SEMANTIC NETWORKS AS ABSTRACT DATA TYPES

Werner Dilger and Wolfgang Womann
Universitat Kaiserslautern
Fachbereich Informatik
Postfach 3049
D-6750 Kaiserslautern
FR Germany

ABSTRACT

A meta-description of semantic networks as abstract data types is given by a set of 35 production schemata.

With this schemata it is possible to specify all types of semantic networks as abstract data types. By instantiation of concrete types for nodes and edges as semantic primitives we get from the meta-description axiomatic definitions of arbitrary types of semantic networks as abstract data types.

The production schemata of the meta-description can be shown to be noetherian and confluent. Each term describing a semantic network can be reduced to an equivalent minimal generating expression.

I INTRODUCTION

Semantic networks are frequently used in NL-systems as a means for knowledge representation. There is a great number of different types of semantic networks which demonstrates their qualification for the purpose of knowledge representation. In (Barr and Feigenbaum, 1981), knowledge representation is conceived as data structures together with interpretative operations. We adopt this interpretation for semantic networks and describe them by means of the theory of abstract data types (Goguen, Thatcher and Wagner, 1976).

We start from a very general and multipurpose type of semantic net: the metanet. In the description of the metanet we abstract from the semantic primitives representing them by variables.

The metanet consists of a set N of nodes and a set E of edges. Every node (edge) belongs to a subset of N (E), which is a representation of a node (edge) type NT_i (ET_e). The number of node and edge types is finite.

We give an axiomatic definition of the metanet using general operations on networks. The set of 35 axioms is a rewrite system which is shown to be noetherian and confluent. The metanet is correct, in the sense that terms built of operations and individuals from the domain of semantic networks,i.e. describing networks can be reduced into an equivalent minimal generating expression which is not further reducible and which contains only constructors.

II SPECIFICATION OF A METANET

Because of the abstraction from concrete types for nodes and edges, we can define operations on nodes and edges independently from their types. So we are able to examine general characteristics of semantic networks. This leads to production schemata instead of productions of the rewrite system, i.e. the axiomatic definition can be conceived as a meta-description of a semantic net, from which the axiomatization of concrete nets can be derived by instantiation of concrete types for all occurrences of type variables. For example, node types could be "concept" and "instance" edge types could be "isa", "subset-of", "object-of", "agent".By means of this abstraction we get an axiomatic definition of the abstract data type "metanet" by 35 axioms which are easy to surrey. Some of the axioms are provided with conditions, enclosed in {...}, wich restrict the applicability of the productions.

We now present the specification of the abstract data type "metanet". This specification is based on the abstract data types "boolean" and "set", "set" consists of the operations: EMPTYSET, ISEMPTYSET, INSERT, DELETE, ISIN.

Abbreviations:

NT = $\{NT_1, NT_2, ..., NT_n\}$ set of node types.
ET = $\{ET_1, ..., ET_m\}$ set of edge types.
n1_i represents a node named n1 with type N_i.
[n1_i,n2_j] represents an edge between the nodes n1 and n2.
ADDNODE.i represents the operation of adding a node with type NT_i to the metanet.
DELETEDGE.e represents the operation of delation of an edge of type ET_e.

```
METANET = SET +
    sorts:  NT_1,...,NT_n, ET_1,...,ET_m
    operationsymbols: ∀ i ∈ NT,∀ e ∈ ET
EMPTYNET:                                → metanet
ADDNODE.i:          metanet NT_i         → metanet
INSERTEDGE.e:       metanet ET_e         → metanet

DELETENODE.i:       metanet NT_i         → metanet
DELETEEDGE.e:       metanet ET_e         → metanet

ISEMPTY:            metanet              → boolean
ISNODE:             metanet NT           → boolean
ISEDGE.e:           metanet ET_e         → boolean

NODES:              metanet              → SET(NT)
EDGES.e:            metanet              → SET(ET_e)
NEIGHBOURNODES:     metanet              → SET(NT)

axioms: ∀ m ∈ metanet   ∀ e,f ∈ ET
                        ∀ i,j ∈ NT
        ∀ n_i,n1_i,n2_i ∈ N_i
        ∀ n1_j,n2_j ∈ N_j
        ∀ n2_k,n3_k ∈ N_k
        ∀ n4_l ∈ N_l

   (*axioms concerning the boolean opera-
     tion "ISEMPTY"*)
1)  ISEMPTY(EMPTYNET) → TRUE
2)  ISEMPTY(ADDNODE.i(m,n_i)) → FALSE
3)  ISEMPTY(INSERTEDGE.e(m,[n1_i,n2_j])) &
    {ISNODE(m,n1_i)∧ISNODE(m,n2_j)}→ FALSE
    (*if an edge will be inserted and one
      or both nodes are not in the net,
      this is an error*)
4)  INSERTEDGE.e(m,[n1_i,n2_j])&{NOT ISNODE
    (m,n1_i)∨NOT ISNODE(m,n2_j)→ Error
                                     metanet
    (*axioms concerning the operation
      "NODES" which provides the set of all
      nodes in the net*)
5)  NODES(EMPTYNET) → EMPTYSET
6)  NODES(ADDNODE.i(m,n_i)) →
    INSERT(NODES(m),n_i)
7)  NODES(INSERTEDGE.e(m,[n1_i,n2_j])) &
    {ISNODE(m,n1_i)∧ISNODE(m,n2_j)}→NODES(m)
    (*axioms concerning the deletion of
      nodes*)
8)  DELETENODE.i(EMPTYNET,n_i) → EMPTYNET
9)  DELETENODE.i(ADDNODE.i(m,n_i),n_i) &
    {ISEMPTYSET(NEIGHBOURNODES(m,n_i))} →
    DELETENODE.i(m,n_i)
10) DELETENODE.i(m,n_i)&{NOT ISEMPTYSET
    NEIGHBOURNODE(m,n_i))} → Error
                                  metanet
11) DELETENODE.i(ADDNODE.j(m,n2_j),n1_i) &
    {n2_j≠n1_i}→ADDNODE.j(DELETENODE.i
    (m,n1_i),n2_j)
12) DELETENODE.i(INSERTEDGE.e(m,[n1_j,n2_k]),
    n_i)&{n_i≠n1_j∧n_i≠n2_k∧ISNODE(m,n1_j)∧
    ISNODE(m,n2_k)}→INSERTEDGE.e(DELETE-
    NODE.i(m,n_i),[n1_j,n2_k])
13) DELETENODE.i(INSERTEDGE.e(m,[n1_i,n2_j]),
    n1_i)&{ISNODE(m,n1_i)∧ISNODE(m,n2_j)}
    → Error
           metanet
14) DELETENODE.i(INSERTEDGE.e(m,[n2_j,n1_i]),
    n1_i)&{ISNODE(m,n2_j)∧ISNODE(m,n1_i)}
    → Error
           metanet

    (*axioms concerning the existence of a
      node in the net*)
15) ISNODE(EMPTYNET,n_i) → FALSE
16) ISNODE(ADDNODE.i(m,n_i),n_i) → TRUE
17) ISNODE(ADDNODE.i(m,n1_i),n2_j) &
    {n1_i≠n2_j} → ISNODE(m,n2_j)
    (*axioms concerning the operation
    "EDGES.e" which provides the set of all
    edges of type e*)
18) EDGES.e(EMPTYNET) → EMPTYSET
19) EDGES.f(INSERTEDGE.e(m,[n1_i,n2_j]))&
    {ISNODE(m,n1_i)∧ISNODE(m,n2_j)}
    → EDGES.f(m)
20) EDGES.e(INSERTEDGE.e(m,[n1_i,n2_j]))&
    {ISNODE(m,n1_i)∧ISNODE(m,n2_j)}
    → INSERT(EDGES.e(m),[n1_i,n2_j])
21) EDGES.e(ADDNODE.i(m,n_i))→EDGES.e(m)
    (*axioms concerning the deletion of
      edges*)
22) DELETEEDGE.e(EMPTYNET,[n1_i,n2_j])
    → EMPTYNET
23) DELETEEDGE.e(INSERTEDGE.e(m,[n1_i,
    n2_j]),[n1_i,n2_j])&{ISNODE(m,n1_i) ∧
    ISNODE(m,n2_j)} → m
24) DELETEEDGE.e(INSERTEDGE.f(m,[n1_i,
    n2_j]),[n3_k,n4_l]) &
    {(n1_i≠n3_k∨n2_j≠n4_l∨e≠f)∧ISNODE
    (m,n1_i)∧ISNODE(m,n2_j)}→INSERTEDGE.f
    (DELETEEDGE.e(m,[n3_k,n4_l]),
   [n1_i,n2_j])
25) DELETEEDGE.e(ADDNODE.i(m,n1_i),
    [n2_j,n3_k])→ADDNODE.i(DELETEEDGE.e
    (m,[n2_j,n3_k]),n1_i)
    (*axioms concerning the existence of
      an edge of type e in the net*)
26) ISEDGE.e(EMPTYNET,[n1_i,n2_j])→FALSE
27) ISEDGE.e(INSERTEDGE.e(m,[n1_i,n2_j]),
    [n1_i,n2_j )&{ISNODE(m,n1_i)∧
    ISNODE(m,n2_j)} → TRUE
28) ISEDGE.e(INSERTEDGE.f(m,[n1_i,n2_j]),
    [n3_k,n4_l])&{(n1_i≠n3_k∨n2_j≠n4_l∨e≠f)
    ∧ISNODE(m,n1_i)∧ISNODE(m,n2_j)}
    → ISEDGE.e(m,[n3_k,n4_l])
    (*axioms concerning the operation
      "NEIGHBOURNODES" which provides the
      set of all adjacent nodes of a node*)
29) NEIGHBOURNODES(EMPTYNET,n_i)→EMPTYSET
30) NEIGHBOURNODES(ADDNODE.i(m,n_i),n_i)&
    {ISNODE(m,n_i)}→NEIGHBOURNODES(m,n_i)
31) NEIGHBOURNODES(ADDNODE.i(m,n_i),n_i)&
    {NOT ISNODE(m,n_i)} → EMPTYSET
32) NEIGHBOURNODES(ADDNODE.i(m,n1_i),n2_j)
    &{n1_i≠n2_j} → NEIGHBOURNODES(m,n2_j)
33) NEIGHBOURNODES(INSERTEDGE.e(m,[n1_i,
    n2_j]),n1_i)&{ISNODE(m,n1_i) ∧
    ISNODE(m,n2_j)}→INSERT(NEIGHBOURNODES
    (m,n1_i),n2_j)
34) NEIGHBOURNODES(INSERTEDGE.e(m,[n1_i,
    n2_j]),n2_j)&{ISNODE(m,n1_i)∧ISNODE
    (m,n2_j)}→INSERT(NEIGHBOURNODES(m,n2_j)
    n1_i)
35) NEIGHBOURNODES(INSERTEDGE.e(m,[n1_i,
    n2_j]),n3_k)&
    {n3_k≠n1_i∧n3_k≠n2_j∧ISNODE(m,n1_i)∧
    ISNODE(m,n2_j)}→NEIGHBOURNODES(m,n3_k)
```

III PROPERTIES OF THE METANET

The following ideas are incorporated in the METANET:

The operations have no side effects.
Example: Before a node can be deleted, all its incident edges have to be detted. Else, if we would try to delete a node prior to the deletion of its incident edges, we had to delete its edges together with itself to avoid mistakes.

- For each sorts there is an error element $Error_s$ for the handling of exceptions (Goguen, Thatcher and Wagner, 1976).
  Example: Axiom 4: If an attempt is made to insert an edge between nodes nl and n2 and one of them does not exist in the network, then the insert operation is evaluated to $Error_{network}$.
- The operations are assumed to be strict, i.e. if an error element occurs somewhere the whole expression is evaluated to $Error_s$ (Goguen, Thatcher and Wagner, 1976).

The production schemata of the metadescriptioncan be proven as noetherian by one of the techniques given in (Dershowitz and Manna, 1973), (Manna, Ness and Vuillemin, 1973). It is easy to find weights for the individual and operation symbols in the production schemata, such that the technique of (Manna, Ness and Vuillemin, 1973) can be applied. Confluence can be shown by means of the superposition algorithm of (Knuth and Bendix, 1969). Both properties make it possible that every term describing a metanet is reducible to an equivalent minimal generating expression (Womann, 1983), i.e. a term consisting only of operations which are constructors.

Metanets can be extended by nodes of higher type representing semantic networks. This extended metanet is specified in the 9ame way as the basic metanet. (Womann, 1983) gives a specification of an extended metanet which corresponds to the partitioned networks (Hendrix, 1979). The axioms are similar to those given here, except that all operations have an additional argument concerning spaces and there are some further axioms for the vistas.

IV A SIMPLE EXAMPLE

We give an example of a concrete network.

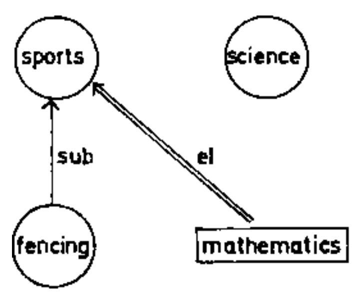
node types = 2 (Concept, Instance).
edge types = 2 (subset_of, element.of).
Con-Concept,Ins-Instance,sub^subset.of, el^eleraent.of.

The full specification of this semantic network as' instantiation of the metanet consists of 78 axioms. We omit the fully instantiated specification, but cf. (Womann, 1983). Rather we present a subset of the axioms, which is used for the reduction of a sample term.

Instantiated axioms for the example:

```
i1) DELETEEL (INSERTSUB (m,[ nl_con,n2_con]),
    [ n3_ins,n4_con])&
    {(nl_con±n3__insvn2_con*n4_convEL*SUB)
    AISNODE(m,nl_con)AISNODE(m,n2_con)}-
    INSERTSUB(DELETEEL(m,[ n3_ins,n4_con]),
    [ n1 con,n2_con])
    (*instance of axiom 24*)
i2) DELETEEL (ADDCON (m,nl_con),[n2_ins,
    n3_con J)-ADDCON(DELETEEL(m, [n2_ins,
    n3_con]),nl_con)
    (*instance of axiom 25*)
i3) DELETEEL(INSERTEL(m,[nl_ins,n2_con]),
    [nl_ins,n2_con])&{ISN0DE(m,nl_ins)A
    ISN0DE(m,n2__con) }->m
    (♦instance of axiom 23*)
```

The concrete network is:

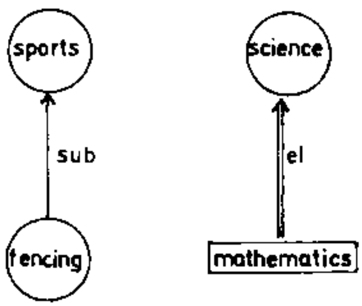


The term T_O describes this semantic net

```
TO := INSERTSUB(ADDCON(INSERTEL(ADDCON
        (ADDINS(ADDCON(EMPTYNET,science),
                mathematics),
          sports),
                                        [mathematics
                                         sports]),
                                    fencing),
                          [fencing,sports])
```

Because we made a mistake classifying mathematics as an instance of sports, we want to adjust this. We extend term T_O adding two new operations. Then we delete the element_of-edge betweeen the nodes for mathematics and sports and we add an element_of-edge between mathematics and science. The extended term is named T_1.

T_1 := INSERTEL(DELETEEL(T_o,[mathematics, sports]),[mathematics,science])



Now we reduce T_1 by means of the instantiated axioms to an equivalent term which is a minimal generating expression (the reader is invited to check what "...|m" means whereever it occurs).

T_1

↓ i_1 with σ={fencing|n1_con,sports|n2_con, mathematics|n3_ins,sports|n4_con, ...|m}

T_2 := INSERTELEMENT(INSERTSUB(DELETEEL (ADDCON(INSERTEL(ADDCON(ADDINS (ADDCON(EMPTYNET,science),mathematics),sports),[mathematics,sports]) fencing),[mathematics,sports]), [fencing,sports]),[mathematics, science])

↓ i_2 with σ={fencing|n1_con,mathematics| n2_ins,sports|n3_con,...|m}

T_3 := INSERTEL(INSERTSUB(ADDCON(DELETEEL (INSERTEL(ADDCON(ADDINS(ADDCON (EMPTYNET,science),mathematics), sports),[mathematics,sports]), [mathematics,sports]),fencing), [fencing,sports]),[mathematics, science])

↓ i_3 with σ={mathematics|n1_ins,sports| n2_con,...|m}

T_4 := INSERTEL(INSERTSUB(ADDCON(ADDCON (ADDINS(ADDCON(EMPTYNET,science), mathematics),sports),fencing), [fencing,sports]),[mathematics, science])

T_4 is a minimal generating expression. No further axioms are applicable. T_4 is equivalent to T_1.

V CONCLUSION

It is difficult to define concrete semantic networks as abstract data types because the specification of such a data type would become too complex and hard to survey. On the other side it is desirable to have semantic networks defined as abstract data types, because the structure of an AI-system depends heavily on the realization of its knowledge representation technique. If this technique is defined by means of abstract data type theory, the whole system can be structured as a hierarchy of abstract data types. The metanet is a means to overcome this problem. The metanet is defined as a specification of an abstract data type and each semantic network can be described as an instance of the metanet.

At present, we are implementing a program which yields instantiations of the metanet from concrete semantic primitives. Together with an interpreter for the axioms which reduces terms to minimal generating expressions we would have a tool for development and testing applicability of semantic networks and for the comparison of different semantic networks on the same level.

REFERENCES

[1] Barr A. and Feigenbaum E.A. (ed): "The Handbook of Artificial Intelligence". Stanford, 1981.

[2] Brachman R.J.: "What's in a concept: Structural foundations for Semantic Networks". Int. J. Man-Machine Studies 9 (1977) 127-152.

[3] Dershowitz N. and Manna Z.: "Proving Termination with Multiset Orderings". Comm. of the ACM 16 (1973) 465-476.

[4] Goguen J.A., Thatcher J.W., Wagner E.G.: "An initial algebra approach to the specification, correctness, and implementation of abstract data types". in Yeh R.T. (ed.): Current Trends in Programming Methodology, Volume IV, Data Structuring, 1976.

[5] Hendrix G.G.: "Encoding knowledge in Partitioned Networks". in Findler N. (ed): Associative Networks. New York, 1979.

[6] Knuth E. and Bendix P.B.: "Simple word problems in universal algebras". in Leech J. (ed.): Computational Problems in Universal Algebras. Oxford, 1969.

[7] Manna Z., Ness S., Vuillemin J.: "Inductive Methods for Proving Properties of Programs". Comm. of the ACM 16 (1973) 491-504.

[8] Schubert, Goebel, Cercone: "The Structure and Organisation of a Semantic Net for Comprehension and Inference". in Findler N., see [5].

[9] Womann, W.: "Abstrakte Datenstrukturen für die Wissensrepräsentation mittels semantischer Netze". Diplomarbeit, Universität Kaiserslautern, 1983.