GENERATING HIERARCHICAL SEMANTIC NETWORKS FROM NATURAL LANGUAGE DISCOURSE*

Camilla Schwmd
Teehnische Universitat, Munchen, Germany

Abstract

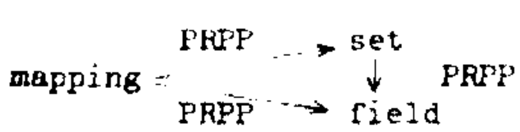
The following paper contains a description of a computer program that constructs hierarchical semantic networks from natural language texts in simulating completely and precisely the meaning of the input text. The program works with a formal grammar describing sentences syntactically and a formal semantics transducing texts to networks dependent on their syntactic description. The networks' nodes have concepts or again networks, as their values. The networks' edges are many-place relations among the concepts respectively networks. They are realized as reference structures. The program in its fietual state works on domains which art to comprehend semantically rather well: A network has been constructured automatically for the area of general topology from a compendi urn's definitions (N. Bourbaki. Elenients of Mathcmatics. General topology). Another one is being constructed for the area of computer science.

Introduction

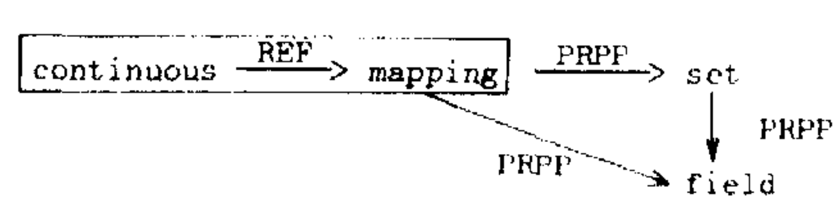
The ultimate goal of the research described here is to develop a computer program that could construct a hierarchically organized semantic network from natural language input text. This network concists of a set of concepts and a set of many-place relations over this set. Examples of these relations are the relation between a term and its su~perordinated term, called SUP, e.g. "mapping" SUP "homeomorphism", or the relation REF between an adjective and each noun that could be modified by this adjective e.g. "commutative" REF "group"- Relations among terms; can expand to relations among terms and networks. For example, the relation *REF* holds between the adjfctive "continuous" and the noun "mapping". This can be described by the network

(*) continuous --REF--> mapping.

Between the nouns "mapping", "set", and "field" there holds the three place relation PRPP giving rise to the network

(⁑) 

Now it is possible to formulate a rule that replaces the value of the one node "mapping" in this last network (⁑) by the first network (*) to obtain the relation PRPP between the network (*) and the single terms "set" and "field". The resultant network



is called hierarchical.

The mathematical structure of such a network is a hierarchical graph (see [3"!). To construct, such a network, one can use hierarchical network generation rules as described in [7]. The network obtained from an input text should be able to reproduce the meaning of the text completely and precisely, that is to say, it should be the semantics of the appropriate text. So the coordination of natural language text to a network, being realized by a computer program, represents what we mean by *understanding* the text: the fitting computer program *underetands* natural language discourse. Complete reproduction of discourse by a network means that no information is lost on the transduction of the text into the network and consequently that it should be possible to reconstruct the primary text, from the network once generated. But the reconstruction is not one-valued. There could be more than one text being reproduced from a network, for there might occur different texts to have the same meaning, and these must correspond to the same network. So generally we obtain one single network for more than one text. Precise reproduction of discourse by a network means that for each text there could be at most one network into which that text is transduced. So the coordination texts-networks is a non-injective mapping. This mapping is realized in the following way: There has been written a formal grammar describing arbitrary natural language texts by which a phrase structure tree is built for each sentence (see C11). Trie semantic rules are formulated dependent on the syntax trees following step-by-step transduction of a phrase structure tree into a semantic network fragment*. The concatenation of all the fragments obtained by a text is the final network. In the following we will demonstrate (for an example sentence) the effect of the program, that is to say, how it understands a text. The example sentence stems from a mathematical text out of general topology.

Definition. In a topological space X, a fundamental system of neighbourhoods of a point x is any set S of neighbourhoods of x such that for each neighbourhood V of x there is a neighbourhood $W \in S$ such that $W \subset V$.

For this definition the program finds out:

1. What concept is defined?
P. What other concepts do help to accomplish the definition?
.3 Which conceptual relations do exist among these concepts? (see Figure 1).

The names of the concepts (given to them in the text) are kept for further references. Later on they can be replaced by the concepts they design. The prepositional relation OF holds between a concept and its prepositional supplement. The relations EPS and CONT belong to the mathematical signs $\in$ and $\subseteq$.

+ This work has been supported by the SFB 49 of the Deutsche Forschungsgemeinsehaft.

* see ([2])

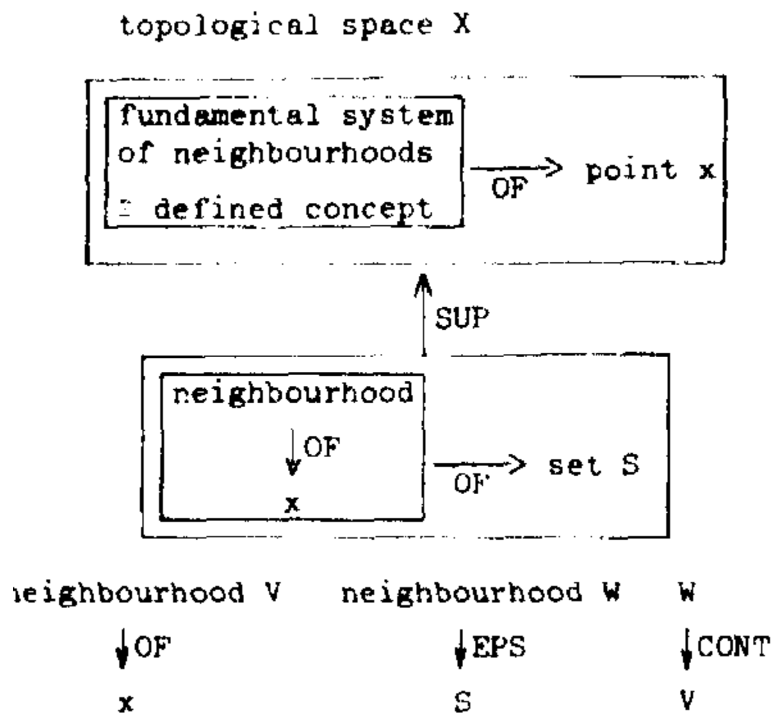


Figure 1

Hierarchical networks

A hierarchical network consists of a set of concepts and a set of many-place relations among the concepts. The following conceptual relations have been sufficient to describe the semantics of mathematical texts.

N denotes the set of the nouns.
A denotes the set of the adjectives.
Av denotes the set of the adverbs.

Superordinated and subordinated terms:
For each noun defined in a text exists a superimposed term. Adjectives not always possess superimposed terms. Nouns that have no superordinated term are called primes (e.g. "set", "class", "element"). Supterm 0, specified for a term B in the thesaurus, must always be a directly superordinated one, i.e. there is no other superordinated term for B subordinated for 0 .
This direct super-relation is noted SUP_0.

$SUP_0 \subseteq (N \times N) \cup (A \times A)$

SUP is the transitive closure of SUP_0 . A term can have more than one superimposed term, e.g.:

injective SUP bijective
surjective SUP bijective.

Referential area for adjectives:
For every adjective C there is at least one noun B that can have the property described by the adjective. B is called referential term for C - B is part of the referential area of C , i.e. the set of nouns to that C is applicable. The referential relation is noted REF .

$REF \subseteq A \times N$.

e.g. "continuous" REF "mapping".

Prepositional relations:
Terms that occur in propositional adjuncts of a term B are said to be in prepositional relation with B . The prepositional relation cannot be specified quite exactly. We can state, however, that a term is modified by its prepositional adjunct. The prepositional relation is noted PREP.

$PREP \subseteq (N \times N) \cup (A \times A)$

e.g. "order" PREP "set" or "continuous" PREP "set".

We also have a three place prepositional relation noted PRPP. $PRPP \subseteq N \times N \times N$. e.g. "mapping" PRPP "set" PRPP "set".

Antonymy:
This relation exists between pairs of terms such as "greater" and "less".
It is noted ANT .

$ANT \subseteq (A \times A) \cup (N \times N)$.

According to our hitherto knowledge, in all pairs of terms for which ANT takes place, ANT is based on the antonymy of "greater" and "less".

Synonymy:
Two terms are called synonymons if they are identical in meaning. Synonymy is noted SYN .

$SYN \subseteq (A \times A) \cup (N \times N)$.

Adverb-adjective relation:
As mentioned above this relation holds between an adverb and an adjective which is modified by the adverb.
This relation is noted AVAJ

$AVAJ \subseteq Av \times A$.

For each pair this relation is given in the thesaurus entry of the adjective.

The verbal relation CONT :
This relation holds between the subject and the object of a sentence with the verb contain.

$CONT \subseteq N \times N$.

For example: The set X contains any point x . Here we have 'set' CONT 'point'. CONT refers to the mathematical connective $\subseteq$.

There exist connections among the relations which can be used to store them in a manner to save considerable amounts of computer storage place. In the following, the set of the superimposed terms of term X is noted OBER(X) .

The class of elements synonymous to a term X is called the synonymy class of X , noted SYN(X) . As there is at most one element antonymous to a term X , the antonymy class ANT(X) of a term only consists of this element and of X .

The set of terms occuring in the definition of a term X is noted DEF_X .

Interrelations between relations:
E1 SUP is an order relation.
E2 If a noun B is a referential term for an adjective C , then each subterm M of B belongs to the referential area of C , except C or a term antonymous to C belongs to the definition of M or of a term between B and M .

$[C \text{ REF } B \wedge B \text{ SUP } M \wedge [\forall X[(B \text{ SUP } X \wedge X \text{ SUP } M) \Rightarrow \neg(C \in DEF_X \vee ANT(C) \in DEF_X)]]] \Rightarrow C \text{ REF } M$.

For example: "commutative" REF "semigroup" and "semigroup" SUP "group" ⇒ "commutative" REF "group".

The additional condition excludes relations as "isomorph" REF "antiisomorphism" or "injective" REF "bijection" or "bijective" REF "bisection".

E3 If a noun C is a prepositional adjunct for a term B , each subterm M of C is a prepositional adjunct of B .

(B PREP C A C SUP M) -> B PREP M .

E.g.: "element" PREP "Set" and "set" SUP "field" -> "element" PREP "field".

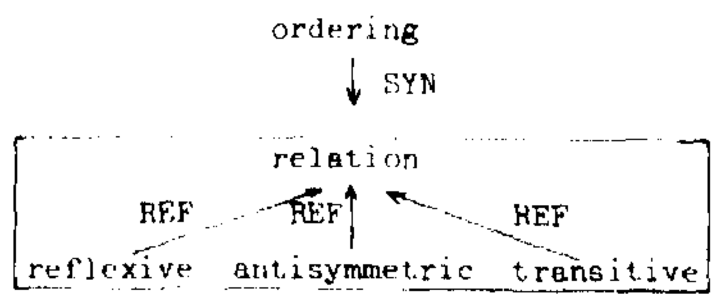
E4 If two nouns L,M are prepositional adjuncts for a term B , then each subterm 0 of L and each subterm P of M is a prepositional adjunct for B .

(B,L,M) G PRPP A L SUP 0 A M SUP P -> (B,0,P) e PRPP.

For example: ("mapping", "set", "set") e PRPP and "set" SUP "group" «+ ("mapping", "group", "set") € PRPP and ("mapping", "set", "group") e PRPP and ("mapping", "group", "group") e PRPP.

E5 SYN is an equivalence relation.

Sample network:



To describe the semantics of computer science texts, we must add more types of relations, e.g. come verbal relations, that is the relations that hold between a verb, its possible subjects and its possible object and prepositional supplements.

Syntactic and semantic description of texts

The texts which are to be transduced into semantic networks are analyzed by context-sensitive rules of a formal grammar G , together with a set of transformation rules T and a lexicon LEX . G is a type-2-language. G = (A, P, PEF). A is the alphabet consisting of grammatical categories like VP.NP (verbal phrase, nominal phrase) and lexical categories like prep, art. The lexical categories are the terminals A„ of the grammar. P is a set of production rules and DEF G A the starting symbol. LEX is a set of type-2-productions. For each lexical category *X* e AT there is at least one production rule of the form *X* ::= x . x is then called lexical entry for *X*. For example: art ::= the € LEX . "the" is a lexical entry fort *art*.

The following sample grammar is a typical part of the grammar describing the definitions of "Elements of mathematics" by N.Bourbaki. It is written using BNF-notation. Terminals are italicized. In some rules subscripts have been used to distinguish between occurrences of identical nonterminals. These subscripts are irrelevant for syntax. They will be needed later by the semantic rules.

(1) DEF ::= $\{S3\}_0$ $\{S\}_1$
(2) S ::= PP S1
(3) S ::= PP S2
(4) S1 ::= C *be* NP|NP *be called* C
(5) S2 ::= NP *be said to be* PREDD *if* SS1
(6) S3 ::= *let Nm be* NP_1 $\{\{\text{and}\}_0^1$ *Nm* $NP_2\}_0$
(7) C ::= DNB
(8) C_1 ::= C_2 PP
(9) DNB ::= $\{art\}_0^1$ ND
(10) ND_1 ::= AP ND_2
(11) ND ::= *NB*
(12) AP_1 ::= *adv* AP_2
(13) AP ::= *AB*
(14) NP ::= NPO $\{SS\}_0^1$
(15) NP ::= NPO *such that* SS1
(16) NPO ::= NP1
(17) NPO ::= NP1 PP
(18) NP1 ::= *art* N
(19) N_1 ::= AP N_2
(20) N ::= *NB*
(21) N ::= *NB Nm*
(22) PP ::= *prep* NPO
(23) PREDD ::= DAB
(24) PREDD ::= DAB PP
(25) DAB ::= AD
(26) AD_1 ::= *adv* AD_2
(27) AD ::= *AB*
(28) SS ::= NP VP
(29) VP ::= *verb* NP
(30) SS1 ::= SS
(31) SS1 ::= *for each* NP_1 *there be* NP_2 *such that* SS

Comments on the grammar:

ad(1). A definition DEF consists of at least one sentence. It can begin with a sentence S3 like "let X be a topological space and A a subset of X" see (5).

ad(2),(3). Every sentence can begin with a prepositional phrase, e.g. fig. 2.

ad(4). Sentences like "A topological space is a set endowed with a topological structure", and fig. 1. The lexical entries of *be* are all morphological paraphrases of the verb "be" i.e. "is", "are", etc.. In general by this construction nouns are defined.

ad(5). Sentences like "an equivalence relation R on a topological space X is said to be open if the canonical mapping of X onto X/R is open". In general by this construction adjectives are defined.

ad(7)-(11). C serves to expand the nominal phrase DNB which contains the defined concept ND . It is possibly followed by a prepositional phrase PP or preceded by one or more adjective clauses AP .

ad(12),(13). An adjective can be modified by one or more adverbs *adv*.

ad(14),(15). A NP can govern an embedded sentence SS or SS1 , e.g. a relative clause.

ad(16)-(21). A NP can have a prepositional supplement PP . It consists of an article *art*, possibly followed by one or more adjectives and a noun *NB* possibly followed by its proper name *Nm* .

ad(22). Prepositional clause. The lexical entries for *prep* are all prepositions occuring in a text.

ad(23)-(27). PREDD serves to expand a defined adjective DAB . It is possibly followed by a PP or preceded by one or more adverbs.

ad(28)-(31). Sentence constructions as usual. Embedded sentences like in the example in Fig. 1 are very frequent in mathematical texts.

T consists of three types of transformations rules:

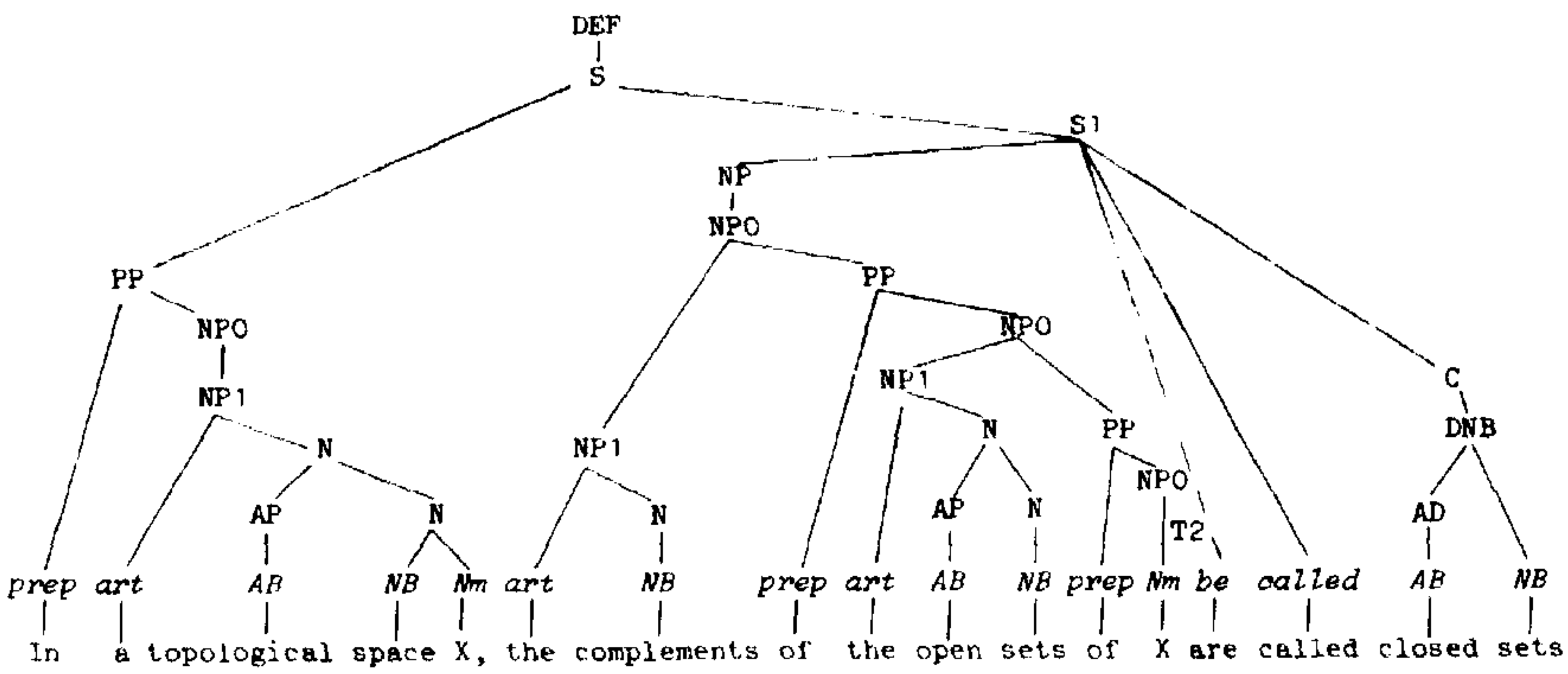


Fig. 2. Sample phrase structure tree. Ti by the side of an arc means that the appropriate reduction was executed by a transformational rule of type i .

1. Transformation of a NP to a relative pronoun.
2. Transformation of a NP to a proper noun.
3. Transformation of an embedded sentence to a partipicial sentence.

We will not list transformations here in detail, for they are not relevant for the meaning of sentences. They only generate surface structures from deep structures. LEX can be seen in detail in the program data.

The set of concepts occurring in a definition together with the hierarchical graph they form is built up in a step-by-step manner corresponding to the phrase structure of the definition. This can be done by assigning *attributes* to the nonterminal symbols and semantic rules to the production rules. Each attribute a corresponds to a set of *values* V_a . The semantic rules define all of the attributes of a nonterminal in terms of the attributes of its immediate descendants, so ultimately values are defined for each attribute.

For our purposes, we have used a set of attributes A = {d, c, g, lex, n} where

V_d is the power set of the set of all concepts. The value of d within a definition will be the set of concepts defined.

V_c is the power set of the set of all concepts. The value of c within a definition will be the set of all concepts occurring in the definition.

V_g is the set of hierarchical graphs. The value of g within a definition will be the hierarchical graph that means this definition.

V_{lex} is the union set of all sets of lexical entries of G .

V_n is the set of relation names. The value of n within parts of a definition is an arc label belonging to the hierarchical graph being built up.

The attributes correspond to the nonterminal symbols in the following way:

g belongs to {DEF, S, S1, S2, S3, C, CNB, ND, AP, NP, NPO, NP1, N, AD, PP, PREDD, DAB, SS, VP, SS1)

c belongs to the same symbols as g .

d belongs to {DEF, S, S1, S2, C, DNB, ND, PREDD, DAB, AD}

lex belongs to *{NB, AB, adv, verb, prep)*

n belongs to {VP, PP}.

Now the grammar may be augmented so, that semantic rules are given for each rule of the syntax.

(A1) $g(DEF) = g(S3) \cdot^{*} g(S)$
$d(DEF) = d(S)$
$c(DEF) = c(S3) \cup c(S)$

(A2) $g(S) = g(PP) \cdot g(S1)$
$d(S) = d(S1)$
$c(S) = c(S1)$

(A3) $g(S) = g(PP) \cdot g(S2)$
$d(S) = d(S2)$
$c(S) = c(S2)$

(A4) $g(S1) = g(C) \cdot g(NP)$
$d(S1) = d(C)$
$c(S1) = c(C) \cup c(NP)$

(A5) $g(S2) = g(NP) \cdot g(PREDD) \cdot g(SS1$
$d(S2) = d(PREDD)$
$c(S2) = c(NP) \cup c(PREDD) \cup c(SS1$

(A6) $g(S3) = g(NP_1) \cdot g(NP_2)$
$c(S3) = c(NP_1) \cup c(NP_2)$

(A7) $g(C) = g(DNB)$
$d(C) = d(DNB)$
$c(C) = c(DNB)$

(A8) $g(C_1) = \boxed{g(C_2) \xrightarrow{n(PP)} g(PP)}$
$d(C_1) = d(C_2)$
$c(C_1) = c(C_2) \cup c(PP)$

(A9) $g(DNB) = g(ND)$
$d(DNB) = d(ND)$
$c(DNB) = c(ND)$

(A10) $g(ND_1) = \boxed{g(AP) \xrightarrow{REF} g(ND_2)}$
$d(ND_1) = d(ND_2)$
$c(ND_1) = c(AP)$

(A11) $g(ND) = \boxed{lex(NB)}$
$d(ND) = \{lex(NB)\}$
$c(ND) = \phi$

* * is the concatenation of graphs. $G_1 \cdot G_2$ is a graph which contains all nodes and edges of G_1 and G_2 where nodes with the same value are identified.

(A12) g(AP_1) = [lex(*adv*) —AVAJ→ g(AP_2)]
c(AP_1) = {lex(*adv*)} ∪ c(AP_2)
(A13) g(AP) = [lex(*AB*)]
c(AP) = {lex(*AB*)}
(A14) g(NP) = g(NPO) • g(SS)
c(NP) = c(NPO) ∪ c(SS)
(A15) g(NP) = g(NPO) • g(SS1)
c(NP) = c(NPO) ∪ c(SS1)
(A16) g(NPO) = g(NP1)
c(NPO) = c(NP1)
(A17) g(NPO) = [g(NP1) —n(PP)→ g(PP)]
c(NPO) = c(NP1) ∪ c(PP)
(A18) g(NP1) = g(N)
c(NP1) = c(N)
(A19) g(N_1) = [g(AP) —REF→ g(N_2)]
c(N_1) = c(AP) ∪ c(N_2)
(A20) g(N) = [lex(*NB*)]
c(N) = {lex(*NB*)}
(A21) g(N) = [lex(*NB*)]
c(N) = {lex(*NB*)}
(A22) g(PP) = g(NPO)
c(PP) = c(NPO)
n(PP) = lex(*prep*)
(A23) g(PREDD) = g(DAB)
d(PREDD) = d(DAB)
c(PREDD) = c(DAB)
(A24) g(PREDD) = [g(DAB) —n(PP)→ g(PP)]
d(PREDD) = d(DAB)
c(PREDD) = c(DAB) ∪ c(PP)
(A25) g(DAB) = g(AD)
d(DAB) = d(AD)
c(DAB) = c(AD)
(A26) g(AD_1) = [lex(*adv*)] —AVAJ→ g(AD_2)
d(AD_1) = d(AD_2)
c(AD_1) = {lex(*adv*)} ∪ c(AD_2)
(A27) g(AD) = [lex(*AB*)]
d(AD) = {lex(*AB*)}
c(AD) = ϕ
(A28) g(SS) = g(NP) —n(VP)→ g(VP)
c(SS) = c(NP) ∪ c(VP)
(A29) g(VP) = g(NP)
c(VP) = c(NP)
n(VP) = lex(*verb*)
(A30) g(SS1) = g(SS)
c(SS1) = c(SS)
(A31) g(SS1) = g(NP_1) • g(NP_2) • g(SS)
c(SS1) = c(NP_1) • c(NP_2) • c(SS)

Sample sentence recognition and network transduction

Definition. A base of the topology of a topological space X is any set B of open subsets of X such that every open subset of X is the union of sets belonging to B -

The reduction and transduction into the network of the first part of this sentence is as follows:

(A11) g = [base]
d = {base}
c = ϕ
(A9) see (A11)
(A7) see (A11)
(A20) g = [topology]
d = ϕ
c = {topology}
$(A18)_1$ see (A20)
$(A16)_1$ see (A20)
$(A22)_1$ g see (A20)
c see (A20)
n = OF
$(A8)_1$ g = [[base] —OF→ [topology]]
d see (A7)
c = {topology}
(A13) g = [topological]
c = {topological}
(A21) g = [space]
c = {space}
(A19) g = [[topological] —REF→ [space]]
c = {topological, space}
$(A18)_2$ see (A19)
$(A16)_2$ see (A19)
$(A22)_2$ g see (A19)
c see (A19)
n = OF
$(A8)_2$ g = [[[base] —OF→ [topology]] ↓OF [[topological] —REF→ [space]]]
d = {base}
c = {topology, topological, space}

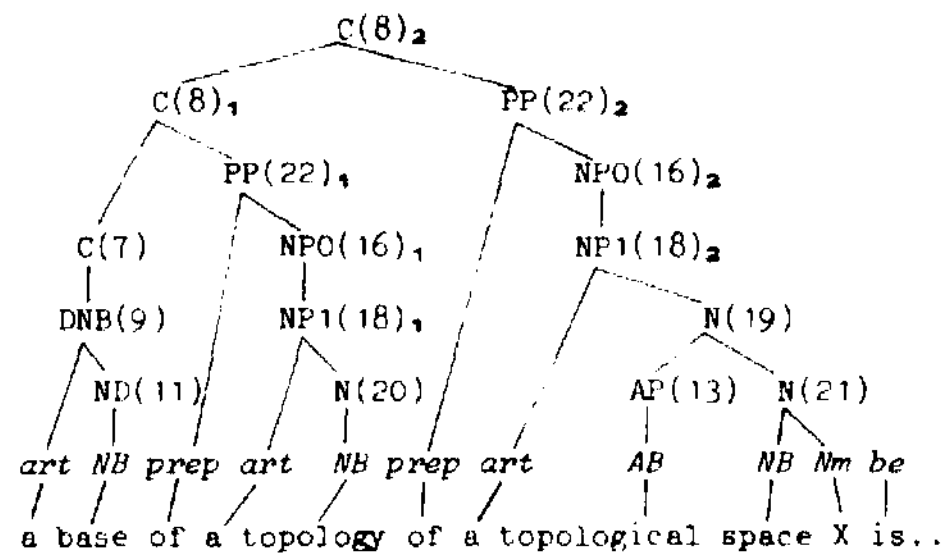


Fig. 3. The numbers given in parentheses with the nonterminal symbols are the production rule numbers. Subscripts are used for references.

The computer program that has been written for the construction of a semantic network from the definition of "general topology" acts on the basis of a dictionary of trivial terms which - at the beginning of a run - only contains articles, prepositions, pronouns and verbs. There are only about seven verbs in this mathematical texts. These can be treated as terms not bearing conceptual information because they generally do not act as concepts. (Exception: to converge. But for this verb there is an adjectival equivalent: convergent.) In performing the network construction, the program recognizer, ("learns") the terms occurring in the text ar. well as their grammatical category using these "learned facts" to analyse the further definitions. The grammar rules do not act as parameters of the program but are inserted as procedures. This makes the program more efficient but less indifferent to changes. The algorithm partly proceeds like a formal language recognition algorithm, i.e. it attempts to match sentence parts with right hand sides of the grammar rules. But the sentence structure is determined only in so far as it is needed by the semantic rules.

Open questions

1. Embeddings of more than one prepositional clause. Production rule (17) allows to embed one prepositional supplement for a noun phrase. By rule (22) and again (17) we can embed recursively another prepositional clause. Clauses like "an interior point of a subset A of X ..." are correctly analysed by these rules. But consider the clause "a mapping f of a topological space X into a set X' ...". The analysis of this clause by the rules described above is not adequate because the two NPO "a topological space X" and "a set X' " are both embedded parallelly into the NPO "a mapping f" . To find the correct analysis in each case, we need the information that "mapping" generally *requires* two prepositional supplements whereas "point" and "subset" require only one.

2. If we embed a relative clause into a noun phrase containing a prepositional supplement we can not - on the basis of syntax rules - correctly determine to which noun the relative pronoun refers. In the clause*an interior point of the set which contains ..." the reference can be determined correctly using the semantic information that only sets but not points can contain anything. But in the clause "a set of the filter which contains ..." the correct coordination is not possible because a filter is a set too. With the exeption *of* failures like those described above the program correctly transduced all definitions of the compendium.

Further development

Actually we are developping an algorithm to work on definitions out of the area of computer science. To perform this we will need more conceptual categories and relations that is to say we will need verbs as concepts and conceptual relations between verbs and nouns. This program is intended to work with rules as parameters. So it can be used for more than one grammar later. We intend to implement it with the help of a compiler compiler.

References

Noam Chomsky. Aspects of the theory of syntax- (1965)

[2] Donald =. Knuth. Semantics of Context Free Languages. Math. Systems Theory, 2 (1968)

Terence W. Pratt. Pair Grammars, Graph Languages and String-to-Graph Translations. J. of Comp. and System Sciences 5, (1971)

[4] M. Ross Quillian. The Teachable Language Comprehender. Comm. ACM 12, 8 (1969)

[5] *Frank de Renter. Transformational Grammars for* Languages and Compilers. Techn. Rep. No. 50, Univ. of Newcastle upon Tyne (1973)

[6] Camilla Schwind. Automatische Bestimmung von Begriffen und Beziehungen zwischen Begriffen in einem wissenschaftlichen Text. Diplomarbeit. Techn. Univ. Miinchen (1971)

[7] Camilla Schwind. Automatische Erzeugung von Stichphrasen auf der Basis eines Beziehungssystems mathematischer Begriffe. Techn. Univ. Miinchen. Bericht Nr. 7224 (1972)

[8] R. Simmons and J. Slocum. Generating English Discourse from Semantic Networks. Comm. ACM 15, 10 (197?)