# DESIGN OF A KNOWLEDGE-BASED REPORT GENERATOR

*Karen Kukich*

University of Pittsburgh
Bell Telephone Laboratories
Murray Hill, NJ 07974

## ABSTRACT

Knowledge-Based Report Generation is a technique for automatically generating natural language reports from computer databases. It is so named because it applies knowledge-based expert systems software to the problem of text generation. The first application of the technique, a system for generating natural language stock reports from a daily stock quotes database, is partially implemented. Three fundamental principles of the technique are its use of domain-specific semantic and linguistic knowledge, its use of macro-level semantic and linguistic constructs (such as whole messages, a phrasal lexicon, and a sentence-combining grammar), and its production system approach to knowledge representation.

## I. WHAT IS KNOWLEDGE-BASED REPORT GENERATION

A knowledge-based report generator is a computer program whose function is to generate natural language summaries from computer databases. For example, knowledge-based report generators can be designed to generate daily stock market reports from a stock quotes database, daily weather reports from a meteorological database, weekly sales reports from corporate databases, or quarterly economic reports from U. S. Commerce Department databases, etc. A separate generator must be implemented for each domain of discourse because each knowledge-based report generator contains domain-specific knowledge which is used to infer interesting messages from the database and to express those messages in the sublanguage of the domain of discourse. The technique of knowledge-based report generation is generalizable across domains, however, and the actual text generation component of the report generator, which comprises roughly one-quarter of the code, is directly transportable and readily tailorable.

Knowledge-based report generation is a practical approach to text generation. It's three fundamental tenets are the following. First, it assumes that much domain-specific semantic, linguistic, and rhetoric knowledge is required in order for a computer to automatically produce intelligent and fluent text. Second, it assumes that production system languages, such as those used to build expert systems, are well-suited to the task of representing and integrating semantic, linguistic, and rhetoric knowledge. Finally, it holds that macro-level knowledge units, such as whole semantic messages, a phrasal lexicon, clausal grammatical categories, and a clause-combining grammar, provide an appropriate level of knowledge representation for generating that type of text which may be categorized as periodic summary reports. These three tenets guide the design and implementation of a knowledge-based report generation system.

## II. SAMPLE OUTPUT FROM A KNOWLEDGE-BASED REPORT GENERATOR

The first application of the technique of knowledge-based report generation is a partially implemented stock report generator called Ana. Data from a Dow Jones stock quotes database serves as input to the system, and the opening paragraphs of a stock market summary are produced as output. As more semantic and linguistic knowledge about the stock market is added to the system, it will be able to generate longer, more informative reports.

Figure 1 depicts a portion of the actual data submitted to Ana for January 12, 1983. A hand drawn graph of the same data is included. The following text samples are Ana's interpretation of the data on two different runs.

DOW JONES INDUSTRIALS AVERAGE -- 01/12/83

| | | | | |
|---|---|---|---|---|
| 01/12 | CLOSE | 30 | INDUS | 1083.61 |
| 01/12 | 330PM | 30 | INDUS | 1089.40 |
| 01/12 | 3PM | 30 | INDUS | 1093.44 |
| 01/12 | 230PM | 30 | INDUS | 1100.07 |
| 01/12 | 2PM | 30 | INDUS | 1095.38 |
| 01/12 | 130PM | 30 | INDUS | 1095.75 |
| 01/12 | 1PM | 30 | INDUS | 1095.84 |
| 01/12 | 1230PM | 30 | INDUS | 1095.75 |
| 01/12 | NOON | 30 | INDUS | 1092.35 |
| 01/12 | 1130AM | 30 | INDUS | 1089.40 |
| 01/12 | 11AM | 30 | INDUS | 1085.08 |
| 01/12 | 1030AM | 30 | INDUS | 1085.36 |
| 01/11 | CLOSE | 30 | INDUS | 1083.79 |

CLOSING AVERAGE 1083.61 DOWN 0.18

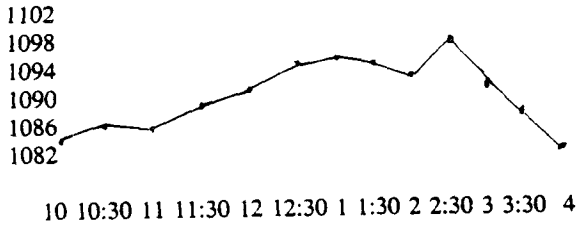


Figure 1

(1)

after climbing steadily through most of the morning , the stock market was pushed downhill late in the day . stock prices posted a small loss , with the indexes turning in a mixed showing yesterday in brisk trading .

the Dow Jones average of 30 industrials surrendered a 16.28 gain at 4pm and declined slightly , finishing the day at 1083.61 , off 0.18 points .

(2)

wall street's securities markets rose steadily through most of the morning , before sliding downhill late in the day . the stock market posted a small loss yesterday , with the indexes finishing with mixed results in active trading .

the Dow Jones average of 30 industrials surrendered a 16.28 gain at 4pm and declined slightly , to finish at 1083.61 , off 0.18 points .

## III. SYSTEM OVERVIEW

In order to generate accurate and fluent summaries, a knowledge-based report generator performs two main tasks: first, it infers semantic messages from the data in the database; second, it maps those messages into phrases in its phrasal lexicon, stitching them together according to the rules of its clause-combining grammar, and incorporating rhetoric constraints in the process. As the work of McKeown[1] and Mann and Moore[2] demonstrates, neither the problem of deciding what to say nor the problem of determining how to say it is trivial, and as Appelt[3] has pointed out, the distinction between them is not always clear.

### A. System Architecture

A knowledge-based report generator consists of the following four independent, sequential components: 1) a fact generator, 2) a message generator, 3) a discourse organizer, and 4) a text generator. Data from the database serves as input to the first module, which produces a stream of facts as output; facts serve as input to the second module, which produces a set of messages as output; messages form the input to the third module, which organizes them and produces a set of ordered messages as output; ordered messages form the input to the fourth module, which produces final text as output. The modules function independently and sequentially for the sake of computational manageability at the expense of psychological validity.

With the exception of the first module, which is a straightforward C program, the entire system is coded in the OPS5 production system language.[4] At the time that the sample output above was generated, module 2, the message generator, consisted of 120 production rules; module 3, the discourse organizer contained 16 production rules; and module 4, the text generator, included 109 production rules and a phrasal dictionary of 519 entries. Real time processing requirements for each module on a lightly loaded VAX 11/780 processor were the following: phase 1 - 16 seconds, phase 2 - 34 seconds, phase 3 - 24 seconds, phase 4 - 1 minute, 59 seconds.

### B. Knowledge Constructs

The fundamental knowledge constructs of the system are of two types: 1) static knowledge structures, or memory elements, which can be thought of as n-dimensional propositions, and 2) dynamic knowledge structures, or production rules, which perform pattern-recognition operations on n-dimensional propositions. Static knowledge structures come in five flavors: facts, messages, lexicon entries, medial text elements, and various control elements. Dynamic knowledge constructs occur in ten varieties: inference productions, ordering productions, discourse mechanics productions, phrase selection productions, syntax selection productions, anaphora selection productions, verb morphology productions, punctuation selection productions, writing productions, and various control productions.

### C. Functions

The function of the first module is to perform the arithmetic computation required to produce facts that contain the relevant information needed to infer interesting messages, and to write those facts in the OPS5 memory element format. For example, the fact that indicates the closing status of the Dow Jones Average of 30 Industrials for January 12, 1983 is:

(make fact ^fname CLSTAT ^iname DJI ^itype COMPOS ^date 01/12 ^hour CLOSE ^open-level 1084.25 ^high-level 1105.13 ^low-level 1075.88 ^close-level 1083.61 ^cumul-dir DN ^cumul-deg 0.18)

The function of the second module is to infer interesting messages from the facts using inferencing productions such as the following:

```
(p ____instan-mixedup
     (goal ^stat act ^op instanmixed)
     (fact ^fname CLSTAT ^iname DJI
          ^cumul-dir UP ^repdate <date>)
     (fact ^fname ADVDEC ^iname NYSE
          ^advances <x> ^declines {<y> > <x>})
   -->
     (make message ^top GENMKT ^subtop MIX
          ^mix mixed ^repdate <date>
          ^subjclass MKT ^tim close)
     (make goal ^stat pend ^op writemessage)
     (remove 1)
)
```

This production infers that if the closing status of the Dow had a direction of 'up', and yet the number of declines exceeded the number of advances for the day, then it can be said that the market was mixed. The message that is produced looks like this:

```
(make message ^repdate 01/12 ^top GENMKT
^subsubtop nil ^subtop MIX ^subjclass MKT
^dir nil ^deg nil ^vardeg | nil | ^varlev | nil | ^mix
mixed ^chg nil ^sco nil ^tim close ^vartim | nil |
^dur nil ^vol nil ^who nil )
```

The inferencing process in phase 2 is hierarchically controlled.

Module 3 performs the uncomplicated task of grouping messages into paragraphs, ordering messages within paragraphs, and assigning a priority number to each message. Priorities are assigned as a function of topic and subtopic. The system "knows" a default ordering sequence, and it "knows" some exception rules which assign higher priorities to messages of special significance, such as indicators hitting record highs. As in module 2, processing is hierarchically controlled. Eventually, modules 2 and 3 should be combined so that their knowledge could be shared.

The most complicated processing is performed by module 4. This processing is not hierarchically controlled, but instead more closely resembles control in an ATN. Module 4, the text generator, coordinates and executes the following activities: 1) selection of phrases from the phrasal lexicon that both capture the semantic meaning of the message and satisfy rhetorical constraints; 2) selection of appropriate syntactic forms for predicate phrases, such as sentence, participial clause, prepositional phrase, etc.; 3) selection of appropriate anaphora for subject phrases 4) morphological processing of verbs; 5) interjection of appropriate punctuation; and 6) control of discourse mechanics, such as inclusion of more than one clause per sentence and more than one sentence per paragraph.

The module 4 processor is able to coordinate and execute these activities because it incorporates and integrates the semantic, syntactic, and rhetoric knowledge it needs into its static and dynamic knowledge structures. For example, a phrasal lexicon entry that might match the "mixed market" message is the following:

```
(make phraselex ^top GENMKT ^subtop MIX
^mix mixed ^chg nil  ^tim close ^subjtype
NAME ^subjclass MKT ^predfs turned ^predfpl
turned ^predpart turning ^predinf |to turn|
^predrem |in a mixed showing| ^len 9  ^rand 5
^imp 11)
```

An example of a syntax selection production that would select the syntactic form subordinate-participial-clause as an appropriate form for a phrase (as in "after rising steadily through most of the morning") is the following:

```
(p ________5.selectsuborpartpre-selectsyntax
     (goal ^stat act ^op selectsyntax)          ; 1
     (sentreq ^sentstat nil)                    ; 2
     (message ^foc in ^top <t> ^tim <> nil
          ^subjclass <sc>)                      ; 3
     (message ^foc nil ^top <t> ^tim <> nil
          ^subjclass <sc>)                      ; 4
     (paramsynforms ^suborpartpre <set>)        ; 5
     (randnum ^randval < <set>)                 ; 6
     (lastsynform ^form << initsent prepp >> )  ; 7
     - (openingsynform ^form
          << suborsent suborpart >>)
     - (message ^foc in ^tim close)
   -->
     (remove 1)
     (make synform ^form suborpart )
     (modify 4 ^foc peek )
     (make goal ^stat act ^op selectsubor)
)
```

D. Context-Dependent Grammar

Syntax selection productions, such as the example above, comprise a context-dependent, right-branching, clause-combining grammar. Because of the attribute-value, pattern-recognition nature of these grammar rules and their use of the lexicon, they may be viewed as a high-level variant of a lexical functional grammar.[5] The efficacy of a low-level functional grammar for text generation has been demonstrated in McKeown's TEXT system.[6]

For each message, in sequence, the system first selects a predicate phrase that matches the semantic content of the message, and next selects a syntactic form, such as sentence or prepositional phrase, into which form the predicate phrase may be hammered. The system's default goal is to form complex sentences by combining a variable number of messages expressed in a variety of syntactic forms in each sentence. Every message may be expressed in the syntactic form of a simple sentence. But under certain grammatical and rhetorical conditions, which are specified in the syntax selection productions, and which sometimes include looking ahead at the next sequential message, the system opts for a different syntactic form.

The right-branching behavior of the system implies that at any point the system has the option to lay down a period and start a new sentence. It also implies that embedded subject-complement forms, such as relative

clauses modifying subjects, are trickier to implement (and have not been implemented as yet). That embedded subject complements pose special difficulties should not be considered discouraging. Developmental linguistics research reveals that "operations on sentence subjects, including subject complementation and relative clauses modifying subjects" are among the last to appear in the acquisition of complex sentences,[7] and a knowledge-based report generator incorporates the basic mechanism for eventually matching messages to nominalizations of predicate phrases to create subject complements, as well as the mechanism for embedding relative clauses.

## IV. THE DOMAIN-SPECIFIC KNOWLEDGE REQUIREMENT TENET

How does one determine what knowledge must incorporated into a knowledge-based report generator? Because the goal of a knowledge-based report generator is to produce reports that are indistinguishable from reports written by people for the same database, it is logical to turn to samples of naturally generated text from the specific domain of discourse in order to gain insights into the semantic, linguistic, and rhetoric knowledge requirements of the report generator.

Research in machine translation[8] and text understanding[9] has demonstrated that not only does naturally generated text disclose the lexicon and grammar of a sublanguage, but it also reveals the essential semantic classes and attributes of a domain of discourse, as well as the relations between those classes and attributes. Thus, samples of actual text may be used to build the phrasal dictionary for a report generator and to define the syntactic categories that a generator must have knowledge of. Similarly, the semantic classes, attributes and relations revealed in the text define the scope and variety of the semantic knowledge the system must incorporate in order to infer relevant and interesting messages from the database.

Ana's phrasal lexicon consists of subjects, such as "wall street's securities markets", and predicates, such as "were swept into a broad and steep decline", which are extracted from the text of naturally generated stock reports. The syntactic categories Ana knows about are the clausal level categories that are found in the same text, such as, sentence, coordinate-sentence, subordinate-sentence, subordinate-participial-clause, prepositional-phrase, and others.

Semantic analysis of a sample of natural text stock reports discloses that a hierarchy of approximately forty message classes accounts for nearly all of the semantic information contained in the "core market sentences" of stock reports. The term "core market sentences" was introduced by Kittredge to refer to those sentences which can be inferred from the data in the data base without reference to external events such as wars, strikes, and corporate or government policy making.[10] Thus, for example, Ana could say "Eastman Kodak advanced 2 3/4 to 85 3/4;" but it could not append "it announced development of the world's fastest color film for delivery in 1983.". Ana currently has knowledge of only six message classes. These include the closing market status message, the volume of trading message, and the mixed market message, the interesting market fluctuations message, the closing Dow status message, and the interesting Dow fluctuations message.

## V. THE PRODUCTION SYSTEM KNOWLEDGE REPRESENTATION TENET

The use of production systems for natural language processing was suggested as early as 1972 by Heidorn,[11] whose production language NLP is currently being used for syntactic processing research. A production system for language understanding has been implemented in OPS5 by Frederking.[12] Many benefits are derived from using a production system to represent the knowledge required for text generation. Two of the more important advantages are the ability to integrate semantic, syntactic, and rhetoric knowledge, and the ability to extend and tailor the system easily.

### A. Knowledge Integration

Knowledge integration is evident in the production rule displayed earlier for selecting the syntactic form of subordinate participial clause. In English, that production said:

IF
1) there is an active goal to select a syntactic form
2) the sentence requirement has not been satisfied
3) the message currently in focus has topic <t>, subject class <sc>, and some non-nil time
4) the next sequential message has the same topic, subject class, and some non-nil time
5) the subordinate-participial-clause parameter is set at value <set>
6) the current random number is less than <set>
7) the last syntactic form used was either a prepositional phrase or a sentence initializer
8) the opening syntactic form of the last sentence was not a subordinate sentence or a subordinate participial clause
9) the time attribute of the message in focus does not have value 'close'

THEN
1) remove the goal of selecting a syntactic form
2) make the current syntactic form a subordinate participial clause
3) modify the next sequential message to put it in peripheral focus
4) set a goal to select a subordinating conjunction.

It should be apparent from the explanation that the rule integrates semantic knowledge, such as message topic and time, syntactic knowledge, such as whether the sentence requirement has been satisfied, and rhetoric knowledge, such as the preference to avoid using subordinate clauses as the opening form of two consecutive sentences.

B. Knowledge Tailoring and Extending

Conditions number 5 and 6, the syntactic form parameter and the random number, are examples of control elements that are used for syntactic tailoring. A syntactic form parameter may be preset at any value between 1 and 11 by the system user. A value of 8, for example, would result in an 80 percent chance that the rule in which the parameter occurs would be satisfied if all its other conditions were satisfied. Consequently, on 20 percent of the occasions when the rule would have been otherwise satisfied, the syntactic form parameter would prevent the rule from firing, and the system would be forced to opt for a choice of some other syntactic form. Thus, if the user prefers reports that are low on subordinate participial clauses, the subordinate participial clause parameter might be set at 3 or lower.

The following production contains the bank of parameters as they were set to generate text sample (2) above:

```
(p _1.setparams
    (goal ^stat act ^op setparams)
  -->
    (remove 1)
    (make paramsyllables ^val 30)
    (make parammessages ^val 3)
    (make paramsynforms
        ^sentence 11
        ^coorsent 11
        ^suborsent 11
        ^prepphrase 11
        ^suborsentpre 5
        ^suborpartpre 8
        ^suborsentpost 8
        ^suborpartpost 11
        ^suborpartsentpost 11
    )
)
```

When sample text (1) was generated, all syntactic form parameters were set at 11. The first two parameters in the bank are rhetoric parameters. They control the maximum length of sentences in syllables (roughly) and in number of messages per sentence.

Not only does production system knowledge representation allow syntactic tailoring, but it also permits semantic tailoring. Ana could be tailored to focus on particular stocks or groups of stocks to meet the information needs of individual users. Furthermore, a production system is readily extensible. Currently, Ana has only a small amount of general knowledge about the stock market and is far from a stock market expert. But any knowledge that can be made explicit can be added to the system prolonged incremental growth in the knowledge of the system could someday result in a system that truly is a stock market expert.

## VI. THE MACRO-LEVEL KNOWLEDGE CONSTRUCTS TENET

The problem of dealing with the complexity of natural language is made much more tractable by working in macro-level knowledge constructs, such as semantic units consisting of whole messages, lexical items consisting of whole phrases, syntactic categories at the clause level, and a clause-combining grammar. Macro-level processing buys linguistic fluency at the cost of semantic and linguistic flexibility. However, the loss of flexibility appears to be not much greater than the constraints imposed by the grammar and semantics of the sublanguage of the domain of discourse. Furthermore, there may be more to the notion of macro-level semantic and linguistic processing than mere computational manageability.

The notion of a phrasal lexicon was suggested by Becker,[13] who proposed that people generate utterances "mostly by stitching together swatches of text that they have heard before. Wilensky and Arens have experimented with a phrasal lexicon in a language understanding system.[14] I believe that natural language behavior will eventually be understood in terms of a theory of stratified natural language processing in which macro-level knowledge constructs, such as those used in a knowledge-based report generator, occur at one of the higher cognitive strata.

A poor but useful analogy to mechanical gear-shifting while driving a car can be drawn. Just as driving in third gear makes most efficient use of an automobile's resources, so also does generating language in third gear make most efficient use of human information processing resources. That is, matching whole phrases and applying a clause-combining grammar is cognitively economical. But when only a near match for a message can be found in a speaker's phrasal dictionary, the speaker must downshift into second gear, and either perform some additional processing on the phrase to transform it into the desired form to match the message, or perform some processing on the message to transform it into one that matches the phrase. And if not even a near match for a message can be found, the speaker must downshift into first gear and either construct a phrase from elementary lexical items, including words, prefixes, and suffixes, or reconstruct the message.

As currently configured, a knowledge-based text generator operates only in third gear. Because the units of processing are linguistically mature whole phrases, the report generation system can produce fluent text without having the detailed knowledge-needed to construct mature phrases from their elementary components. But there is nothing except the time and insight of a system implementor to prevent this detailed knowledge from being added to the system. By experimenting with additional knowledge, a system could gradually be extended to shift into lower gears, to exhibit greater interaction between semantic and linguistic components, and to do more flexible, if not creative, generation of semantic

messages and linguistic phrases. A knowledge-based report generator may be viewed as a starting tool for modeling a stratiform theory of natural language processing.

## VII. CONCLUSION

Knowledge-based report generation is practical because it tackles a moderately ill-defined problem with an effective technique, namely, a macro-level, knowledge-based, production system technique. Stock market reports are typical instances of a whole class of summary-type periodic reports for which the scope and variety of semantic and linguistic complexity is great enough to negate a straightforward algorithmic solution, but constrained enough to allow a high-level cross-wise slice of the variety of knowledge to be effectively incorporated into a production system. Even so, it will be some time before the technique is cost effective. The time required to add knowledge to a system is greater than the time required to add productions to a traditional expert system. Most of the time is spent doing semantic analysis for the purpose of creating useful semantic classes and attributes, and identifying the relations between them. Coding itself goes quickly, but then the system must be tested and calibrated (if the guesses on the semantics were close) or redone entirely (if the guesses were not close). Still, the initial success of the technique suggests its value both as a basic research tool, for exploring increasingly more detailed semantic and linguistic processes, and as an applied research tool, for designing extensible and tailorable automatic report generators.

## ACKNOWLEDGEMENT

I wish to express my deep appreciation to Michael Lesk for his unfailing guidance and support in the development of this project.

## REFERENCES

1. Kathleen R. McKeown, "The TEXT System for Natural Language Generation: An Overview," *Proceedings of the Twentieth Annual Meeting of the Association for Computational Linguistics*, Toronto, Canada (1982).
2. James A. Moore and William C. Mann, "A Snapshot of KDS: A Knowledge Delivery System," in *Proceedings of the 17th Annual Meeting of the Association for Computational linguistics*, La Jolla, California (11-12 August 1979).
3. Douglas E. Appelt, "Problem Solving Applied to Language Generation," pp. 59-63 in *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, University of Pennsylvania, Philadelphia, PA (June 19-22,1980).
4. C. L. Forgy, "OPS-5 User's Manual," CMU-CS-81-135, Dept of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213 (July 1981).
5. Joan Bresnan and Ronald M. Kaplan, "Lexical-Functional Grammar: A Formal System for Grammatical Representation," Occasional Paper #13, MIT Center for Cognitive Science (1982).
6. Kathleen Rose McKeown, "Generating Natural Language Text in Response to Questions about Database Structure," Doctoral Dissertation, University of Pennsylvania Computer and Information Science Department (1982).
7. Melissa Bowerman, "The Acquisition of Complex Sentences," pp. 285-305 in *Language Acquisition*, ed. Michael Garman, Cambridge University Press, Cambridge (1979).
8. Richard Kittredge and John Lehrberger, *Sublanguages: Studies of Language in Restricted Semantic Domains*, Walter DeGruyter, New York (in press).
9. Naomi Sager, "Information Structures in Texts of a Sublanguage," in *The Information Community: Alliance for Progress - Proceedings of the 44th ASIS Annual Meeting, Volume 18*, Knowlton Industry Publications for the American Society for Information Science, White Plains, N.Y. (October 1981).
10. Richard I. Kittredge, "Semantic Processing of Texts in Restricted Sublanguages," *Computers and Mathematics with Applications* 8(0), Pergamon Press (1982).
11. George E. Heidorn, "Natural Language Inputs to a Simulation Programming System," NPS-55HD72101A, Naval Postgraduate School, Monterey, CA (October 1972).
12. Robert E. Frederking, *A Production System Approach to Language Understanding*, To appear (1983).
13. Joseph Becker, "The Phrasal Lexicon," pp. 70-73 in *Theoretical Issues in Natural Language Processing*, ed. B. I. Nash-Webber, Cambridge, Massachusetts (10-13 June 1975).
14. Robert Wilensky and Yigel Arens, "PHRAN -- A Knowledge-Based Natural Language Understander," pp. 117-121 in *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics*, University of Pennsylvania, Philadelphia, Pennsylvania (June 19-22, 1980).