Service Type Specification through Conceptual Graphs*

Arno Puder
Department of Computer Science
University of Frankfurt
D–60054 Frankfurt, Germany
puder@informatik.uni-frankfurt.de

Ameneh Alireza
Deutsche Telekom AG, Research Centre
P.O. Box 100003
D–64276 Darmstadt, Germany
alireza@fz.telekom.de

August 19, 1996

Abstract

An open distributed environment can be perceived as a service market where services are freely offered and requested. Any infrastructure which seeks to provide appropriate mechanisms for such environments has to include some mediator functionality to bring together matching service requests and service offers. We propose a type specification notation based upon *conceptual graphs* to support the cognitive domain of application users. Furthermore, they help to cope with the problem of different extensions for the same intension of a service type. Conceptual graphs, which can easily be visualized, originate from a perception model in psychology. In this paper we present a formal model and a new standard interpretation for conceptual graphs.

Keywords: Open service markets, trading, service types, conceptual graphs.

1 Introduction

An open distributed environment can be perceived as a service market where services are offered and requested. Such environments require standardization of the underlying infrastructure in order to be able to scale up to a world wide heterogeneous network. One standardization effort is led by the *Telecommunications Information Networking Architecture* (TINA) Consortium; an international collaboration of telecommunication and IT companies aiming at defining and validating an open architecture for information and telecommunication services, which meets the future market needs (see [1]). The TINA architecture is based on distributed computing and object orientation. A crucial aspect of this architecture is the way services are *typed.* A type specification abstracts from specific internal details as it decouples the service usage from its implementation. Service types play an integral role during the mediation process. A software component known as a *trader* matches service offers and service requests based on their types.

A service type consists of an *intension* and an *extension.* The intension of a type is abstract in the sense that it serves as an abstract image of the properties of a service type. The intension is not to be confused with a physical representation, rather it is something we can think about. Any physical representation of an intension, such as a specification, is an extension of that intension. The extension provides a specific point of view of the intension of the service type. A service type specification (i.e. an extension) based on an intension is the only artefact which can be transported between users. Within an open service market, it is likely that two application users develop different extensions (or views) for the same intension (see left side of figure 1).

In this paper we present a type specification language which supports the cognitive domain of an application user. We decided to use conceptual graphs with a different standard interpretation. The advantage of conceptual graphs is that they stem from a cognitive model of perception and that they can be easily visualized. This paper is organized as follows: section 2 describes the cognitive perception model leading to conceptual graphs. Section 3 presents a formal model which we have devised in order to argue

*Appeared in *Workshop on Visual Reasoning at the 13th International Conference on Automated Deduction, Rutgers University, 1996*

about aspects of conceptual graphs. This section also describes the visualization of conceptual graphs. In section 4 we describe our method of mapping conceptual graphs to first order predicate formulas. This mapping differs from the standard mapping.

2 Cognitive perception model

Conceptual graphs stem from a psychological model of perception. The perception process generates an abstract image of reality which Sowa called a *conceptual graph* (see [3]). Objects or actions in the real world are mapped via input sensors to *sensory icons*. Based on background knowledge the sensory icons are matched against *percepts*. A *concept* is an interpretation of a percept. The perception process is divided into the following steps:

1. the perception process decomposes an external entity into a set of sensory icons.
2. all sensory icons are being compared with previously acquired percepts.
3. matching percepts are assembled to an approximation of the perception.
4. each percept corresponds to a concept which interprets the percept.
5. the concepts are connected to each other with relations which reflect the context in which the concepts occur.

A conceptual graph is a finite, connected, directed and bipartite graph, consisting of concept and relation nodes. A relation node connects two or more concepts. The direction of the arcs determines an ordering of the concepts which reflects the context of the nodes. The graph is bipartite because two concepts are always connected via a relation node. A conceptual graph is finite because of the limited storage capacity of the human brain (i.e. the set of sensory icons which are being produced by an external entity is always finite). A concept may have one or more instances which are described by that concept (e.g. a `HP-DESKJET` is an instance of concept `PRINTER`).

3 Formal definition of conceptual graphs

The textual representation of a conceptual graph is called the *linear notation*. In this notation concept nodes are surrounded with square brackets and relation nodes with round brackets. A simple conceptual graph in the linear notation would be `[`C_1`] -> (`R`) -> [`C_2`]`. The arrows denote the directed vertices of the graph. The conceptual graph can be transformed into a natural language sentence via the template *"The R of a C_1 is a C_2"*. Beside the linear notation there exists a graphical representation called the *display form*. In the display form a concept is denoted by a rectangle and a relation by a circle (see right hand side of figure 1). The syntax of conceptual graphs is currently being standardized by the *ANSI Technical Committee X3T2*.

We have adopted a formal notation to represent a conceptual graph in a mathematical notation. In the following $A \dot{\cup} B$ denotes set union with the intersection of A and B being empty. Furthermore, $Pot(M)$ denotes the powerset of a set M.

Definition 1 (Conceptual graph): The tupel $G = (N, L, F)$ is called a *conceptual graph* with

- C the set of *concept names*, R the set of *relation names*, IN the set of *instance names*.
- $I = Pot(IN)$ the set of all *instances*,
- $N \subseteq \mathbb{N}$ the finite set of node numbers,
- $L \subseteq N \times N$ the finite set of vertices,

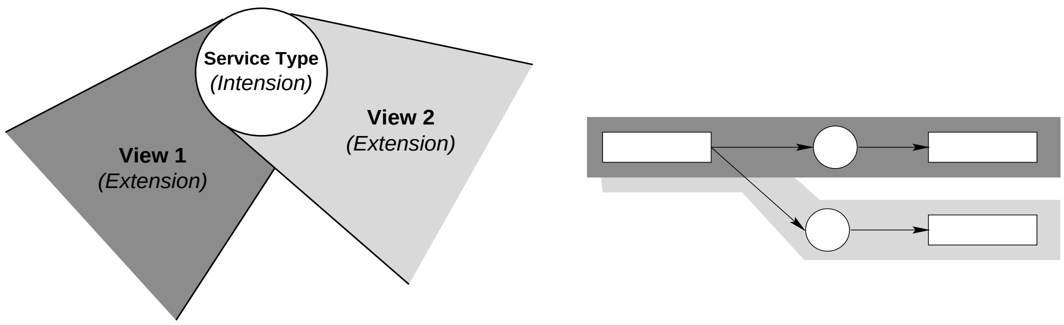

Figure 1: Alternative graphs as different extensions of a service type.

- $F \subseteq \mathbb{N} \times P$ the finite set of nodes, where each node consists of a node number and a property tupel $P = (C \times I) \cup R$,

if the following inductive construction principle applies:

1. Let (N, L, F) be a triple with

   (i) $N = \{n\}$, $n \in \mathbb{N}$, the number of the *root node*,
   (ii) $L = \emptyset$,
   (iii) $F = \{(n, p)\}$, $n \in N$, $p \in C \times I$,

   then $G = (N, L, F)$ is a conceptual graph.

2. If (N, L, F) is a conceptual graph and

   (i) $N' = N \,\dot{\cup}\, \{n', n''\}$, $n', n'' \in \mathbb{N}$, $n' \neq n''$ and
   (ii) $L' = L \,\dot{\cup}\, \{(n, n'), (n', n'')\}$, $n \in N$, $(n, p) \in F$ and
   (iii) $F' = F \,\dot{\cup}\, \{(n', p'), (n'', p'')\}$ with $p' \in R$ and $p'' \in C \times I$

   then $G = (N', L', F')$ is a conceptual graph.

3. If (N, L, F) is a conceptual graph and

   (i) $N' = N \,\dot{\cup}\, \{n'\}$, $n' \in \mathbb{N}$ and
   (ii) $L' = L \,\dot{\cup}\, \{(n, n')\}$, $n \in N$, $(n, p) \in F$, $p \in R$ and
   (iii) $F' = F \,\dot{\cup}\, \{(n', p')\}$ with $p' \in C \times I$

   then $G = (N', L', F')$ is a conceptual graph.

□

The set P includes all imaginable concept and relation nodes. The inductive construction principle guarantees that the root node as well as all leaf nodes of a conceptual graph are concept nodes. Furthermore, the construction principle guarantees that the graph is bipartite. Every node is associated with a unique number. This allows the same concept or relation to be referenced at different locations within the graph. The mapping for the formal specification of a concpetual graph to its display form is definite except for the graph layout. But there are several ways to convert the display form of a conceptual graph to its formal specification by choosing different node numbers. The following notations are defined with respect to the formal definition of conceptual graphs:

Notation 1: Let CG be the set of all conceptual graphs and $G = (N, L, F) \in CG$. Then the following operations are defined for conceptual graphs:

1. $root \in \mathrm{ABB}(CG, \mathbb{N})$, $root(G) =_{df} n'$ with $(n' \in N)(\forall\, n'' \in N)((n'', n') \notin L)$, determines the root node of a conceptual graph,

2. $succ \in \mathrm{ABB}(CG \times \mathbb{N}, Pot(\mathbb{N}))$, $succ(G, n) =_{df} \{n' \in N \mid (n, n') \in L\}$, the set of immediate successor nodes of a node n,

3. $pred \in \mathrm{ABB}(CG \times \mathbb{N}, \mathbb{N})$, $pred(G, n) =_{df} \begin{cases} n' & : \ (\exists (n', n) \in L) \\ undef. & : \ \text{otherwise} \end{cases}$,

   determines the immediate predecessor node of a node,

4. $node \in \mathrm{ABB}(CG \times \mathbb{N}, \mathbb{N} \times P)$, $node(G, n) =_{df} \{(n, p) \in F \mid n \in N,\ p \in P\}$, determines the corresponding node of a node number.

4 The semantics of conceptual graphs

The previous section dealt with the structure of conceptual graphs. We now briefly discuss their semantics. Every conceptual graph maps to a first order predicate formula through the definition of the operator Φ. We have changed Sowa's original definition of this operator to allow for the representation of multiple contexts within one conceptual graph. We view a conceptual graph as an approximation of a service type intension. Different users have different views of a service type. Each of these is represented by one alternative. Each branch of a conceptual graph is seen as a different extension of the service type. An *alternative* of a conceptual graph is formally defined as:

Definition 2 (Alternative of a conceptual graph): Let $G = (N, L, F)$ be a conceptual graph. The conceptual graph $G' = (N', L', F')$ is called an *alternative* of $G \Leftrightarrow$

1. $N' \subseteq N$, $root(G) \in N'$
2. $\forall\, n' \in N' : (pred(G, n') \in N')$
3. $\forall\, n' \in N' : ((node(G, n') = (n', (r'))) \Rightarrow (succ(G, n') \subseteq N'))$
4. $\forall\, n' \in N' : ((node(G, n') = (n', (c, i)) \wedge |\, succ(G, n') \,| \neq 0) \Rightarrow \exists_1\, n'' \in N' : (n'' \in succ(G, n')))$
5. $\forall\, n', n'' \in N' : ((node(G, n') = (n', (r')) \wedge pred(G, n') = pred(G, n'')) \Rightarrow n' = n'')$
6. $L' =_{df} \{(n, n') \in L \mid n, n' \in N'\}$
7. $F' =_{df} \{(n, p) \in F \mid n \in N'\}$

$\square$

The alternative of a conceptual graph is itself a conceptual graph. An alternative consists of the root node and a complete path to some leaves of the graph. Whereas from each concept node only one successor belongs to an alternative, for a relation node along the path all its successors belong to the alternative. The conceptual graph in figure 1 has a total of two alternatives (grey shaded branches of the graph). The next step in the definition of the Φ operator consists in assigning a predicate to each of the nodes of a conceptual graph, called the *corresponding predicate*.

Definition 3 (Corresponding predicate): Let $G = (N, L, F)$ be a conceptual graph. Each node in F of conceptual graph G is assigned a *corresponding predicate* through:

1. $(n, (c, \emptyset)) \in F$ is mapped to the corresponding predicate $c(x)$ with x as a unique free variable.
2. $(n, (c, i)) \in F$ is mapped to the corresponding predicate $c(i)$.

3. $(n, (r)) \in F$, $| succ(G, n) |= m$, is mapped to the corresponding predicate $r(x_0, x_1, \ldots, x_m)$ with arity $m + 1$. The variables $x_0, x_1, \ldots, x_m$ correspond to the parameters of the corresponding predicates of the neighboring concept nodes. x_0 matches the parameter of the corresponding predicate of the predecessor node $pred(G, n)$. Each x_i, $i \in \{1, \ldots, m\}$, matches the parameter of the corresponding predicate of one of the successor concept nodes $succ(G, n)$ of n.

□

The predicate names are identical to the names of the concept and relation names of the conceptual graph. All concept nodes are mapped to corresponding predicates with exactly one parameter. Those concepts with no associated instances are assigned a unique free variable as a parameter. Otherwise the parameter matches the instances associated with the concept. The number of parameters for the corresponding predicate of a relation node depends on the number of successors of that node. The parameters of the corresponding predicate for relation nodes reflect the context of the immediate predecessor and immediate successor nodes. All the parameters match with the ones for the appropriate concept nodes.

On the basis of the previous two definitions, the Φ operator which maps conceptual graphs to first order predicate formulas can now be defined:

Definition 4 (Φ operator): Let $G = (N, L, F)$ be a conceptual graph. The Φ operator maps G to a first order predicate formula obtained through the following construction rule: Let G' be an arbitrary but fixed alternative of G. The conceptual graph G' maps to a sub–formula consisting of the conjunction of the corresponding predicates belonging to G'. All sub–formulas of all alternatives of G are combined disjunctively to a first order predicate formula. $\Phi(G)$ is defined as the existential closure of that formula.

□

Although the Φ operator maps all conceptual graphs to first order predicate formulas, there is no corresponding inverse mapping. There are first order predicate formulas which have no corresponding conceptual graph (e.g. a formula containing a negation cannot be mapped to a conceptual graph). The Φ operator as defined above differs from Sowa's definition. Sowa does not make a distinction between different alternatives of a conceptual graph. According to his definition, the Φ operator combines all corresponding predicates conjunctively. The consequence is that his conceptual graphs can only specify precisely one extension of a service type.

5 Conclusion

We have proposed a specification technique for service types in an open service market based on conceptual graphs. The conceptual graphs support the cognitive domain of application users because they stem from a perception model and have an easy to understand graphical notation. A conceptual graph as a union of different extensions approximates the intension of a service type. This requires a different standard interpretation of conceptual graphs. We have implemented a prototype of a knowledge–based trader employing conceptual graphs (see [2]) using Java applets as a graphical user interface for the interactive specification of service types.

References

[1] F. Dupuy, G. Nilsson, and Y. Inoue. The TINA Consortium: Toward networking telecommunications information services. *IEEE Communications Magazine*, pages 78–83, November 1995.

[2] A. Puder. Introduction to the AI–Trader Project. http: //www.vsb.informatik.uni-frankfurt.de/ projects/ aitrader/ intro.html, Computer Science Department, Goethe University Frankfurt, 1995.

[3] J.F. Sowa. *Conceptual Structures, information processing mind and machine.* Addison–Wesley Publishing Company, 1984.