*Information Processing & Management*, Vol. 12, pp. 147-153. Pergamon Press 1976. Printed in Great Britain

# AUTOMATIC, SEMANTICS-BASED INDEXING OF NATURAL LANGUAGE TEXTS FOR INFORMATION RETRIEVAL SYSTEMS†

STEPHAN BRAUN and CAMILLA SCHWIND
Institut für Informatik, Technische Universität München, Germany

(*Received* 5 *December* 1975)

**Abstract**—The fundamental idea of the work reported here is to extract index phrases from texts with the help of a single word concept dictionary and a thesaurus containing relations among concepts. The work is based on the fact, that, within every phrase, the single words the phrase is composed of are related in a certain well defined manner, the type of relations holding between concepts depending only on the concepts themselves. Therefore relations can be stored in a semantic network. The algorithm described extracts single word concepts from texts and combines them to phrases using the semantic relations between these concepts, which are stored in the network. The results obtained show that phrase extraction from texts by this semantic method is possible and offers many advantages over other (purely syntactic or statistic) methods concerning preciseness and completeness of the meaning representation of the text. But the results show, too, that some syntactic and morphologic "filtering" should be included for effectivity reasons.

## MOTIVATION

An information retrieval system should have an indexing device which automatically extracts concepts from the documents to be stored, in order to represent the content of the documents for later retrieval. Many different indexing methods have been used: semiautomatic methods[1], frequency criteria[2], syntax[3] and semantically oriented[4] methods, and a combination of frequency and syntax oriented criteria[5]. The desirability of using phrases as index terms rather than single words was already recognized in the research effort[6].

There are two main advantages of semantically oriented methods over other approaches: First, the index vector obtained will be more precise, because it is based on indexing by inclusion (i.e. what is meant by a good or useful index word is defined by a word list, and index terms are selected by help of this list only from the words of the text). Therefore, we will not be embarrassed by insignificant terms occurring in the text. This could happen when a dictionary is given containing all those words which are not good index terms. The selection method then consists in the exclusion of these "bad" index terms and the designation of the remaining terms as "good" index terms.

Secondly, the index vector obtained will be more complete. We will be able by the semantic method to find phrases which are intended by the text, but can in no way be extracted by simple matching (inclusive or exclusive) nor by more sophisticated syntax oriented methods.

We will show later that we cannot get along with semantic information alone, but that we must use some syntactic information to avoid errors and to resolve ambiguities. Syntax can be thought of as a filter for the phrases found by the semantic method.

## INTRODUCTION

As described in detail in[7] it is possible to generate all phrases (i.e. sequences of terms), which represent concepts within the scope of a certain scientific theory, by combining the "single word" concepts by means of a hierarchical semantic network. In[7] a phrase turned out to be represented by a hierarchical graph, it was generated by a hierarchical graph language. An example of such a rule is: If we know an adjective $A$ to be a possible reference term for a noun $N$, then $AN$ is a noun phrase. So we have: *continuous* refers to *mapping*, therefore *continuous mapping* is a noun phrase. In the work reported here we have applied these results to find, using semantic information only, all phrases occurring in a text. In a sentence we find the terms relevant to its meaning with the help of the dictionary appropriate to the network. From this subset of the set of all terms we determine the set of all phrases appropriate to the sentence: i.e. all phrases which are mentioned or intended in the sentence. This determination is done by applying rules which in turn are computed by inserting the single word concepts of the text into

†This work was supported by the SFB 49 of the Deutsche Forschungsgemeinschaft.

rule patterns. A rule pattern is a linear representation of a hierarchical graph generation rule. When applying the rules to the terms of a sentence we must attentively avoid using a term more frequently within a phrase than it occurs within the sentence. Therefore, the application of the rules is modified by a control language (see [8]). The algorithm proceeds as follows: From the single-word terms occurring in the sentence a set of "start phrases" is formed by combining two (or three) terms to make a phrase if one of the appropriate relations holds between them. This last information is gained from the semantic network. A start phrase is an expression with a form like $X$ REF $Y$ (or $X$ PREP $Y$ etc. see later). It is so designated, because these expressions act exactly like start symbols in a formal language. By inserting the start phrases into rule patterns we obtain a set of production rules. By controlled application of the production rules to the start phrases, we obtain the set of phrases appropriate to the sentence.

*Examples.* From the sentence *A poset is a set in which a binary relation* $x \leq y$ *is defined, which satisfies for all* $x$, $y$, $z$ *the following conditions...* we obtain the following start phrases: *binary relation, relation on set, relation on poset*; from these we obtain by combination the noun phrases *binary relation on set* and *binary relation on poset.* It would be very expensive to find syntactically the phrases *relation on set* and *relation on poset*, while semantically we obtain *relation on set* directly from the network. By profitably using the generic relation between broader and narrower terms, which holds between *set* and *poset*, we obtain *relation on poset.*

From the sentence *In chains, the notions of minimal and least (maximal and greatest) element of a subset are effectively equivalent* we get, among others, the phrases *minimal element of chain* and *element of minimal chain.* The second phrase does not occur within the sentence but is found from the network relation *minimal* REF *chain.* It sometimes happens that we get phrases which describe concepts not contained in the sentence. But in all such cases it has been possible to exclude these "false" phrases by neighbourhood criteria. So in the preceding example *minimal* occurs before *element* and between them there are only other adjectives and a conjunction, while *chain* precedes *minimal* and is separated from this adjective by words other than *is* or *is called.*

To include such criteria means using syntactic information also, and this runs counter the idea of generating phrases from semantic information only. But one of the objects of the research performed here has been to see, as well as to show, how far we can go using only semantic information and "how much" syntactic information must indispensably be used. We have seen that the "amount" of syntactic information to be included is very small and does not exceed criteria such as "Term $A$ is directly preceded by term $B$, whereas $C$ is not. So we coordinate $B$ to $A$ and not to $C$ when $B$ belongs semantically to $A$ as well as to $C$". Moreover, we have been able to find, on the basis of the semantic network, phrases occurring in the text which can never be formed on the basis of syntactic information.

## HIERARCHICAL SEMANTIC NETWORKS AND PHRASE GENERATION

First, we will briefly explain of what relations the hierarchical semantic network is composed and by what types of rules phrases are generated.

Let $N$ denote the set of all nouns, $A$ the set of all adjectives and $Av$ the set of the adverbs.

$\mathrm{SUP}_0 \subset (N \times N) \cup (A \times A)$ relates to a term its directly superordinated term; e.g. *injection* $\mathrm{SUP}_0$ *mapping.* The transitive closure of $\mathrm{SUP}_0$ is denoted by SUP.

$\mathrm{REF} \subset A \times N$ gives for an adjective all nouns which can be modified by this adjective; e.g. *continuous* REF *mapping.*

$\mathrm{PREP} \subset (A \cup N) \times N$ denotes the prepositional relation, which holds between a term (adjective or noun) and each of its possible prepositional supplements; e.g. *element* PREP *set* or *continuous* PREP *set.* There are several types of prepositional relations according to the different prepositions. We will handle the notation PREP like a variable whose values are names of prepositional relations. So PREP could be OF or IN etc.

$\mathrm{PRPP} \subset (N \cup A \times N \times N$ denotes the three-place prepositional relation; e.g. $(mapping, set, set) \in \mathrm{PRPP}$, $(isomorphic, set, set) \in \mathrm{PRPP}$.

$\mathrm{ANT} \in (A \times A) \cup (N \times N)$ denotes the antonymy relation which holds between pairs of terms such as *greater* and *less.*

$\mathrm{SYN} \subset (N \times N) \cup (A \times A)$ denotes the synonymy relation; e.g. *function* SYN *mapping.*

$\mathrm{AVAJ} \subset A \times A$ gives for an adverb all adjectives which are possibly modified by this adverb; e.g. *partially* AVAJ *ordered.*

We note only briefly that within the network we can restrict the domain of the relations SUP, REF, PREP, SYN, ANT and PRPP by taking into account the following inter-relations between the relations.

I1 SUP is an ordering relation. So it suffices to give the directly superordinated term in the network.

I2 REF is right compatible with SUP according to the restriction SHA. If A REF N and M $SUP_0$ N and not SHA($A, M$) then $A$ REF $M$ holds, where $SHA(x, y) = x \in DEF(y) \vee DEF(y) \cap [ANT(x) \cup SYN(x)] \neq \emptyset$, here DEF($y$) denotes the set of terms to which references exist from $y$ and ANT($x$) (resp. SYN($x$)) denotes the antonymy (resp. synonymy) class of $x$. The use of SHA suppresses relationships like *isomorph* REF *antiisomorphism* or *injective* REF *bijection.*

I3 PREP and PRPP are both right compatible with SUP. So we can state: If $A$ PREP $B$ (resp. $A$ PRPP $B$ PRPP $C$) and $N$ $SUP_0$ $B$ (resp. $N$ $SUP_0$ $B$ or $M$ $SUP_0$ $C$) then $A$ PREP $N$ (resp. $A$ PRPP $N$ PRPP $M$) holds.

I4 SYN is an equivalence relation.

I5 ANT is a symmetric relation.

I6 REF is left total, i.e. any adjective must refer to some substantive, because an adjective without appropriate substantives would have no contents.

It is obvious that within a phrase some of these relations hold between the single words of which the phrase is composed. In the phrase *a continuous mapping from a topological space to an ordered set* we find the relation REF to hold between the pairs (*continuous, mapping*), (*topological, space*) and (*ordered, set*), and the triplet (*mapping, space, set*) to belong to PRPP. Using these facts we can state the following phrase generation rules.†

(1) If an adjective $A$ and a noun $N$ are related by REF, then the sequence $A$ REF $N$ is a noun phrase,

(2) If an adverb $A$ is adverbial adjunct to an adjective $B$, then the sequence $A$ AVAJ $B$ is an adjectival phrase.

(3) If a term $B$ stands in a PREP-relation to a noun $N$, then the sequence $B$ PREP $N$ is a phrase.

(4) If PRPP holds between three nouns $A$, $B$ and $C$ then the sequence $A$ PRPP $B$ PRPP $C$ is a noun phrase.

From the elementary phrases generated so far we can form the "larger" phrases by inserting phrases into phrases in the following way:

(5) If $A$ REF $N$ is a noun phrase and $N$ occurs within a phrase $P$, then $N$ can be replaced by $A$ REF $N$. This rule can only be applied if SHA($X, Y$) or SHA($Y, X$) does not hold for any $B$ occurring within $P$ as adjectival modifier of $N$ and $X \in SUP(A)$, $Y \in SUP(B)$. This restriction avoids redundancies like *bijective injective mapping* and inconsistencies like *isomorphic antiisomorphic mapping.*

(6) If $A$ is an adjective and $A$ PREP $N$ (or $B$ AVAJ $A$ resp.) is a phrase, then we can replace each occurrence of $A$ within a phrase by $A$ PREP $N$ (or $B$ AVAJ $A$ resp.).

(7) If a noun $M$ occurs within a phrase "... $N$ PREP $M$..." or "... $N$ PRPP $M$..." and $M$ PREP $K$ (or $M$ PRPP $K$ PRPP $L$ resp.) is an elementary phrase, then we can replace $M$ by $M$ PREP $K$ (or by $M$ PRPP $K$ PRPP $L$ resp.). With these rules we can generate all possible correct phrases from single terms. We have used them to generate phrases from the single word concepts found by a dictionary match with a text.

## RULE PATTERNS AND RULES

Let $B$ be the set of all terms, $S$ be a sentence, $M_S \subset B$ the set of all index terms occuring in $S$. Then $A\chi_S = \{(X\,R\,Y) | X, Y \in M_S,\ (X, Y) \in \text{Rel},\ \text{Rel} \in \{\text{REF, PREP, AVAJ}\}\} \cup \{(X \text{ PRPP } Y \text{ PRPP } Z) | X, Y, Z \in M_S,\ (X, Y, Z) \in \text{PRPP}\}$ is the set of all start phrases over $M_S$. $A\chi_S$ corresponds to the set of elementary phrases over $M_S$. A rule pattern $Q$ is an expression of the form $X \to P/\alpha\_\beta$ where $P = X_1\,R\,X_2$ or $P = X_1$ PRPP $X_2$ PRPP $X_3$, with $X, X_1, X_2, X_3$ acting as variables and $\alpha, \beta \in RM \cup \{PRPP\}$, $R \in RM$, $RM = \{R \in F, PR \in P, AVAJ\}$. The variable $X$ occurring on the left hand side of $Q$ is supposed to occur on the right hand side, too, i.e. $X = X_1$ or $X = X_2$ (or $X = X_3$), $\alpha$ and $\beta$ are the left and the right context, resp. The set $R_{Q,S}$ of the rules

†For a more formal development in terms of hierarchical graphs see [7].

belonging to $Q$ and a sentence $S$ is obtained from $Q$ by replacing the variables of $Q$ by terms of $M_S$ in such a way that $P$ is an element of $A\chi_S$. $R_{Q,S} = \{X \to P/\alpha\_\beta | X \in M_S, P \in A\chi_S\}$. These are context sensitive phrase structure rules. The set $R$ of the following rule patterns, that have been used, will act exactly like the phrase generation rules informally described above:

$R1$ $X \to Y$ REF $X$ if not ($Z = Y$ or $Z$ ANT $Y$ or $Z$ SYN $Y$ or $Z$ SUP $Y$ or $Y$ SUP $Z$) for all $Z$ such that $Z$ REF $x\_$ and the string $x$ not containing PREP, AVAJ nor PRPP.†

$R2$ $Y \to Y$ PREP $Z$/PRPP_

$R3$ $Y \to Y$ PRPP $Z1$ PRPP $Z2$/PRPP_

$R4$ $Y \to Y$ PREP $Z$/PREP_

$R5$ $Y \to Y$ PRPP $Z1$ PRPP $Z2$/PREP_

$R6$ $X \to Y$ AVAJ $X$/_ REF

$R7$ $X \to X$ PREP $Y$/_ REF

Rule pattern $R1$ corresponds to rule (5) above, $R2$, $R3$, $R4$ and $R5$ correspond to (7), and $R6$ and $R7$ correspond to (6).

The set of rules $R_S$ belonging to a sentence $S$ then turns out to be $\{R_{X,S} | X \in R\}$.

## CONTROLLED GENERATION OF PHRASES

The set of phrases which can be formed from terms of $M_S$ is obtained by applying the grammar rules $R_S$ successively to each of the start phrases, paying attention to use a term within the phrases at most as frequently as it occurs in $S$. Consequently every rule can be applied at most as frequently as the terms of its right hand side occur in $M_S$. Thus, the application of the rules is limited. We can describe these limitations with the help of a control language.

Definition (see [8]): Let $G = (A_N, A_T, \Pi, \gamma)$ be a grammar, $A_N$ the set of nonterminal symbols, $A_T$ the set of the terminals, $\Pi$ the set of production rules and $\gamma \subset A_N$ the set of the start symbols. Let $F$ be a set of labels for production rules and $\kappa \subseteq \Pi \times F$ a labeling of $\Pi$. We will write $\kappa p$ for $f \in F$ if $(p, f) \in \kappa$. Let $x \in L(G)$ be a word generated by $G$ and $\alpha = X \to x_1 \to \underset{p_1}{\phantom{x}} \underset{p_2 \cdots p_n}{\phantom{x}} \to x_n (x_n = x)$ be a derivation of $x$ from $X \in \gamma$ in $G$, $p_i$ being the appropriate generation rules from $\Pi$, i.e. $x_i$ is directly derived from $x_{i-1}$ by $p_i \in \Pi$. Then the sequence $\kappa p_1 \kappa p_2 \ldots \kappa p_n \in F^*$ is called the control word for $x$ and $\alpha$ and is noted $\mathrm{Kon}(x, \alpha)$. Let $C \subset F^*$ be a set of words over $F$. Then $L_C(G) = \{x | x \in L(G)$ and there is a derivation $\alpha$ of $x$ and $\mathrm{Kon}(x, \alpha) \in C\}$ is called the language generated by $G$ controlled by $C$, $C$ being a control language for $G$.

For our purpose, we will need the following control language: Let $S$ be a text unit, $F$ a finite set of labels where $|F| \geq |R_S|$. Then we associate labels to production rules in such a way that two production rules, which have on their right hand side an adjective or an adverb in common, obtain the same label: $\kappa \subset R_S \times F$ so that

$$\kappa p = \kappa q \Leftrightarrow p = A \to C \text{ BEZ } D/\alpha\_\beta, q = B \to C \text{ BEZ } E/\gamma\_\delta \text{ where BEZ} \in \{\text{REF, AVAJ}\}.$$

If $x$ is a word over an alphabet we denote by $|X|_x$ the number of occurrences of $X$ in $x$. According to our remarks we define a control language $C$ to be $\{x | x \in F^*; x = f_1 \ldots f_n$ and $f_i \neq f_j$ for $i \neq j\}$. Then $L_C(G)$ is the set of all phrases we can generate in $S$.

As can easily be seen, the following theorem holds: $x \in L_C(G) \Leftrightarrow \forall X \in M_S |X|_x \leq |X|_S$ and $x \in L(G)$.

## IMPLEMENTATION AND RESULTS

The presupposition for the execution of the semantic analysis of a text, as described above, is the existence and availability of a semantic network appropriate to the conceptional domain the text comes from. We have tested the phrase extraction algorithm with texts dealing with lattice theory. We had at our disposition a thesaurus of about 600 terms. The relations are realized within the thesaurus in the following way:

$\mathrm{SUP}_0$ is specified in one direction only. Each term possessing superordinated ones has thesaurus entries to all its directly superordinated terms.

†The formation of such contextual conditions naturally exceeds the definition of context sensitive production rules. $R1$ can easily be reformulated so that it is context sensitive, but we then would get a very great number of rules (corresponding to the number of elements of $M_S$) and our grammar scheme would become very difficult to survey. For the treatment of negations within the context of grammar rules see [9].

REF, too, is always specified in one direction, namely for the adjectives. Each adjective has thesaurus entries to the substantives it modifies. If the SUP relation holds between some of these the thesaurus entry is always given only for the most general term.

PREP is always only specified in one direction. The prepositional supplements are given for the terms as thesaurus entries. As for REF, only the most general terms are specified in this sense.

PRPP is handled like PREP, with the exception that a pair of possible prepositional supplements is always given together.

ANT being symmetric is given in the two directions; i.e. if $X$ ANT $Y$, $X$ has a thesaurus entry ANT $Y$, and $Y$ has a thesaurus entry ANT $X$.

SYN is handled like ANT.

AVAJ is only specified for the adverb, each adverb has thesaurus entries to all adjectives it modifies.

Each term of our dictionary has no more than 9 references to other terms, their average number being 3. This thesaurus has been constructed by hand.

Text processing then proceeds as follows: A text unit called $S$ (generally a sentence) is read and each of its words is searched in the dictionary. The dictionary match routine comprises a very simple morphological analysis identifying inflected forms of words, the latter not being stored separately, with the exception of the plural forms of greek or latin words like criterion–criteria or datum–data. In this morphological analysis a word occurring within the dictionary is identified with a text word if the latter contains the former left-justified and has more than four letters in common with it. This morphological analysis has turned out to be sufficient for our purposes in the sense, that all inflected words occurring in our text have been correctly identified with their appropriate basic forms and no incorrect coordinations have been established. After that, the set of start phrases is calculated by simple thesaurus match as well as by exploiting the interrelations between the relations. The thesaurus match consists in looking for each pair (triple) $X$, $Y(, Z)$ of terms of $M_S$ where $X$ has a thesaurus entry $Y$ (resp. $Y$, $Z$) named REF, AVAJ, PREP, (resp. PRPP). The interrelations I1 to I6 between the relations are utilized by special routines which run along the SUP-chains within the thesaurus.

*Example.* If the algorithm is looking for *finite* REF *lattice*, it does not find *lattice* as a thesaurus entry for *finite*. But it will find *finite* REF *set*. Then it inspects the thesaurus entries of *lattice* to find one of the form $\text{SUP}_0$ *set*. It doesn't, but it finds instead *lattice* $\text{SUP}_0$ *semilattice*, this result giving rise to an inspection of the thesaurus entries of *semilattice* for $\text{SUP}_0$ *set*. In fact, it finds now directly *semilattice* $\text{SUP}_0$ *set*. At this point we have achieved our goal and we now are able to state that *finite* REF *lattice* holds. Consequently, we can establish the start phrase *finite* REF *lattice*. When we look for thesaurus entries we use word category information, i.e. we would, for example, not look for a thesaurus entry $B$ REF $A$ for a noun $B$ and a noun $A$. A word may be in order about the search along the SUP-chains. It could happen that, in case an adjective has more than one reference term, only one leads to the noun looked for. If we are so unlucky as not to hit immediately upon the noun we need, we must execute one or more SUP-chain inspections in vain until finding the correct one. By these thesaurus inspections and look-ups we form the set of all the start phrases $A_{\chi S}$ of $S$ according to the definitions given above.

Then these start phrases are inserted into the rule schemata $R1$ to $R7$. This insertion presupposes a phrase vs rule match to verify the correct binding according to the relation names and to identic variables on the left- and the right-hand sides of a rule scheme. If we have a rule scheme $X \to Y\,R\,X/\alpha\_\beta$ we first bind $X$ to a term $A \in M_S$ and then look for all start phrases of the form $B\,R\,A$ to establish the rule $A \to B\,R\,A/\alpha\_\beta$ ($\alpha$ and $\beta$ do not contain variables). This process is executed for all terms $A \in M_S$ successively.

Having calculated our rules we can now go on generating the phrases. The phrase generation procedure applies the generation rules to a start phrase until no more generation rules can be applied. The procedure is working on a stack in order to include all branching. Generation proceeds as follows: The leftmost symbol of the start phrase is searched in the rule set in order to be replaced by the appropriate phrase. If it is found the context conditions are examined, where rules belonging to $R1$ require a special treatment. In this case we must go to the left in the phrase until we find a relation name other than REF and verify that none of the relations SYN, ANT, SUP or identity holds among the new adjective to be inserted (in $R1$: $Y$) and one of the terms between the REF-relation names. Before the application of a rule the algorithm looks for all other possibilities of processing the phrase at this stage, i.e. for all other rules that can be applied at this moment. For this rule examination those rules which have the same left hand side are stored together and we need only check if we are on the last alternative of the right hand side or if there is another one to be inserted. If there is another possibility the phrase at this stage and the rule alternative is stored on the stack for further processing. The phrase is stacked, too, if we do not yet process the rightmost element of it, because in this case, too, we can get further results if we do not process the leftmost possible element of the phrase, but one further to the right, instead. This is not the case for a general formal-language generation algorithm but is a consequence of the non-free application of the rules. It could be the case that of two rules only one can be applied, and only once, within the whole process, because the terms they contain occur only once within the sentence. In this case these two rules have the same label. If we then have a phrase two elements of which match the left hand side of these two rules the algorithm must take into account the two possibilities of further processing and if the element farther to the left is replaced, remember that there could be an element to the right of the one being processed, to be replaced, too, but only if one of the appropriate rules is still "free". For example, if we had obtained the terms *chain*, *maximal* and *element* from a sentence where something like... *all elements of maximal chains*... occurs, we obtain the start phrases *maximal* REF *element*, *maximal* REF *chain*, *element* PREP *chain* and the production rules *element* → *maximal* REF *element*, *chain* → *maximal* REF *chain*, *element* → *element* PREP *chain*/PRPP_, *element* → *element* PREP *chain*/PREP_. When we work on the start phrase *element* PREP *chain* (the other two not producing a phrase), we first replace *element* by *maximal* REF *element* obtaining the phrase *maximal* REF *element* PREP *chain*. If we do not stack *element* PREP *chain* to try later to replace *chain* first, we do not get the correct phrase *element* PREP *maximal* REF *chain*, the single occurrence of *maximal* being used up already by the application of the rule to the first symbol *element*. For this reason the phrase being processed is always stored on the stack if we are not processing its rightmost element. When we are at the end of a phrase and there is no further rule to be applied, we have generated a phrase appropriate to the text unit $S$. The algorithm then looks to the stack in order to see if there is another "branch" to be pursued and processes this phrase on the stack with the next alternative of the appropriate rule or, if there is no further alternative, with the next symbol (further to the right) of the phrase to be replaced. If the stack is empty the process is terminated and all phrases have been generated. The control mechanism described above is executed by a counter array containing the number of occurrences of each term of $S$. Each time a rule is applied the counters of its right-hand side terms, which are newly inserted by the rule, are decreased by 1. If a counter of a term is 0 no more rules containing it can be applied. Naturally, the counter information must be stored on the stack, too, if something is put on the stack. If the text unit contains homographs, the whole process beginning with the start phrase calculation must be executed with all possible combinations of word meanings.

This process probably looks complicated and complex. The complications arise, because always all possibilities must be gone through. If we use very simple syntactic criteria for the succession of terms in the text unit, we can in most cases resolve the ambiguities resulting from homographs, as well as from the possibility of coordinating a term to more than one other term, by some relation, i.e. we are not further obliged to stack the phrase when we are not processing its last element in each stage, because the coordination will be determined by sentence position criteria.

To conclude the description of the algorithm let us demonstrate the phrase generation algorithm for the example sentence $S$ = *Any finite chain has a least and greatest element.*† From this sentence we get the set of all index terms together with the word categories and the numbers of occurrences:

*finite adjective* 1, *chain substantive* 1, *least adjective* 1, *greatest adjective* 1, *element substantive* 1.

The appropriate thesaurus relations are: *finite* REF *set*, *least* REF *element*, *greatest* REF *element*, *element* PREP *set*. Furthermore, we have a SUP-chain from *chain* to *set* via *poset*: *chain* $\text{SUP}_0$ *poset* $\text{SUP}_0$ *set*. So we get the following set of axioms: {*finite* REF *chain*, *least* REF *element*, *greatest* REF *element*, *element* PREP *chain*}, the first and the last axiom being set up by finding *finite* REF *set* (resp. *element* PREP *set*), *chain* $\text{SUP}_0$ *poset* and *poset* $\text{SUP}_0$ *set* in the thesaurus. The rule calculating procedure then proceeds as follows: We begin by $R1$ and the first index term *finite* of the sentence, inserting *finite* for $X$. This gives no rule, since there is no axiom $A$ REF *finite*. So we try the second term *chain* which gives us the rule

$P1$: *chain* → *finite* REF *chain*.

The process is repeated for each rule scheme and each term and we obtain the following rules:

$P2$: *element* → *least* REF *element*/*greatest* REF *element* [from $R1$]
$P3$: *element* → *element* PREP *chain*/PRPP__ [from $R2$]
$P4$: *element* → *element* PREP *chain*/PREP__ [from $R4$].

Then the phrase generation algorithm begins to process the first start phrase, namely *finite* REF *chain*. No rule can be applied to *finite* nor to *chain*, $P1$ being refused by the special context examination finding *finite* REF before *chain*. The second start phrase *least* REF *element* gives no phrase, too, the insertion of *greatest* REF *element* by $P1$ being refused because of the antonymy of *greatest* and *least*. This is the same for the start phrase *greatest* REF *element*. Only the last start phrase *element* PREP *chain* gives rise to a generation: First *element* is replaced by *least* REF *element*, applying $P2$. *element* PREP *chain* is stacked together with $P2$ and a reference to the place of *least* in the counter array, this one containing now 0 instead of 1 for *least*. Then the phrase *least* REF *element* PREP *chain* is further processed, beginning with the first term *least*, which cannot be replaced, nor can the next. *chain* finally can be replaced according to $P1$ giving the phrase *least* REF *element* PREP *finite* REF *chain*. No further processing is possible so we refer to the stack and try to process *element* PREP *chain* by the second alternative of $P2$ from which the phrase *greatest* REF *element* PREP *chain* is obtained. *element* PREP *chain* is stacked as before, not because there is another alternative rule, but because we could have replaced *chain* by something else instead of *element*. *greatest* REF *element* PREP *chain* leads to, as before, *greatest* REF *element* PREP *finite* REF *chain*. Processing of the stacked phrase then leads to *element* PREP *finite* REF *chain*. It can already be seen in this simple sentence, that the semantically based phrase generation algorithm is able to find phrases which can in no way be found syntactically, namely the coordination *element* PREP *chain*.

*Some further examples.* From the sentence *A function* $\Theta: P \to Q$ *from a poset* $P$ *to a poset* $Q$ *is called order-preserving or isotone if it satisfies* $x \leq y$ *implies* $\Theta(x) \leq \Theta(y)$ we obtain the noun phrases *order-preserving* REF *function* PRPP *poset* PRPP *poset* and *isotone* REF *function* PRPP *poset* PRPP *poset*. The REF-relation found here that gave rise to the coordinations *order-preserving* REF *function*... and *isotone* REF *function*... could have been found syntactically, too, but only by a rather sophisticated analysis comprising the segmentation of the sentence into NP COP AP where NP is the whole first part up to *is*, COP the part *is called* and AP the rest. To find this phrase semantically did not pose a problem, *isotone* as well as *order-preserving* refering to *function* in the thesaurus.

From the text unit *Let* $L$ *be any complete lattice, and let* $S$ *be any subset of* $L$ *such that* (*i*) $I \in S$, *and* (*ii*) $T \subset S$ *implies* $\inf T \in S$. *Then* $S$ *is a complete lattice*, we get the phrase *subset* PREP *complete* REF *lattice*. The coordination *subset* PREP *lattice* could have been found syntactically only by, first, noting that $L$ is a name for *lattice*, and then identifying later $L$ with *lattice*, recognizing then *subset* PREP *lattice*.

We have tested the algorithm with approx. 100 sentences obtaining the following results:

(1) Always all phrases mentioned or intended have been obtained.

(2) In about 30 sentences "false" coordinations were established, leading to phrases not intended by the text. However, all these failures could be eliminated by neighbourhood criteria, i.e. by the syntax filter, or by looking for an occurrence of the word *not* within the sentence.

## SYNTACTIC FILTERING

The second "result" mentioned at the end of the last paragraph gave rise to establishing a syntax filter at two levels of the text processing algorithm, first when the start phrases are formed and secondly when rules are applied. Consider a very hard example: From the sentence *Any graded poset satisfies the following JORDAN–DEDEKIND CHAIN CONDITION: All maximal chains between two elements have the same finite length* we get the start phrases *graded* REF *poset*, *graded* REF *chain*, *maximal* REF *chain*, *maximal* REF *element*, *maximal* REF *length*, *finite* REF *poset*, *finite* REF *chain*, *finite* REF *length*, *element* PREP *poset*, *element* PREP *chain*, *length* PREP *chain*. The reader can imagine what types of incorrect phrases are formed from

†All our example sentences are extracted of [Garret–Birkhoff. Lattice Theory].

these start phrases, e.g. *maximal* REF *element* PREP *finite* REF *graded* REF *poset* etc. We will not list all foolish coordinations here. All of the incorrect coordinations are already produced when forming the start phrases. By using the neighbourhood criterion, "If one adjective occurring in the text unit can be coordinated to more than one noun of the text, because it refers to more than one noun by the thesaurus, and if the adjective immediately precedes a noun, then coordinate it to this noun only", we can suppress all incorrect relations between *graded, maximal, finite* and the substantives set up in the above sentence, and we now obtain the start phrases which remain, namely *graded* REF *poset, maximal* REF *chain, finite* REF *length, element* PREP *poset, element* PREP *chain, length* PREP *chain*. The phrases formed from these start phrases, *element* PREP *graded* REF *poset, element* PREP *maximal* REF *chain, finite* REF *length* PREP *chain*, are all correct, i.e. occur within the sentence. By this very simple syntactic criterion we have gained very much without loosing any of the advantages offered by the semantic approach which is still fully utilized.

## CONCLUSION

We have obtained the following results:

(1) A semantically oriented approach to the problem of extracting phrases from texts offers many advantages. In particular, phrases mentioned within a text can be extracted, which can never be found by more syntactically based methods.

(2) At the same time, we have seen that some information about the succession of the terms in a text must indispensably be included. Otherwise the amount of incorrect coordinations is too large.

To conclude, a little remark concerning the construction of the thesaurus may be in order. The thesaurus we worked with has been constructed by hand. Although this work must be done only once when our semantic oriented approach to the indexing problem is used, it will perhaps be very difficult to find always somebody to do this work for any scientific area where texts are to be indexed. It might be very hard to persuade some master of the terminology of, say, law to use his knowledge to construct a thesaurus. For these reasons the next step must be to construct a thesaurus automatically from definitions of a scientific domain. A first approach for the mathematical domain of *General Topology* has been performed with success. The methods used there are by now being applied to computer science definitions (see [10]).

## REFERENCES

[1] W. A. Gray and A. J. Harley, Computer assisted indexing. *Inform. Stor. Retr.* 1971, 7, 167–174.

[2] G. Salton, C. S. Yang and C. T. Yu, Contribution to the theory of indexing. *Preprints of the IRIP Congress* 74. North-Holland, Amsterdam (1974).

[3] Stephan Braun, Automatische Indexierung durch linguistische Syntaxanalyse. GI 3. Jahrestagung, 1973, Lecture in Computer Sciences. Springer-Verlag, Berlin.

[4] Claude Ainsley Mackenzie, Relational indexing as a German language indexing tool. *Inform. Stor. Retr.* 1973, **9**, 665–688.

[5] Lois L. Earl, Experiments in automatic extracting and indexing. *Inform. Stor. Retr.* 1970, **6**, 313–334.

[6] Harold Borko, Experiments in book indexing by computer. *Inform. Stor. Retr.* 1970, **6**, 5–16.

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

[8] Arto Salomaa, *Formal Languages*. Academic Press, New York and London (1973).

[9] Andries P. J. Van der Walt, Random context languages. *Preprints of the IFIP Congress* 71. North-Holland, Amsterdam (1971).

[10] Camilla Schwind, Generating hierarchical semantic networks from natural language text. *Advance papers of the* IVth *Int. Joint Conf. on Artificial Intelligence* (1975).