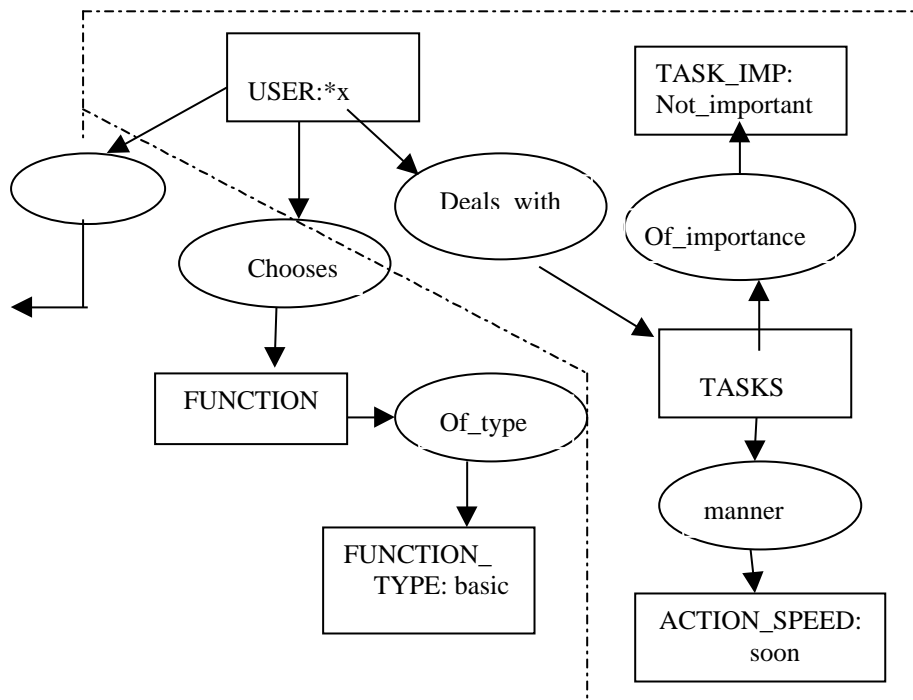


Fig. 2. Prototype graph P1 and P1' (within dotted line)

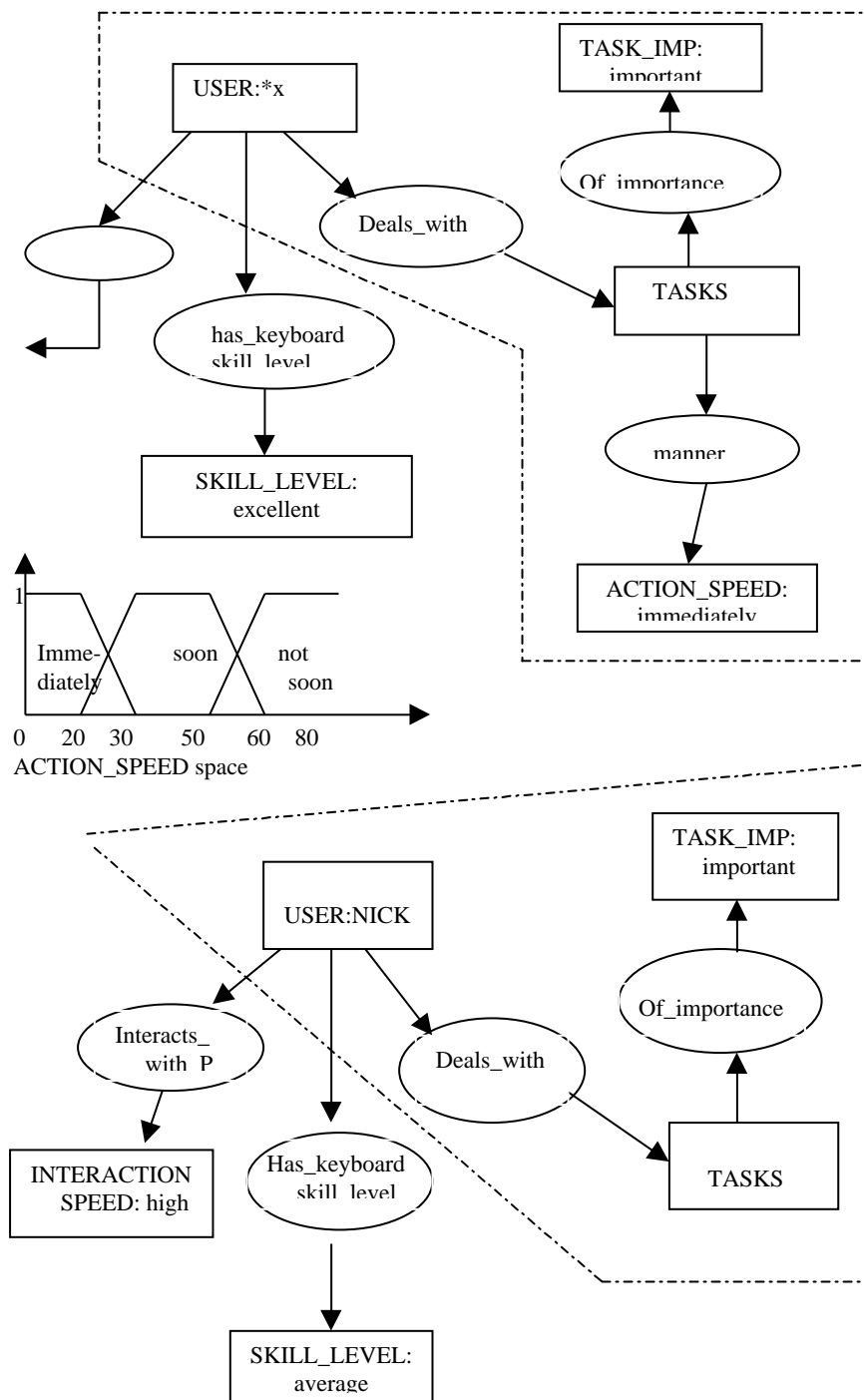


Fig. 3. Prototype Graph P2 and P2' (within dotted line, top) and User Graph S (bottom)

Prototype graphs P1, P2 will be representative of the prototype graph category. In our example we will only use two graphs for illustrative purposes. However, in reality there will be a significantly greater number.

As we mentioned above, the computer will not always have all the necessary information on a particular user in order to take action on his behalf. Its knowledge will be limited, so it should make inferences from the information it has available. If the user were one of the prototypes, then the answer would be given by part of the respective graph.

The information that the computer will be looking for, or in other words the information that will be missing from the user's graph, could be expressed in a query/question format when using natural language. For our example, let us suppose we would like to answer the question: "How quickly does the user act when dealing with important tasks?" Since every natural language sentence can be represented with a conceptual graph, we do this for our question and we end up with the graph Q in figure 4. To construct Q, we must use concept and relation node labels that exist in the prototypes as well, in order to achieve some matching. The information required will be concept nodes with empty referent fields (ACTION_SPEED in this case).

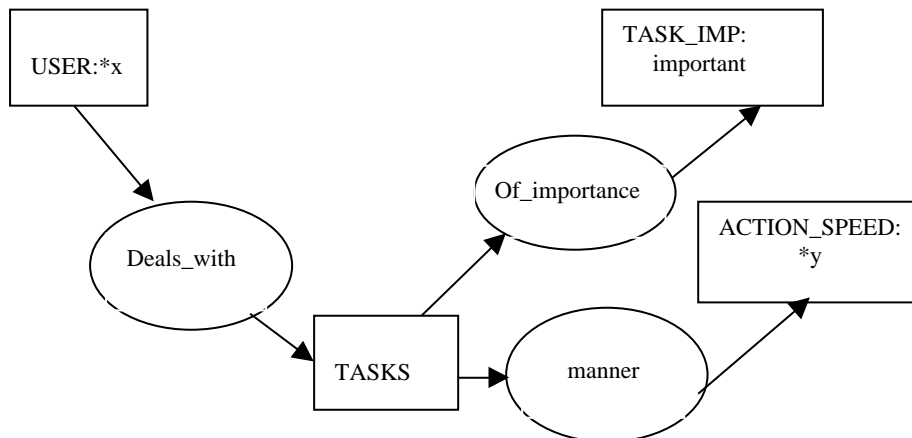


Fig. 4. Question/ Query Graph Q

We will match graph Q to each of P1, P2...Pn – this will correspond to a maximal join operation – all nodes in Q should match corresponding nodes in P1, P2...Pn. If nodes exist in Q that do not find matching nodes in a certain prototype graph, then that graph is not capable of providing us with the information we are looking for and so we do not consider it any further. After the maximal join, in the resulting graphs, the concepts with the empty referent fields will be instantiated. These graphs will look like the sub-graphs defined by the dashed line drawn on P1 and P2 with the appropriate instantiations. For each graph, we identify the sub-graphs that correspond to the answer and strip the remaining concepts and nodes. The new graphs obtained

will be called P'1, P'2...P'n. At this point we need to mention that the stripping operation would be more efficient if we kept parts of the graph that might be relevant to our question. Looking at the main concept of the question graph and keeping sub-graphs that directly support it, can possibly identify these parts. This will further be analysed in following reports.

The graph S corresponding to the specific user is also treated in this way. Q is mapped onto S and S is stripped of non-relevant nodes to give S'. This will look like the sub-graph defined by the dashed line in S, with an additional ACTION_SPEED concept with empty referent field. S of course will not contain the sub-graph corresponding to the answer, otherwise we would not be referring to the prototype graphs to obtain this information.

We now take S' and match onto P'1, P'2...P'n by performing maximal join operations. This results to graphs P''1, P''2...P''n. These maximal joins will not be complete joins and some measure of completeness will be used to give a support for how well S' matches each of P'1, P'2... P'n. When we talk about matching two graphs, we mean matching one's relations and concepts against the other's. Two relations match when their type is the same. Two concepts match with a support $s=1$ when their types and referents are identical and with a support $s<1$ when their types are the same, but their referents different. Since we consider the case where referents can be fuzzy sets as well [4], in that specific case we perform Point Semantic Unification [1], which is based on the Mass Assignment Theory [5], to obtain a support. Because we get a support from each pair of concepts that is matched, we accept the overall support to be the conjunction of the individual supports. Consequently, we end up with a support for each pair S' and P'n. Let these supports be given by $s_1, s_2...s_n$. In our example, the information we would get from P''1 is that “Nick deals with non-important tasks soon” with a support s_1 and from P''2 that “Nick deals with important tasks immediately” with a support s_2 . We can now pick out the parts of P''1, P''2...P''n that correspond to our answer. Let these answer graphs be A1, A2...An. We do this by projecting P''1, P''2...P''n on to our query graph and identifying the part of the former that does not project on anything. In our example they will just consist of the ACTION_SPEED concept instantiated to one of the fuzzy sets of the respective space (Fig. 3).

At this stage we have a set of answer graphs A1, A2...An with a support for each one. It is now necessary to fuse these to obtain a final answer graph A. This will be done with a combing scheme, based on the Mass Assignment Theory [5], that will take into account the supports as well. Essentially, our answers will have the form of fuzzy sets with accompanying supports. If we wanted to combine for example f_1 and f_2 with supports s_1 and s_2 respectively:

$$\begin{aligned} \text{ExpLeastprejudicedDistribution}(f_1) &= s_1 \text{ LPD}(f_1) + (1 - s_1) \text{ LPD}(\overline{f_1}) = f_1 \\ \text{ExpLeastprejudicedDistribution}(f_2) &= s_2 \text{ LPD}(f_2) + (1 - s_2) \text{ LPD}(\overline{f_2}) = f_2 \end{aligned}$$

$$f_{final} = f1 \quad f2$$

The answer graph A will contain the information required from the computer to act on the user's behalf. The feedback given by the user can be potentially used to adjust and improve the combining schemes used to match the user's preferred choice.

At this present time, a Conceptual Graph Toolkit software package has been implemented in FRIL [2, 3, 10], which allows graphs to be stored in linear format and us to perform several operations such as maximal join, simplification, restriction, etc. Furthermore, the whole inference mechanism described above has been developed.

6 Application - "The Forum"

Researchers at BT's Adastral Park have developed the "Forum" [11], which is an on-line collaborative virtual working environment [6] aiming to bring people together both informally and formally. It is designed to allow people who should meet each other do so easily and naturally and provides the means for them to have richer on-line meetings. It is divided in two parts: the Contact Space and the Meeting Space. We will focus on the first. The Contact Space consists of a number of zones that represent subject interests. According to the user's "overall" interest at a particular moment, his avatar is placed in the appropriate zone along with other users' avatars. An avatar can chat to any other on the same zone.

Our aim in this project is not only to categorize a certain user based on his interaction with the computer and subsequently place his avatar on the relevant zone. That can be achieved in several different ways. We would propose one related to the mechanism described previously, however we will not get into it in detail in this paper. Our aim is to make the whole system act a bit more intelligently. A system is not necessarily intelligent simply because it knows e.g. all the symptoms of 1000 diseases. It must be able to make rational diagnoses based on the presence and absence of different combinations of them. In the Contact Space case, one possibility would be to need to make inference about a user not knowing everything about him, e.g. how much is he interested in interest X. Another would be to have to suggest to someone to go and "chat" with person Y because they have a common interest, apart from the "zone's" interest. We will now analyze some ideas on the way our novel inference method can be applied in the "Forum".

7 Applying Intelligent User Modelling to "the Forum"

Our target is to create prototype user models, which will constitute the different user categories; the user groups in other words. After "constructing" those models and a similar user's model, we will be able to identify similarities between them, which will allow us to draw conclusions. "What should a prototype consist of", is a question that can be answered in a lot of different ways depending on the information we have

available and the scope of our system. It is very crucial to find the right combination of pieces information to form the prototype user models. Two requirements need to be satisfied in every case. To take advantage of the information gathered from different sources, so that important information is not left out, and at the same time not to include too much detail that would make the system too complex and less flexible.

Initially, we need to identify the different sources of information for a particular user. These can be the topics the user works on, the applications he/she uses, the topics he/she searches for on the Internet and his/her interest areas based on Jasper [7, 8]. This kind of information can be easily collected and summarized. It can be generalized in a way that nothing important is omitted and everything that is not of great interest is left out. In any case, the information used should be enough to distinguish one prototype from another.

Each prototype can represent a different zone in Contact Space. That means that we will have a “Software Engineering” prototype, a “Wearable Computing” prototype, a “Distributed Computing”, a “Shared Virtual Worlds”, etc. Each one of them will basically include the characteristics of a person that would definitely belong to that category. For example a “Shared Virtual Worlds” prototype user will be someone whose interests are “Virtual Reality”, “Animation”, “Computer Games”, who works on “VRML” and “C++”, who searches for “avatars” and “navigation in virtual space” on the Internet and who uses applications such as “The Forum” and “Visual C++”.

As we specified in the inference method description, we are going to use conceptual graphs as a method of knowledge representation. These graphs will have concepts instantiated to fuzzy sets in order to be able to obtain a support when performing the matching later on. In our case, the concepts nodes will be key phrases as the ones mentioned above, with instantiations to fuzzy sets that will show the level of interest to that particular topic. The relation nodes will represent the source of information. In figure 5 (top) we can see an example conceptual graph and the fuzzy sets defined in LEVEL_OF_INTEREST space.

A collection of graphs with a similar architecture to the one of graph in figure 6 would play the role of the different user groups. The user himself will have a graph that will encapsulate his characteristics. At that point, we can actually say that we have all the necessary information in an appropriate for manipulation format. We are now able to perform several operations.

One is to compare the user’s graph to all the prototypes and find a degree of matching to each one. That will give us the ability to say in which category and thus in which zone the user belongs to, having his specific attributes in mind. The other probably more useful operation we can perform, as analyzed in the proposed method’s description, is to infer things about the user that we don’t know. Supposedly, we wanted to find out how interested is user K in interest topic 5 (INT5 in graph). We would hypothetically have graphs that would resemble the ones shown in figure 5 (bottom), 6 and 7.

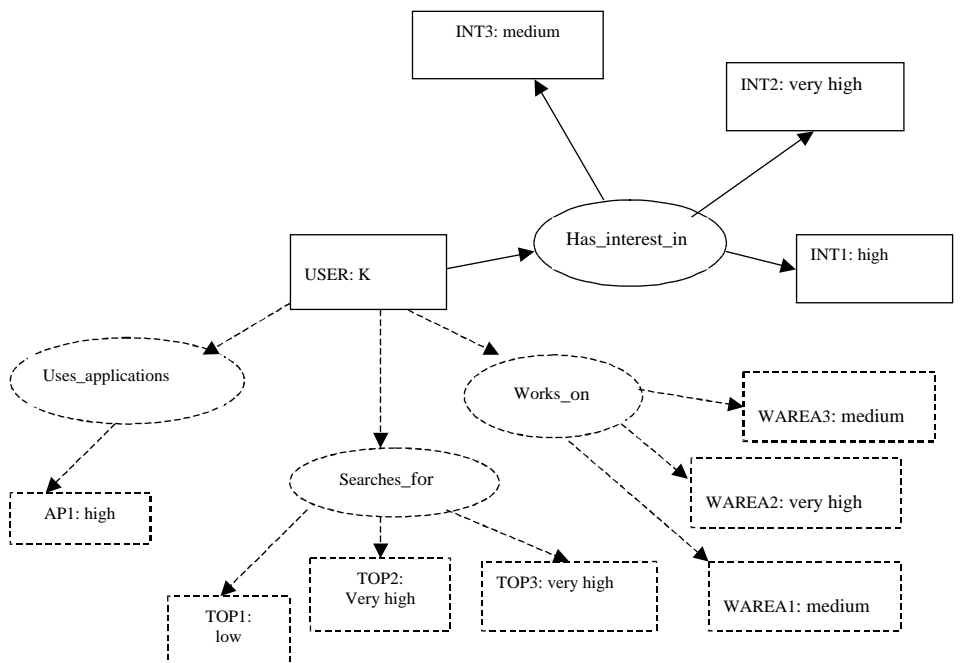
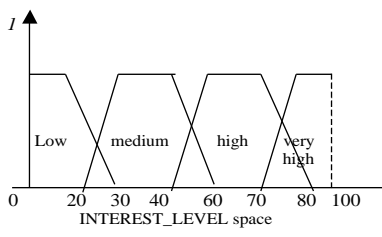
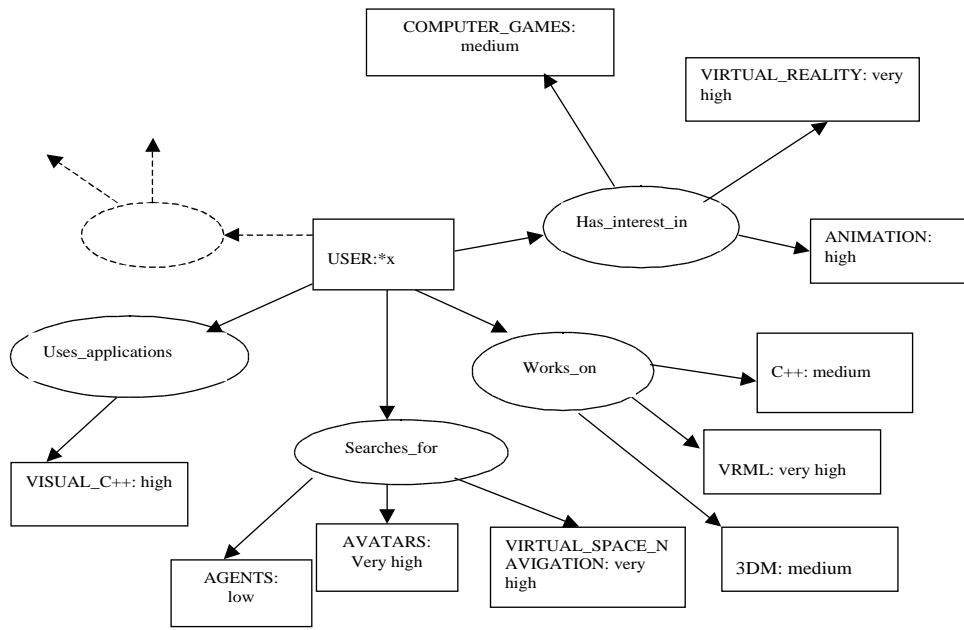


Fig. 5. Example Conceptual Graph (top) and Abstract User's Graph (bottom)

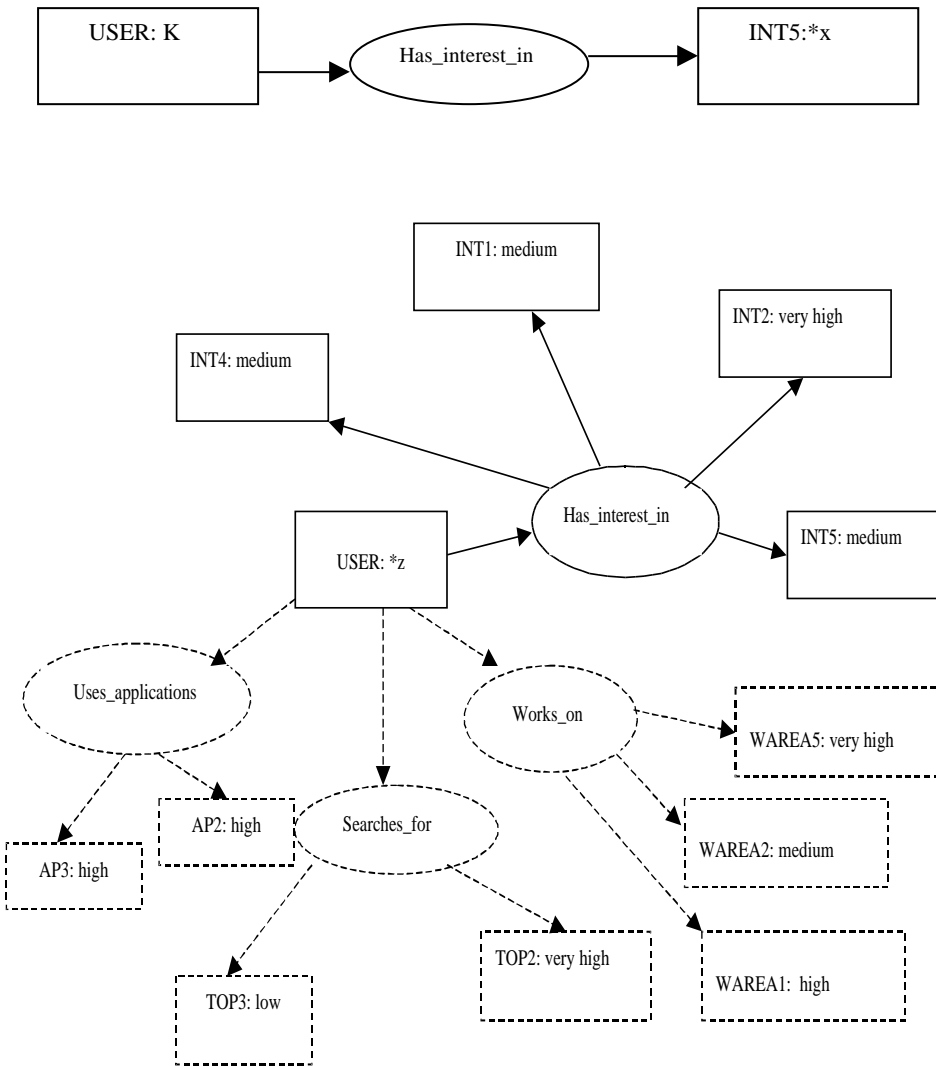


Fig. 6. Query Graph (top) and Prototype Graph 1 (bottom). Concept and relation nodes in dotted lines are stripped

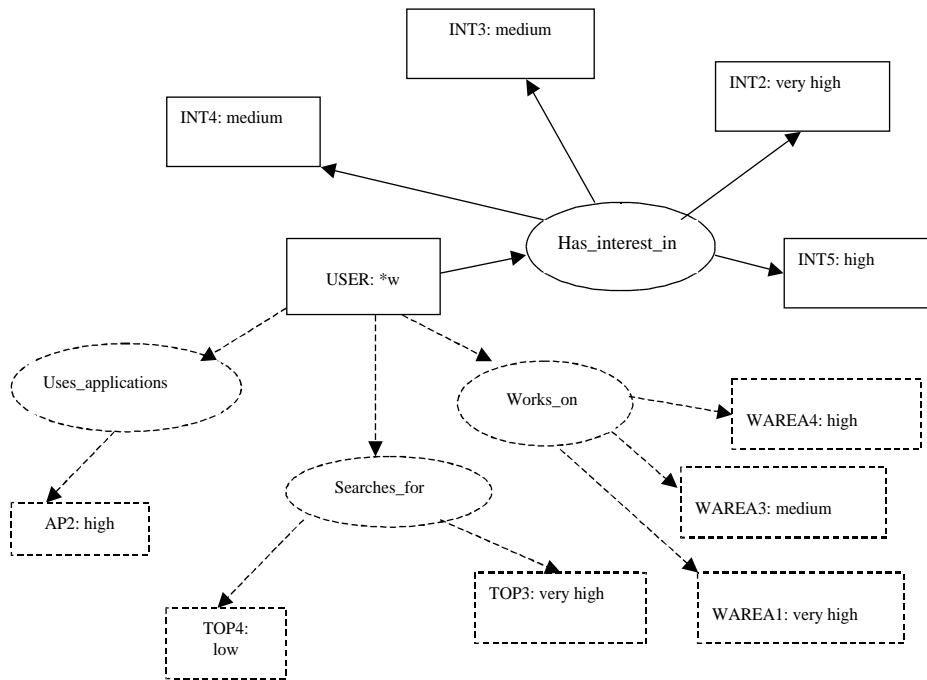


Fig. 7. Prototype Graph 2. Concept and Relation nodes in dotted lines are stripped

By following the procedure described previously, we “strip” both the user graph and the prototype graphs from any irrelevant sub-graphs having the query graph as our guide. The irrelevant parts can be seen in dotted lines. We then match the resulting user’s stripped graph on to each stripped prototype graph. Obviously some concepts match and some do not. The ones that match will probably match with a certain degree/support, which will come from combining their fuzzy sets (point semantic unification [1]). An overall support per prototype-user pair can be obtained by means of conjunction.

The answer, which in our case would be the instantiation of the INT5 concept, can be found by projecting the query graph on the prototypes. Consequently, we end up with a fuzzy set as an answer and a support associated with it, for every prototype we considered. We can combine these using a combination scheme based on Mass Assignment theory that combines probability distributions taking supports into account. In our example, we would get “medium” as an answer from prototype1 with a support S_1 e.g. and “high” from prototype2 with a support S_2 . The combination of these two would result to a fuzzy set defined on INTEREST LEVEL space. Depending on what format we want our answer to have, we can get an exact value in that space, or we can get a linguistic output such as “fairly high” or “rather low”.

8 Summary

We have thoroughly described a novel method that gives us the ability to infer information we do not know about the user, based on the information we already have. The method uses Conceptual Graphs [15] as a form of knowledge representation and employs Fuzzy Set Theory [17] as well as the Mass Assignment Theory [5]. "The Forum" has been introduced as a system to which we can apply our mechanism to obtain advanced results.

References

1. Baldwin, J.F.: Probabilistic, Fuzzy and Evidential Reasoning in Fril. Proc. Two Decades of Fuzzy Control, IEE London (1993) 1-20
2. Baldwin, J.F., Martin, T.P., Pilsworth, B.W.: FRIL - Fuzzy and Evidential Reasoning in AI. Research Studies Press, John Wiley (1995)
3. Baldwin, J.F., Martin, T.P., Pilsworth, B.W.: FRIL Manual. Fril Systems Ltd, Bristol Business Centre, Maggs House, Queens Road, Bristol, BS8 1QX, UK (1988)
4. Baldwin, J.F., Morton, S.K.: Conceptual Graphs and Fuzzy Qualifiers in Natural Language Interfaces. Research Report ITRC-85, University of Bristol (1985)
5. Baldwin, J.F.: A Theory of Mass Assignments for Artificial Intelligence. In: Driankov, D., Eklund, P.W., Ralescu, A.L. (eds): Fuzzy Logic and Fuzzy Control. (1991)
6. Bradley, L., Walker, G., McGrath, A.: Shared Spaces. British Telecommunications Engineering Journal, Vol. 15 (1996)
7. Davies, N.J., Weeks, R., Revett, M.C.: An Information Agent for WWW. Proc. Int. Conf. On WorldWideWeb, Boston, USA (1995)
8. Davies, N.J., Weeks, R., Revett, M.C.: Jasper: Communicating Information Agents for WWW. Proc. Int. Conf. On WorldWideWeb, Boston, USA (1995)
9. Kobsa, A.: User Modeling: Recent work, prospects and hazards. In: Schneider-Hufschmidt, M., Kuhme, T., Malinowski, U. (eds): Adaptive User Interfaces: Principles and Practice. Elsevier Science Publishers B.V., Amsterdam (1993) 111-128
10. Martin, T.P., Arcelli Fontana, F.: Logic Programming and Soft Computing, Research Studies Press/ Wiley (1998)
11. McGrath, A.: The Forum. SIGGROUP Bulletin, Vol. 19, no. 3 (1998)
12. McTear, M.: User modelling for adaptive computer systems: A survey of recent developments. Artificial Intelligence Review, Vol. 7 (1993) 157-184
13. Nwana, H.S.: Software Agents: an overview. The Knowledge Engineering Review, Vol. 11, no. 3 (1996) 205-244
14. Rich, E.: User Modeling via Stereotypes. Cognitive Science, Vol. 3 (1979) 329-354
15. Sowa, J.F.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, Massachusetts (1984)
16. Wooldridge, M., Jennings, N.: Intelligent Agents: Theory and Practice. The Knowledge Engineering Review, Vol. 10, no. 2 (1995) 115-152
17. Zadeh, L.A.: Fuzzy Sets. Information and Control, Vol. 8 (1965) 338-353

URL References

1. BT Labs - The Forum: <http://www.bt.com/innovation/exhibition/forum/index.htm>