Special Issue on Natural Language Generation:
Computational Intelligence 8(1), pages 161-171, 1992

Some Neglected Aspects of the Generation Task

Nigel Ward

Computer Science Division
University of California at Berkeley

Author's current address
Systems Analysis, RCAST, University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153 JAPAN
nigel@tansei.cc.u-tokyo.ac.jp
office +81 (3) 3481-4524
fax +81 (3) 3481-4544
home +81 (3) 5453-4364

subject categories: Cognitive Science

Abstract

Looking to the future, generators will have more knowledge of language and will have to deal with inputs which are very rich in information. As a result, several problems will become more acute, including selecting what to say at the sub-proposition level and dealing with interaction among goals and dependencies among choices. This paper explains how these problems arise and why they are hard to handle within traditional architectures for generation. It also discusses why these issues have not been well addressed; reasons include the current lack of demanding applications, undue respect for linguistic traditions, the use of reverse engineering to determine generator inputs, and the tendency to research only one issue at a time.

keywords: idealized, computational issues, information-rich inputs, goal-conflict, interaction, critique, modularity, natural language generation, language production

Some Neglected Aspects of the Generation Task

The low-level aspects of generation often seem easy. For example, it is not hard to map *'(kill John Mary past)'* to *"John killed Mary"*, and so there are many generators which do well at simple tasks. The purpose of this paper is to discuss some issues which arise when the task is not so simple.

One can foresee that in the future generators will have more world knowledge and more knowledge of syntactic and lexical resources and other aspects of language. They will also have more knowledge about their task: more information in the inputs, and more knowledge of their goals.

This paper examines the nature of the generation task under these conditions, focusing on computational aspects, that is, what a generator needs to do in order to generate well. Unlike previous studies of generation as a computational problem (McDonald 1983, 1987b, 1988; McDonald et al 1987) this paper does not assume any particular computational mechanism; in particular, it does not assume that a generator necessarily makes choices, builds intermediate representations, or is composed of modules.

Thus this paper is intended to lead towards an idealistic, computational, abstract understanding of the generation task: Section 1 identifies some aspects which make generation hard. Section 2 then discusses some aspects of the history and methodology of generation research which have kept these issues from receiving the attention they deserve.

1 Two Characteristics of the Generation Task

Generation is, in the general case, much more complex than is commonly supposed; this section explains. It is by no means an exhaustive description of the issues in generation; in particular, it focuses on problems of the expression of information, leaving out the roles of the situation and the medium, and most of the goal-directed aspects. Discussion of previous generation research is mostly critical; this is not to be negative but only for the sake of illustrating the problems.

1.1 Information-Rich Inputs

Most generators accept inputs which are relatively poor in information, such as feature structures, lists of propositions, logical forms, 'messages', 'realization specifications', and so on. There are reasons for this, as discussed in Section 2. Yet ideally the input to a generator should be 'rich' in information. Figure 1 illustrates the large amount of information which can (and should) be the input causing the generation of a single sentence.

Appropriate outputs for this input might include

1 *When the old woman got to the stream, she was surprised to see a big peach bobbing down towards her.*

[0]Thanks to Robert Wilensky, the anonymous reviewers, and Daniel Jurafsky. This research was sponsored in part by the National Science Foundation, via a Graduate Student Fellowship; by the Sloan Foundation, via the Berkeley Cognitive Science Program; by the Defense Advanced Research Projects Agency (DoD), via the Space and Naval Warfare Systems Command under contracts N00039-84-C-0089 and N00039-88-C-0292; and by the Office of Naval Research under contract N00014-89-J-3205.

genre: folk tale
previous sentence: *"The old woman went to the stream to wash clothes."*
topic of next sentence: describe woman's reaction to peach
role of this sentence in the narrative: introduce the peach into the story
important 'characters': peach and old woman, not the stream
old woman's condition: old, poor
viewpoint the story is being told from: old woman's
attitude towards old woman: sympathetic
attitude towards peach: amazing, later turns out to be enchanted

old woman's destination: stream
old woman's intention: wash clothes, in the stream
method of motion: walk
old woman's expectation: another normal day of chores
familiarity of stream to old woman: very

setting: fairly swift mountain stream, emptying into a pool
location of woman upon arrival: side of stream by pool
event enabling her to see peach: arrival at stream
time of seeing peach: immediately after arrival
result of seeing peach: old woman becomes surprised
specific surprising fact: enormous size of peach
prior probability of a peach being greater than 20 cm in diameter: 0

initial location of peach: in the stream, visible above water, upstream of the woman
direction of peach motion: downstream, coming closer to the woman
manner of peach motion: bobbing up and down and from side to side
cause of peach motion (floating, bobbing and moving): all due to current of stream

aspect of floating in a stream: continuous action
content of stream: water
speed of current in mountain streams: swift
depth of mountain streams: shallow
density of peaches: lighter than water
inanimate(peach)

Figure 1: A Sample Input

2 *She got to the stream, and, upstream, there was an enormous peach! bobbing down towards her.* (better if read aloud with dramatic pauses)

3 *She got to the stream, and upstream, to her great surprise, there was an enormous peach bobbing along.*

Producing these sentences, rather than something similar but wrong, can depend on nuances of meaning. To point out just one aspect of this, *"a peach surprised her"* is inappropriate given that the woman *saw* the peach, rather than hearing or bumping into it. Another example where word choice depends on a nuance is *"in"* versus *"on"*, as in *"eating ice cream on the bus"*, versus *"eating ice cream in the taxi"*, where the determining factor is whether the mode of transportation is running on a fixed route (Fillmore 1985). The danger of inaccurate choice due to the lack of a rich input increases, of course, with the number of lexical and syntactic resources available to the generator.

One might object that the task of a generator is merely to 'express the facts', and that it is not important for it to accurately convey nuances of meaning. But in fact these nuances can be helpful to a reader. For example, in the text

4 *There are 10,635,264 words available if you elect not to garbage collect.*

"if you elect not to" instead of *"if you do not elect to"* or *"unless you"* reflects the fact that to garbage collect is the normal choice. *"Available"* instead of nothing suggests that *"words"* are a resource for the user.

Thus a generator needs access to a lot of information in order to make correct word choices. It may not always use it all — the specific target language may not be sensitive to some of it — but the information must be there just in case.

It is true that there is currently no application which requires (or even allows) a generator to deal with such rich inputs. Yet this will change as the programs which interface with generators become more knowledge-intensive. For example, imagine an advanced expert system with a complete causal model, that is, a full understanding of its domain, not just a set of if-then rules. For a generator to express the rationale behind some conclusion, it will probably need access to the entire working memory and perhaps to the entire knowledge base of the system. As another example, imagine an advanced parser/analyzer, able to infer all the implicit information in a text, including the background knowledge of the writer and the nuances the reader should pick up. To ensure a faithful expression of the result into, say, French, a generator should have access to of all that information.

Indeed, a generator should be a process which shares memory and knowledge with other processes. Most generators, at least as implemented, are modules which operate in isolation; they are passed a few symbols as parameters and have no other access to the knowledge of the system. Indeed, the very word 'input' is inappropriate to the extent that it suggests that a generator can operate autonomously (but I will continue to use the term 'information-rich input').

One problem which arises if a generator can accept information-rich inputs, is that not every fact in the input need appear explicitly in the output, because of the hearer's inference ability. For example,

5 *she saw a peach floating in the stream,* being moved by the current, and moving downstream

is redundant in that the information given by the words in bold is inferrable from the first clause. It is well-known that the task of a parser/analyzer is not merely to 'extract' some meaning already present in the utterance, but rather to take the utterance as a sequence of clues which provoke and guide the formation of an interpretation. Conversely, in generated text "the content explicitly expressed ... constitutes a dotting of disconnected islets of information. The fully continuous 'sea' exists in the minds of the intercommunicators as a skein of knowledge and familiarities, presuppositions and expectations, presumptions and deductions. ... the manifest communication ... has interpretable significance only within the context" of the sea of information (Talmy 1976).

This issue of selectivity is a serious one. Existing generators are largely based on the idea that in the normal case the task is to map directly from input elements to output words. Even systems which do choose content to some extent do not really face up to this issue squarely. One problem is that there is very little use of target language knowledge in this process, as discussed in the next subsection. Another problem is that choice of what to say is only done at the proposition level (with some exceptions (Hasida *et al.* 1988; Reiter 1991)).

That is, approaches deciding 'what to say' typically treat inputs rich in propositions but still information-poor within propositions, and so the problem at the finer grain, as illustrated in examples 1-3, does not arise.

It is worth noting that this, the problem of having more information available than should be explicitly expressed, can arise even for information-poor inputs, at least for generators which have, as they should, the power to augment their 'inputs' by inference at run-time. This inference ability can be needed in order to 'realize' that a certain word is applicable to describe a situation (Hovy 1988). For example, from 'the peach was upstream of the woman and moving downstream' a generator should be able to infer that 'the peach was approaching the woman', and thus that words like *"approach"* and *"come"* may be appropriate. Such a generator has a 'virtual' rich input, and thus also faces the problem of selecting information to express explicitly.

1.2 Multiple Interacting Goals

A generator must strive to simultaneously achieve many goals, and these goals may be in conflict. This is most obviously true for pragmatic goals (Hovy 1988); for example the goal of not offending the hearer can conflict with the goal of conveying all the intended information. Some conflicts are almost intrinsic to the generation task; for example, the goals of being clear and unambiguous inherently conflict with the goal of being concise (Meteer 1990).

Even at a more mundane level there is goal conflict. Perhaps the two most basic goals of generation are 'express the input' and 'express it in the target language'. Most descriptions of the generation task conflate these, but considering them as distinct goals makes it easier to discuss cases where a speaker must make trade-offs between them, that is, cases where a generator must choose between being faithful to the original intention and sounding natural and fluent.

Examples where the speaker or author insists on expressing his exact meaning, riding roughshod over the language, are rare enough to be noteworthy. Heidegger's philosophical writings are the only example that comes to mind; these are notoriously unreadable. (Many generators, most obviously input-driven generators (McDonald *et al.* 1987), straightforwardly express their input, and run into no problems of goal conflicts. This only works, however, because their inputs are fairly close, in various ways, to the target language sentences they are expected to lead to. Cases where this is not true, such as generation in the context of machine translation, whose inputs are the representations output by a parser of another language, make this clear: the outputs of such generators tend to be unnatural, at best.)

Thus there should be a give-and-take between and language and thought; between the medium and the message. This is hard to show: Whorfian effects are hard to illustrate, since it is seldom possible to determine the content of thought other than by studying its expression in language.

But a simpler effect is easier to demonstrate. There is a continuum between things said because the speaker wants to say them and things said because the language facilitates or forces their inclusion. That is, the effort to meet the goal of producing a natural sentence, has effects on what information gets conveyed. For instance, it requires more effort to specify the sex of a person in Japanese than in English, where that information can be encoded concisely on a pronoun. In Japanese you refer to a person without specifying the sex unless it's important;

in English you don't leave it out unless it's important to do so. Experiments confirm that speakers of different languages describing the same scene tend to include the information that is concisely expressible in their language (Slobin 1987), and, for the sake of natural output, computers should do the same.

Many systems do provide for some interplay between the 'faithful' and 'natural' goals, in that they are allowed to take turns being in control. In other words, it is common for some of the generator's decisions to be governed by the language and the rest by the content to be expressed. Examples of this include choosing a verb for semantic reasons, and then letting its case frame determine which concepts from the input to realize, or choosing a rhetorical structure and letting that determine which propositions to include.

But interleaved control still does not make it possible to make trade-offs sensibly; to maximize the overall value of the output requires both goals to be active at once. And not just these two goals, given that the language at hand can cause conflicts to arise between pragmatic goals. For example, if a certain word is likely to remind the hearer of an unhappy past event, one may wish to avoid using that word even at the cost of not perfectly expressing the desired nuance. As another example, English has post-verbal particles which are convenient for expressing directional information, as in *"come straight back down from up there"*. In many languages such information could not be expressed so as to make it clearly only background information, and so conflicts could arise between the goals of expressing something but of not exaggerating its importance. (From the other point of view, if the language being generated into is English there is a rare opportunity to concisely achieve both goals.) In planning terms, constraints on the co-occurrence and ordering of plans give rise to goal conflicts (Appelt 1985). To put it differently, the goal of naturalness affects the ability to simultaneously achieve other goals. To put it still differently, higher-level goals cannot be traded-off correctly without using knowledge of what the target language options are.

Consider the fact that purpose clauses with *"to"* take actions not states, as in *"go to the hills to gather wood"* versus **"go to the hills to have wood"*. This low-level syntactic fact can affect what can be expressed compactly in one sentence. This can also be described in terms of goals and goal conflict; one can consider each 'part' of the input as giving rise to the goal of expressing it. A generator must satisfy these goals while still achieving the orthogonal goal of producing a valid utterance by respecting the dependencies among syntactic and lexical choices.

Interaction among various decision-making processes is needed for these fine-grained goals too, but, again, most models of generation have little place for interaction. Most are based on the assumption that, in the normal case, each of the various decisions involved in the process of generating a sentence can be made in isolation. Such generators map each individual portion of the input onto a word or construction, ignoring questions of compatibility among the choices. Generators which do this can be said to be 'compositional'. Of course no system is strictly compositional, but interaction is typically limited to that which fits into a serial generation algorithm.

In future generators will have more lexical and syntactic options available. If so, there will be many options for expressing each 'part of the input', and the danger of these being incompatible with options for expressing 'nearby' parts will become more serious. In other words, a generator able to use many words and syntactic constructions will have to seriously deal with dependencies among choices. Moreover, addition of new types of knowledge about language will bring new subgoals of the naturalness goal, which, since they interact with each other, will make dependencies even more of an issue. For example a generator should avoid accidental alliteration and duplication, as in *"I like most animals, but I can't bear bears"*; this of course interacts with other considerations in lexical choice. Another thing the ideal generator will deal with is prosody, including intonation contours; this also interacts with lexical choice, since the stress patterns and lengths of words must allow the desired overall intonation, and with clause structure, clause length and sentence type, as in the choice between a declarative statement and a rhetorical question (Cutler & Isard 1980).

Any one of the types of dependencies among choices is easy to handle, but handling all of them together is difficult — an architecture issue.

This leads to the question of modularity. Most generators get away with little interaction, for reasons discussed below. Generators which do allow more interaction often employ special mechanisms, for example, allowing one module to send a message, query, or request to another (Hovy 1988; Nirenburg *et al.* 1988; Finkler & Neumann 1989; Reithinger 1991). This style of computation may be adequate if interaction is a rare event, but the overhead appears extreme if interaction is pervasive, as seems to be required. Indeed, if a great deal of communication among modules is required, there is little point in having modules at all.

One popular way of modularizing the generation process is to separate very language-y tasks from less language-y ones. Proposals in this vein include separating what-to-say from how-to-say-it (Goldman 1974), strategy from tactics (Thompson 1976), and planning from (linguistic) realization (McDonald 1987a). The first stage is perceived as having the task of reducing information-rich inputs to simple 'messages' (or 'meanings', 'contents', or 'realization specifications'), containing exactly and only the things to be expressed in the output. The real generator, that is, the part that has knowledge about language, only receives this information, can access no more, and has no recourse to world knowledge or inference capability.

Yet, as many researchers have pointed out, and as suggested in the previous subsection, no such 'message' can contain enough information, since even seemingly low-level language decisions can depend on seemingly irrelevant information, or on the speaker's goals (Danlos 1984; Appelt 1985; Hovy 1988). Conversely, the process responsible for choosing conceptual content is also in trouble if such a separation is used; it is then denied necessary information about the resources available in the language at hand, as illustrated above and also pointed out elsewhere (McDonald 1988; Danlos 1987).

The extent to which modularity is possible or appropriate given current technology is not resolved (McDonald 1988). However it seems clear that, as inputs become more complex and generators' knowledge increase, modular designs will become less workable.

1.3 Where Will the Solutions Lie?

These problems being unsolved, this subsection can do no more than speculate on some probable components of solutions. The use of numeric weighting of preferences seems necessary in order to perform trade-offs among interacting goals. Uniform or at least generally accessible representation of the generator's working memory, rather than use of various layers of representation or working memories private to specific modules, seems necessary, again for the sake of interaction (Appelt 1985).

The biggest need seems to be for parallelism: the need for interaction suggests that goals and knowledge sources be active in parallel, and interaction and information-rich inputs both suggest parallelism for speed. This raises at least two questions: First, where and how to use parallelism; here there have been several discussions and proposals (Stemberger 1985; Finkler & Neumann 1989; De Smedt 1990; Meteer 1990). Second, how to allow parallelism, given that most generator designs are fairly intrinsically serial. This problem is far from solved; current parallel implementations can only be described as messy. Discussions of and proposals for parallelism in generation are surveyed at length in (Ward 1991), which also points out some further reasons for parallelism and proposes a highly parallel architecture.

2 Why Previous Research has Missed the Point

Although the issues raised in the previous section are not critical for current generators, it is still curious that they have received so little direct attention. This section discusses some tendencies in the generation field which have led to this neglect. This section is not intended to be a balanced description of the history or current status of generation research; more comprehensive accounts include (McDonald 1987a) and (Kempen 1989).

2.1 Linguistic Heritage

The need for information-rich inputs is not generally acknowledged; indeed, impoverished inputs are assumed without question by most generation researchers. One reason for information-poor inputs is the fact that, for most applications, the input only needs to contain enough information to select the right canned text, govern choices of the variables in the template (eg *"rise"* versus *"fall"*), and perhaps to control syntactic and morphological variation (eg singular versus plural). Deeper reasons for the prevalence of information-poor inputs lie in the historical development of generation research.

Early generators were showcases of syntactic theories. Indeed, the very word 'generate' suggests a mathematical operation to produce all and only the sentences of a language from a finite description, attesting to the origins of the field. To demonstrate the coverage of a generation mechanism it is necessary to make it exhibit variation. One way to do this is to have it choose among options more or less randomly (Friedman 1969). A better way is to determine what governs the variation (Reich 1970), and call this information the 'input' to the generator, even though it really does no more than specify the behavior of a generator. Typically such 'inputs' contains exactly enough information to specify one sentence of a specific language; this is most obviously true for the inputs to systemic generators (Davey 1978; Mann 1983).

Another important early influence on the field of generation was the idea of a 'deep structure' underlying and prerequisite to the surface realization. One can convert this to surface form or, using a different metaphor but not necessarily different technology, adopt an 'input-driven strategy'. In input-driven generators also, inputs straightforwardly control behavior. For example, traversal of a downward link might cause a subroutine call, and arriving at a node for a verb-concept might lead to unifying slots with cases. In general it is common for the structure of the input to directly determine the sequence and constituency of the text, as McDonald has observed (McDonald *et al.* 1987). Deep-structure-like inputs have the advantage that they can somewhat more plausibly be considered to be 'meaning'.

If, for either reason, a generator is directly controlled by its input, it must receive an input in exactly the expected format. This might involve having: a distinguished top node, as a place to begin traversal; a hierarchical structure, for recursive decomposition; a target-language-friendly inventory of concepts, so that the mapping from input concepts to output words can be one-to-one; or particular structural configurations, to enable matching to syntactic structures or word meanings. Requiring such properties seems an unreasonable constraint for a generator to impose on the programs that use it.

It should not be necessary to rigidly prescribe the format of the generator's input; a generator should be robust in the face of variations in the structure or conceptual vocabulary of the input. In slogan form, a generator should be flexible. This does not, of course, require that a generator needs to be made specifically flexible. If a generator accepts information-rich inputs, as it must, or if it can do inference, as it should, the possibility of simply mapping input to output is clearly absurd, so any good generator will necessarily be a 'flexible generator'.

2.2 Reverse Engineering

Perhaps the deepest underlying reason for the acceptance of information-poor inputs is the common practice of 'reverse-engineering' sentences to find out what goes into them, and then making that be the input to the generation process (some generation research relies on reverse-engineering done previously by linguists). There is nothing wrong with this in principle, but the process is never carried far enough; it typically stops far short of 'information-rich inputs'.

Reverse engineering leads to information-poor inputs in another way; if all the inputs analyzed are taken from the same task or domain, this tends to lead to generators which use implicit, built-in knowledge about that domain. McDonald (1987a) and Meteer (1990) show how many generators produce impressive output only because they presume information which is not explicit in the input. While this is not a problem if the generator is intended to be used for one specific application, this is clearly not the way to build general-purpose systems; they will require inputs replete with explicit information.

Another problem is that inputs based on reverse-engineered sentences of one language are clearly not useful for generating into other languages. (This points up one advantage of studying generation in the context of machine translation: it makes it easier to avoid the temptation to start from inputs which are thinly disguised versions of the desired outputs.)

The possibility of ascribing a 'meaning' to an utterance cannot be taken to imply that that 'meaning' was the input to the generation process. For one thing, the words included in, and hence the 'meaning' of, an utterance depend on the language which the speaker happens to be producing at the time. Reverse engineering is guaranteed to fail to reveal problems of goal interaction, including in particular the effect of naturalness considerations on the meaning conveyed.

The information-poor inputs typically inferred by reverse engineering are congenial to those who want a concise answer to the question of the relation between the generator's input and its output. The answer is typically in terms of the metaphor of 'mapping' the input into the output, which again leads to the idea of an input with no more symbols that the output. It is reasonable to want a specification of the generation task before one builds a system for it (and indeed this is a purpose of this paper), but to describe the generation task by formally specifying the relationship between input and output seems to deflect attention from the real issues more than to clarify the nature of the problem. To use Marr's terminology it seems that generation is a 'Type 2' task: one where the "problem is solved by the simultaneous action of a considerable number of processes, whose interaction is its own simplest description" (Marr 1977).

There is a clear alternative to reverse-engineering, namely choosing an independently motivated input format and then working from that, but this is also not risk-free. Given a set of inputs there is a tendency to build a generator able to handle those inputs, but not necessarily capable of handing different types of inputs. Descriptions of such generators tend to degenerate into discussions of how they deal with specific problems raised by the specific representation languages they use, such as Conceptual Dependency's decomposition of verb meanings into primitives (Goldman 1974), or NIKL's handling of definitions (Sondheimer *et al.* 1990). Most generators used in machine translation are degenerate in this way — they tend to focus narrowly on problems which arise when trying to express an input which comes from a specific parser for a specific source language.

2.3 Single-Issue Research

The general lack of recognition of the need for interaction is doubtless due in part to the fact that each researcher typically focuses on only one dimension of variation. For example, (to oversimplify) BABEL's main task was to choose among words with some common element of meaning (Goldman 1974), and PENMAN's forte (Mann 1983) was ordering words and choosing among syntactic options. When studying only one issue, the problems of conflicting goals and of dependencies among diverse kinds of choices obviously do not arise.

A historical anecdote can illustrate the danger of researching generation one module at a time. The original strategy/tactics distinction was proposed by Thompson to justify addressing certain syntactic aspects of generation in isolation; these he called the "tactical aspects" (Thompson 1976). By "strategic aspects" of generation he meant everything else, including word choice and the interface of thought to language. Later McKeown addressed the problem of 'strategic generation' as that of determining "the content and structure of the discourse" (McKeown 1985), and worked on some issues in text above the clause level, touching on only a fraction of the problems set aside by Thompson. Yet the terminology suggests, and many researchers seem to believe, that the task of generation has been exhaustively decomposed, and that the current bifurcation of generation research is satisfactory.

The tendency to do single-issue research is reinforced by the goal of modularity. In general this is a valid design objective — if modules have only limited interactions with each other, they can be built one by one, and the system as a whole is easier to extend and understand. Most generators are very modular (McDonald 1988): there are, for example, commonly divisions between thinking and generating, between deciding what-to-say and deciding how-to-say-it, and between syntactic choice and word choice. But in generation the convenience of modularity comes at the price of restricting interaction, and thereby risking poor quality output, as discussed above.

While it is reasonable to partition the *problem* in order to study it, one must take care not to define away important problems, such as interaction between syntactic and other considerations. It is fair to focus one's research on a single issue, but one can not assume that that issue should be handled by an autonomous module. That is, divisions of the task which are proposed for the sake of organizing research should not be imposed on designs for generation.

3 Hopes

Clarifying the nature of the generation task, the aim of this paper, is only a first step. On the theoretical side, the next step is to reason from the issues discussed in Section 1 to design principles which a generator should embody; preliminary attempts at this include (Ward 1989, 1990, 1991). Armed with such principles it will hopefully be clearer how to advance the practical side of the field, to build generators able to perform demanding tasks well.

References

Appelt, Douglas E. (1985). *Planning English Sentences.* Cambridge University Press.

Cutler, A. & S. D. Isard (1980). The Production of Prosody. In Brian Butterworth, editor, *Language Production, Volume 1: Speech and Talk*, pp. 245–269. Academic Press.

Danlos, Laurence (1984). Conceptual and Linguistic Decisions in Generation. In *Proceedings 10th COLING*, pp. 501–504.

Danlos, Laurence (1987). *The Linguistic Basis of Text Generation.* Cambridge University Press.

Davey, Anthony (1978). *Discourse Production.* Edinburgh University Press.

De Smedt, Koenrad J.M.J. (1990). Incremental Sentence Generation: a computer model of grammatical encoding. Technical Report 90-01, Nijmegen Institute for Cognition Research and Information Technology.

Fillmore, Charles J. (1985). Frames and the Semantics of Understanding. *Quaderni di Semantica*, IV(2).

Finkler, Wolfgang & Günter Neumann (1989). POPEL-HOW: A Distributed Parallel Model for Incremental Natural Language Production with Feedback. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 1518–1523.

Friedman, Joyce (1969). Directed Random Generation of Sentences. *Communications of the ACM*, 12(6):40–46.

Goldman, Neil (1974). *Computer Generation of Natural Language from a Deep Conceptual Base.* PhD thesis, Stanford University.

Hasida, Kôichi, Shun Ishizaki, & Hitoshi Isahara (1988). An Approach to Abstract Generation. *Bulletin of the Electrotechnical Laboratory*, 52(4):551–564.

Hovy, Eduard (1988). *Generating Natural Language Under Pragmatic Constraints.* Lawrence Erlbaum Associates.

Kempen, Gerard (1989). Language Generation Systems. In István S. Bátori, Winfried Lenders, & Wolfgang Putschke, editors, *Computational Linguistics: An International Handbook*, pp. 471–480. Walter de Gruyter, Berlin and New York.

Mann, William C. (1983). An Overview of the Penman Text Generation System. In *Proceedings of the Third National Conference on Artificial Intelligence*, pp. 261–265.

Marr, David (1977). Artificial Intelligence — A Personal View. *Artificial Intelligence*, 9:37–48.

McDonald, David D. (1983). Natural Language Generation as a Computational Problem. In Michael Brady & Robert Berwick, editors, *Computational Theories of Discourse*, pp. 209–255. MIT Press.

McDonald, David D. (1987a). Natural-Language Generation. In Stuart C. Shapiro, editor, *Encylopedia of Artificial Intelligence, Volume 1*, pp. 642–655. Wiley.

McDonald, David D. (1987b). Natural Language Generation: Complexities and Techniques. In Sergei Nirenburg, editor, *Machine Translation*, pp. 192–224. Cambridge University Press.

McDonald, David D. (1988). Modularity in Natural Language Generation: Methodological Issues. In *Proceedings of the AAAI Workshop on Text Planning and Realization*, pp. 91–98.

McDonald, David D., Marie Vaughan, & James Pustejovsky (1987). Factors Contributing to Efficiency in Natural Language Generation. In Gerard Kempen, editor, *Natural Language Generation*, pp. 159–181. Nijhof. distributed by Kluwer.

McKeown, Kathleen R. (1985). Discourse Strategies for Generating Natural-Language Text. *Artificial Intelligence*, 27(1):1–41.

Meteer, Marie W. (1990). The “Generation Gap”: The Problem of Expressibility in Text Planning. Technical Report 7347, BBN Systems and Technologies Corporation, Cambridge MA.

Nirenburg, Sergei, Rita McCardell, Eric Nyberg, Philip Werner, Scott Huffman, Edward Kenschaft, & Irene Nirenburg (1988). DIOGENES-88. Technical Report CMU-CMT-88-107, Carnegie Mellon University, Center for Machine Translation.

Reich, Peter A. (1970). *A Relational Network Model of Language Behavior.* PhD thesis, University of Michigan.

Reiter, Ehud (1991). A New Model for Lexical Choice for Open-Class Words. *Computational Intelligence*, 7(4):240–251.

Reithinger, Norbert (1991). POPEL — A Parallel and Incremental Natural Language Generation System. In Cécile L. Paris, William R. Swartout, & William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pp. 179–199. Kluwer.

Slobin, Dan I. (1987). Thinking for Speaking. In *Berkeley Linguistics Society, Proceedings of the Thirteenth Annual Meeting*, pp. 435–445.

Sondheimer, Norman, Susanna Cumming, & Robert Albano (1990). How to Realize a Concept: Lexical Selection and the Conceptual Network in Text Generation. *Machine Translation*, 5:57–78.

Stemberger, Joseph Paul (1985). An Interactive Activation Model of Language Production. In Andrew W. Ellis, editor, *Progress in the Psychology of Language, Volume 1*, pp. 143–186. Lawrence Erlbaum Associates.

Talmy, Leonard (1976). Communicative Aims and Means — A Synopsis. Technical Report 20, Working Papers on Language Universals, Stanford University.

Thompson, Henry (1976). Towards a Model of Language Production: Linguistic and Computational Foundations. *Statistical Methods in Linguistics.* (a journal published by Sprakforlagel Skriptor, Stockholm).

Ward, Nigel (1989). Capturing Intuitions about Human Language Production. In *Program of the Eleventh Annual Conference of the Cognitive Science Society*, pp. 956–963. Lawrence Erlbaum Associates.

Ward, Nigel (1990). A Connectionist Treatment of Grammar for Generation: Relying on Emergents. In *Proceedings, Fifth International Workshop on Natural Language Generation.* a revised version of a paper in the *Program of the Twelfth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, 1990, pp 380-387.

Ward, Nigel (1991). *A Flexible, Parallel Model of Natural Language Generation.* PhD thesis, University of California at Berkeley, Computer Science Division.