TWO DISCOURSE GENERATORS

William C. Mann
USC Information Sciences Institute

WHAT IS DISCOURSE GENERATION?

The task of discourse generation is to produce multisentential text in natural language which (when heard or read) produces effects (informing, motivating, etc.) and impressions (conciseness, correctness, ease of reading, etc.) which are appropriate to a need or goal held by the creator of the text.

Because even little children can produce multisentential text, the task of discourse generation appears deceptively easy. It is actually extremely complex, in part because it usually involves many different kinds of knowledge. The skilled writer must know the subject matter, the beliefs of the reader and his own reasons for writing. He must also know the syntax, semantics, inferential patterns, text structures and words of the language. It would be complex enough if these were all independent bodies of knowledge, independently employed. Unfortunately, they are all interdependent in intricate ways. The use of each must be coordinated with all of the others.

For Artificial Intelligence, discourse generation is an unsolved problem. There have been only token efforts to date, and no one has addressed the whole problem. Still, those efforts reveal the nature of the task, what makes it difficult and how the complexities can be controlled.

In comparing two AI discourse generators here we can do no more than suggest opportunities and attractive options for future exploration. Hopefully we can convey the benefits of hindsight without too much detailed description of the individual systems. We describe them only in terms of a few of the techniques which they employ, partly because these techniques seem more valuable than the system designs in which they happen to have been used.

THE TWO SYSTEMS

The systems which we study here are PROTEUS, by Anthony Davey at Edinburgh [Davey 79], and KDS by Mann and Moore at ISI [Mann and Moore 80]. As we will see, each is severely limited and idiosyncratic in scope and technique. Comparison of their individual skills reveals some technical opportunities.

Why do we study these systems rather than others? Both of them represent recent developments, in Davey's case, recently published. Neither of them has the appearance of following a hand-drawn map or some other humanly-produced sequential presentation. Thus their performance represents capabilities of the programs more than capabilities of the programmer. Also, they are relatively unfamiliar to the AI audience. Perhaps most importantly, **they have written some of the best machine-produced discourse of the existing art.**

First we identify particular techniques in each system which contribute strongly to the quality of the resulting text. Then we compare the two systems discussing their common failings and the possibilities for creating a system having the best of both.

DAVEY'S PROTEUS

PROTEUS creates commentary on games of tic-tac-toe (noughts and crosses.) Despite the apparent simplicity of this task, the possibilities of producing text are rich and diverse. (See the example in Appendix .) The commentary is intended both to convey the game (except for insignificant variations of rotation and reflection), and also to convey the significance of each move, including showing errors and missed opportunities.

PROTEUS can be construed as consisting of three principal processors, as shown in Figure 1.

Move characterization employs a ranked set of move generators, each identified as defensive or offensive, and each identified further with a named tactic such as blocking, forking or completing a win. A move is characterized as being **a use of the tactic which is associated with the highest-ranked move generator which can generate that move in the present situation.** The purpose of move characterizaiton is to interpret the facts so that they become significant to the reader. (Implicitly, the system embodies a theory of the significance of facts.)

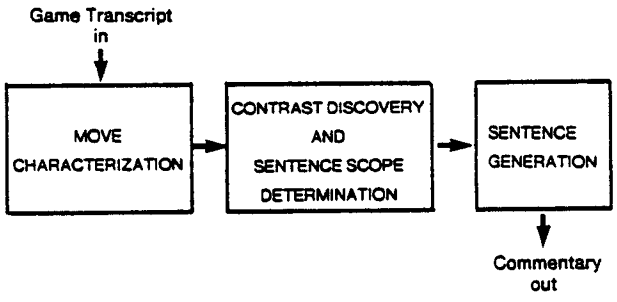


Figure 1: Principal Processors of PROTEUS

Contrast arises between certain time-adjacent moves and also between an actual move and alternative possibilities at the same point. For example:

- **Best move VS. Actual move:** The move generators are used to compute the "best" move, which is compared to the actual one. If the move generator for the best move has higher rank than any generator proposing the actual move, then the actual move is treated as a mistake, putting the best move and the actual move in contrast.

- **Threat VS. Block:** A threat contrasts with an immediately following block. This contrast is a fixed reflex of the system. It seems acceptable to mark any goal pursuit followed by blocking of the goal as contrastive.

Sentence scope is determined by several heuristic rules including

1. Express as many contrasts as possible explicitly. (This leads to immediate selection of words such as "but" and "however".)

2. Limit sentences to 3 clauses.

3. Put as many clauses in a sentence as possible.

4. Express only the worst of several mistakes.

The main clause structure is built before entering the grammar.

Both the move characterization process and the use of contrasts as the principal basis of sentence scope contribute a great deal to the quality of the resulting text. However, Davey's central concern was not with these two processes but with the third one, sentence generation. His system includes an elaborate Systemic Grammar, which he describes in detail in [Davey 79]. The grammar draws on work of Halliday [Halliday 76], Hudson [Hudson 71], Winograd [Winograd 72], Sinclair [Sinclair 72], Huddleston [Huddleston 71] and E. K. Brown, following Hudson most closely.[1]

Hudson's work offers a number of significant advantages to anyone considering implementing a discourse generation system.

1. Comprehensiveness- Its coverage of English is more extensive than comparable work.

2. Explicitness- the rules are spelled out in full in formal notation.

3. Unity- Since the grammar is defined in a single publication with a single authorship, the issues of compatibility of parts are minimized.

It is interesting that Davey does not employ the Systemic Grammar derivation rules at the highest level. Although the grammar is defined in terms of the generation of sentences, Davey enters it at the clause level with a sentence description which conforms to Systemic Grammar but was built by other means. A sentence at this level is composed principally of clauses, but **the surface conjunctions have already been chosen.**

Although Davey makes no claim, this may represent a general result about text generation systems. Above some level of abstraction in the text planning process, planning is not conditioned by the content of the grammar. The obvious place to expect planning to become independent of the grammar is at the sentence level. But in both PROTEUS and KDS, operations independent of the grammar extend down to the level of independent clauses within sentences. Top level conjunctions are not within such clauses, so they are determined by planning processes before the grammar is entered.

It would be extremely awkward to implement Davey's sentence scope heuristics in a systemic grammar. The formalism is not well suited for operations such as maximizing the total number of explicit contrastive elements. However, the problem is not just a problem with the formalism; grammars generally do not deal with this sort of operations, and so are poorly equipped to do so.

[1] The direction of derivation in Systemic Grammar is the generation direction, so that there is no need to "reverse" it. Generation is divided into two kinds of activity: choices, which come in sets of alternatives (called "systems," hence "Systemic"), and rule-applications. A system of choices (such as the choice between "demonstrative" and "possessive" among determiners which are "selective") is reached through other choices and so is conditional, but any choice, once reached, is unconstrained. Rule application is deterministic. Successive intervals of choice and rule-application lead to a sequence of feature-sets, each of type "Word," which enable lexical substitutions. There are no transformations.

Although the computer scientist who tries to learn from [Davey 79] will find that it presents difficulties, the underlying system is interesting enough to be worth the trouble. Davey's implementation generally attempts to be orthodox, conforming to [Hudson 71]. Davey regularizes some of the rules toward type uniformity, and thus reduces the apparent correspondence to Hudson's formulations. However, the linguistic base does not appear to have been compromised by the implementation.

One of the major strengths of the work is that it takes advantage of a comprehensive, explicit and linguistically justified grammar.

Text quality is also enhanced by some simple filtering (of what will be expressed) based on dependencies between known facts. Some facts dominate others in the choice of what to say. If there is only one move on the board having a certain significance, say "threat", then the move is described by its significance alone, e.g. "you threatened me" without location information, since the reader can infer the locations. Similarly, only the most significant defensive and offensive aspects of a move are described even though all are known.

The resulting text is diverse and of good quality. Although there are awkwardnesses, the immense advantage conferred by using a sophisticated grammar prevails.

MANN AND MOORE'S KDS

Major Modules of KDS

Space precludes a thorough description of KDS, but fuller descriptions are available [Mann and Moore 80], [Mann 79], [Moore 79].

KDS consists of five major modules, as indicated in Figure 2. A Fragmenter is responsible for extracting the relevant knowledge from the notation given to it and dividing that knowledge into small expressible units, which we call fragments or protosentences. A Problem Solver, a goal-pursuit engine in the AI tradition, is responsible for selecting the presentational style of the text and also for imposing the gross organization onto the text according to that style. A Knowledge Filter removes protosentences that need not be expressed because they would be redundant to the reader.

KDS MODULES	MODULE RESPONSIBILITIES
FRAGMENTER	• Extraction of knowledge from external notation • Division into expressible clauses
PROBLEM SOLVER	• Style selection • Gross organization of text
KNOWLEDGE FILTER	• Cognitive redundancy removal
HILL CLIMBER	• Composition of concepts • Sentence quality seeking
SURFACE SENTENCE MAKER	• Final text creation

Figure 2: KDS Module Responsibilities

The largest and most interesting module is the Hill Climber, which has three responsibilities: to compose complex protosentences from simple ones, to judge relative quality among the units resulting from composition, and to repeatedly improve the set of protosentences on the basis of those judgments so that it is of the highest overall quality. Finally, a very simple Surface Sentence Maker creates the sentences of the final text out of protosentences.

The data flow of these modules can be thought of as a simple pipeline, each module processing the relevant knowledge in turn.

The principal contributors to the quality of the output text are:

1. The Fragment and Compose Paradigm: The information which will be expressed is first broken down into an unorganized collection of subsentential (approximately clause-level) propositional fragments. Each fragment is created by methods which guarantee that it is expressible by a sentence (usually a very short one. This makes it possible to organize the remainder of the processing so that the text production problem is treated as an improvement problem rather than as a search for feasible solutions, a significant advantage.) The fragments are then organized and combined in the remaining processing.

2. Aggregation Rules: Clause-combining patterns of English are represented in a distinct set of rules. The rules specify transactions on the set of propositional fragments and previous aggregation results. In each transaction several fragments are extracted and an aggregate structure (capable of representation as a sentence) is inserted. A representative rule, named "Common Cause," shows how to combine the facts for "Whenever C then X" and "Whenever C then Y" into "Whenever C then X and Y" at a propositional level.

3. Preference Assessment: Every propositional fragment or aggregate is scored using a set of scoring rules. The score represents a measure of sentence quality.

4. Hill Climbing: Aggregation and Preference Assessment are alternated under the control of a hill-climbing algorithm which seeks to maximize the overall quality of the collection, i.e. of the complete text. This allows a clean separation of the knowledge of what could be said from the choice of what should be said.

5. Knowledge Filtering: Propositions identified by an explicit model of the Reader's knowledge as known to the reader are not expressed.

The knowledge domain of KDS' largest example is a Fire Crisis domain, the knowledge of what happens when there is a fire in a computer room. The task was to cause the reader, a computer operator, to know what to do in all contingencies of fire.

SYSTEM COMPARISONS

The most striking impression in comparing the two systems is that **they have very little in common.** In particular,

1. KDS has sentence scoring and a quality-based selection of how to say things; PROTEUS has no counterpart.

2. PROTEUS has a sophisticated grammar for which KDS has only a rudimentary counterpart.

3. PROTEUS has only a dynamic, redundancy-based knowledge filtering, whereas the filtering in KDS removes principally static, foreknown information.

4. KDS has clause-combining rules which make little use of conjunctions, whereas PROTEUS has no such rules but makes elaborate use of conjunctions.

5. KDS selects for brevity above all, whereas PROTEUS selects for contrast above all.

6. PROTEUS takes great advantage of fact significance assessment, which KDS does not use.

They have little in common technically, yet both produce high quality text relative to predecessors. This raises an obvious question-- Could the techniques of the two systems be combined in an even more effective system?

There is one prominent exception to this general lack of shared functions and characteristics. Recent text synthesis systems [Davey 79], [Mann and Moore 80], [Weiner 80], [Swartout 77], [Swartoutthesis 81], all include a facility for keeping certain facts or ideas from being expressed. There is an implicit or explicit model of the reader's knowledge. Any knowledge which is somehow seen as obvious to the reader is suppressed.

All of the implemented facilities of this sort are rudimentary; many consist only of manually-produced lists or marks. However, it is clear that they cover a deep intellectual problem. Discourse generation must make differing uses of what the reader knows and what the reader does not know.

It is absolutely essential to avoid tedious statement of "the obvious." Proper use of presupposition (which has not yet been attempted computationally) likewise depends on this knowledge, and many of the techniques for maintaining coherence depend on it as well. But identification of what is obvious to a reader is a difficult and mostly unexplored problem. Clearly, inference is deeply involved, but what is "obvious" does not match what is validly inferable. It appears that as computer-generated texts become larger the need for a robust model of the obvious will increase rapidly.

POSSIBILITIES FOR SYNTHESIS

This section views the collection of techniques which have been discussed so far from the point of view of a designer of a future text synthesis system. What are the design constraints which affect the possibility of particular combinations of these techniques? What combinations are advantageous? Since each system represents a compatible collection of techniques, it is only necessary to examine compatibility of the techniques of one system within the framework of the other.

We begin by examining the hypothetical introduction of the KDS techniques of fragmentation, the explicit reader model, aggregation, preference scoring and hill climbing into PROTEUS. We then examine the hypothetical introduction of PROTEUS' grammar, fact significance assessments and use of the contrast heuristic into KDS. Finally we consider use of each system on the other's knowledge domain.

Introducing KDS techniques into PROTEUS

Fragment and Compose is clearly usable within PROTEUS, since the information on the sequence of moves, particular move locations and the significance of each move all can be regarded as composed of many independent propositions (fragments of the whole structure.) However, Fragment and Compose appears to give only small benefits, principally because the linear sequences of tic-tac-toe game transcripts give an acceptable organization and do not preclude many interesting texts.

Aggregation is also useable, and would appear to allow for a greater diversity of sentence forms than Davey's sequential assembly procedures allow. In KDS, and presumably in PROTEUS as well, aggregation rules can be used to make text brief. In effect, PROTEUS already has some aggregation, since the way its uses of conjunction shorten the text is similar to effects of aggregation rules in KDS.

Preference judgment and Hill climbing are interdependent in KDS. Introducing both into PROTEUS would appear to give great improvement, especially in avoiding the long awkward referring phrases which PROTEUS produced. The system could detect the excessively long constructs and give them lower scores, leading to choice of shorter sentences in those cases.

The Explicit Reader model could also be used directly in PROTEUS; it would not help much however, since relatively little foreknowledge is involved in any tic-tac-toe game commentary.

Introducing PROTEUS techniques into KDS

Systemic Grammar could be introduced into KDS to great advantage. The KDS grammar was deliberately chosen to be rudimentary in order to facilitate exploration above the sentence level. (In fact, KDS could not be extended in any interesting way without upgrading its grammar.) Even with a Systemic Grammar in KDS, aggregation rules would remain, functioning as sentence design elements.

Fact significance assessments are also compatible with the KDS design. As in PROTEUS they would immediately follow acquisition of the basic propositions. They could improve the text significantly.

The contrast heuristic (and other PROTEUS heuristics) would fit well into KDS, not as an a priori sentence design device but as a basis for assigning preference. Higher score for contrast would improve the text.

In summary, the principal techniques appear to be completely compatible, and the combination would surely produce better text than either system alone.

Exchange of Knowledge Domains

The tic-tac-toe domain would fit easily into KDS, but the KDS text-organization processes (not discussed in this paper) would have little to do. The fire crisis domain would be too complex for PROTEUS. It involves several actors at once, several parallel contingencies and no single clear organizing principle. PROTEUS lacks the necessary text-organization methods.

SHARED SHORTCOMINGS

These systems share (with many others) the primitive state of the computer-based discourse-generation art. Their processes are primarily devoted to activities that go without notice among literate people. The deeper linguistic and rhetorical phenomena usually associated with the term "discourse" are hardly touched. These systems make little attempt at coherence, and they do not respond in any way to the coherence (or lack of it) which they achieve. Presupposition, topic, focus, theme, the proper role of inference, implicature, direct and indirect speech act performance and a host of other relevant concepts all go unrepresented. Even worse, the underlying conceptual apparatus in both systems is extremely adhoc and idiosyncratic, severely limiting the possiblities for using general knowledge of the semantics of English.

Despite these deficiencies, the systems produce relatively smooth readable text. They are significant principally as collections of methods which may well survive to become parts of more competent and general-purpose systems of the future.

CONCLUSIONS

PROTEUS and KDS are distinct collections of discourse generation techniques. There is little overlap between them, and no apparent conflict involved in using the techniques in combination. A system which combined the strengths of both would be significantly better than either.

APPENDIX: TEXT SAMPLES FROM THE TWO SYSTEMS

KDS: Fire-Alarm Text for Computer Operators

Whenever there is a fire, the alarm system is started, which sounds a bell and starts a timer. Ninety seconds after the timer starts, unless the alarm system is cancelled, the system calls Wells Fargo. When Wells Fargo is called, they, in turn, call the Fire Department.

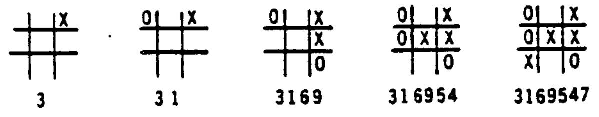
When you hear the alarm bell or smell smoke, stop whatever you are doing, determine whether or not there is a fire, and decide whether to permit the alarm system or to cancel it. When you determine whether there is a fire, if there is, permit the alarm system, otherwise cancel it. When you permit the alarm system, call the Fire Department if possible, then evacuate. When you cancel the alarm system, if it is more than 90 seconds since the timer started, the system will have called Wells Fargo already, otherwise continue what you were doing.

PROTEUS: The move sequence 3169547

Position numbering

1	2	3
4	5	6
7	8	9

The moves:



The game started with my taking a corner, and you took an adjacent one. I threatened you by taking the middle of the edge-opposite that and adjacent to the one which I had just taken but you blocked it and threatened me. I blocked your diagonal and forked you. If you had blocked mine, you would have forked me, but you took the middle of the edge opposite the corner which I took first and the one which you had just taken and so I won by completing my diagonal.

References

[Davey 79]	Davey, Anthony. *Discourse Production.* Edinburgh University Press, Edinburgh, 1979.
[Halliday 76]	Kress, G. R. (editor). *System and Function in Language.* Oxford University Press, London, 1976.
[Huddleston 71]	Huddleston, R. D. *The sentence in written English: a syntactic study based on an analysis of scientific texts.* Cambridge University Press, London, 1971.
[Hudson 71]	Hudson, R. A. *North Holland Linguistic Series.* Volume 4: *English complex sentences.* North Holland, London and Amsterdam, 1971.
[Mann and Moore 80]	Mann, William C., and James A. Moore. *Computer as Author--Results and Prospects.* Research report 79-82, USC/Information Sciences Institute, 1980.
[Mann 79]	Mann, William C. and James A. Moore. Computer Generation of Multiparagraph English Text. 1979. AJCL, forthcoming.
[Moore 79]	Moore, James A., and W. C. Mann. A snapshot of KDS, a knowledge delivery system. In *Proceedings of the Conference, 17th Annual Meeting of the Association for Computational Linguistics*, pages 51-52. August, 1979.
[Sinclair 72]	Sinclair, J. McH. A course in spoken English: Grammar. 1972.
[Swartout 77]	Swartout, William. *A Digitalis Therapy Advisor with Explanations.* Technical Report, MIT Laboratory for Computer Science, February, 1977.
[Swartout 81]	Swartout, William R. *Producing Explanations and Justifications of Expert Consulting Programs.* Technical Report Massachusetts Institute Technology/LCS/TR-251, Massachusetts Institute Technology, January, 1981.
[Weiner 80]	Weiner, J. L. BLAH, A System Which Explains its Reasoning. *Artificial Intelligence* 15:19-48, November, 1980.
[Winograd 72]	Winograd, Terry. *Understanding Natural Language.* Academic Press, Edinburgh, 1972.