THE METANET. A KNOWLEDGE REPRESENTATION TOOL
BASED ON ABSTRACT DATA TYPES

Werner Dilger 1)
Wolfgang Womann 2)

ABSTRACT

The schema METANET is the specification of some kind of generalized semantic network as abstract data type. The theoretical basis of this work, namely $\Sigma$-termalgebras and rewrite rule systems, is introduced. It is described how the axiom schemas of the METANET arise by abstraction from real axioms and how axioms can be derived from axiom schemas by instantiation in turn, yielding a specification of a concrete sort of semantic networks. Some remarks on the properties and the use of METANET are passed.

INTRODUCTION

The concept of abstract data type is a well-defined means for the specification of data structures in large software systems. In the last ten years, large software systems have been developed in the area of AI and have become more and more important. It seems therefore a good idea to adopt mathematically well founded software engineering principles to such systems. We have made a step in this direction, defining one of the main data structures in AI, the semantic networks, which are closely related to frames, as abstract data types. In addition to the advantages that can be gained from such a definition in general, our approach has yielded a tool for testing and developing representation languages based on semantic networks. Our paper is divided in three sections. In the first and second ones, the theoretical basis of our work is given, namely $\Sigma$-termalgebras and rewrite rule systems. These parts are based on the work of Goguen, Thatcher, and Wagner [6], Dershowitz and Manna [3], Manna and Ness [10], Huet [8], and Knuth and Bendix [9].

In the last section the schema METANET is introduced as a generalized version of semantic networks, its specification as abstract data type is given, and some of its properties are described. This part is oriented on the type of networks introduced by Hendrix [7]. For other types see Findler [11], Barr and Feigenbaum [1], and Brachman [2]. More extended presentations of the METANET are given in Dilger and Womann [4, 5].

1) Institut für Informations- und Datenverarbeitung D-7500 Karlsruhe 1
   Sebastian-Kneipp-Str. 12

2) Nixdorf Computer AG, Berlinerstrasse 95, D-6000 München 40
**E-TERMALGEBRAS**

A **signature** is a pair \((S, E)\) of sorts and operation symbols respectively. To each operation symbol \(f\) a **functionality** is assigned, specifying its domain and range. It is denoted by \(f: s_1 \ldots \cdot s_k \rightarrow s\) or by \(f \in E_{w,s}\) where \(w \in S^*, s \in S, w = s_1 \ldots \cdot s_k\). An example is the signature \(\text{nat}\), consisting of the sort nat and the zero-element and the successor-function as operations. Thus we could write \(\text{nat} = \{(\text{nat}), (0, \text{SUCC})\}\), but we prefer to denote it in another form, including the functionalities of the operation symbols:

\[
\text{nat} \\
\text{sorts: nat} \\
\text{operation symbols: } O: \text{nat} \\
\text{SUCC: nat} \rightarrow \text{nat}
\]

A signature can be regarded as a framework for algebras which all have in some sense the same structure. For algebras, however, we need real data and real operations on these data, not only abstract symbols. Thus, given a signature \((S, E)\), an **algebra** is a pair \((A, E_A)\), where \(A = \{A_s | s \in S\}\) is a set of data sets and \(E_A\) is a set of \(k\)-ary operations \((k \geq 0)\) \(f_A: A_{s_1} \times \ldots \times A_{s_k} \rightarrow A_s\) with \(f \in E_{s_1 \ldots \cdot s_k, s}\), \(s, s \in S (i = 1, \ldots, k)\). It is immediately clear in which way an algebra "fits" to the underlying signature. Take as an example the algebra \(\text{NAT} = \{\{\text{N}\}, \{0_{\text{NAT}}, \text{SUCC}_{\text{NAT}}\}\}\).

From the operation symbols in \(E\) we can build terms in the usual way. For each sort \(s\) the set \(T_{E, s}\) of \(E\)-**terms** is defined by \(\Sigma_{s,s} \subseteq T_{E, s}\) and if \(f \in E_{s_1 \ldots \cdot s_k, s}\) and \(t_i \in T_{E, s_i} (i = 1, \ldots, k)\), then \(f(t_1, \ldots, t_k) \in T_{E, s}\). The \(\text{nat}\)-terms e.g. are 0, \text{SUCC}(0), \text{SUCC}(\text{SUCC}(0)), .... Now for the "fitting" of an algebra to the underlying signature it is required that the objects of the algebra have the same structure as the terms. This leads to the idea, that the terms could be themselves used as data sets for algebras.

Given a signature \((S, E)\), the \(E\)-**termalgebra** \(T_E\) is defined by (1) \(\{T_{E, s} | s \in S\}\) is the set of data sets, (2) the operations are \(f_{T_E}: T_{E, s_1} \times \ldots \times T_{E, s_k} \rightarrow T_{E, s}\) with \(f_{T_E}(t_1, \ldots, t_k) = f(t_1, \ldots, t_k)\), \(f \in E_{s_1 \ldots \cdot s_k, s}, t_i \in T_{E, s_i}, s, s \in S (i = 1, \ldots, k)\).

A \(E\)-termalgebra for a signature \((S, E)\) is an initial algebra, and that means, it has two nice properties: It is unique except for isomorphic algebras, and for each other algebra \(A\) there is exactly one homomorphism leading from the \(E\)-termalgebra to \(A\). Thus, the \(E\)-termalgebra can be taken as a representant of the class of algebras fitting to the signature, it describes the whole class in a **representation independent** way. In addition, all data of the \(E\)-termalgebra can be produced by operations, starting with the 0-ary operations. These two properties