Natural Language Generation

John Bateman and Michael Zock

The Oxford Handbook of Computational Linguistics (1 ed.)
Edited by Ruslan Mitkov

Print Publication Date: Jan 2005 Subject: Linguistics, Computational Linguistics
Online Publication Date: Sep 2012 DOI: 10.1093/oxfordhb/9780199276349.013.0015

Abstract and Keywords

Communication via a natural language requires two fundamental skills, producing text and understanding it. This article introduces the field of computational approaches to the former-natural language generation (NLG) showing some of the theoretical and practical problems that linguists, computer scientists, and psychologists have encountered when trying to explain how language works in machines or in their minds. The corresponding task of NLG spans a wide spectrum: ranging from planning some action to executing it. Providing architectures in which all of these decisions can be made to coexist, while still allowing the production of natural sounding texts within a reasonable amount of time, is one of the major challenges of NLG. Another challenge is ascertaining just what the decisions involved in NLG are. This article overviews the cognitive, social and linguistic dimensions of NLG and finally opens issues and problems related to the field.

Keywords: communication, natural language generation, machines, planning, executing, spectrum, architectures

Abstract

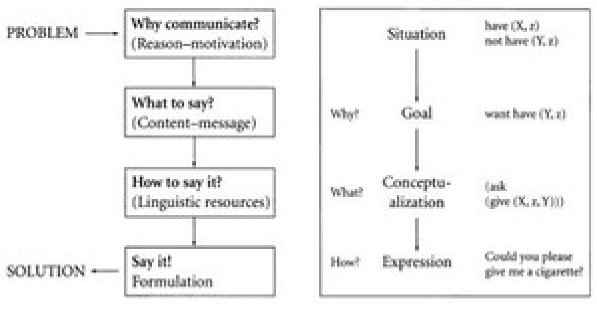
Communication via a natural language requires two fundamental skills, producing ‘text’ (written or spoken) and understanding it. This chapter introduces newcomers to the field of computational approaches to the former-natural language **generation** (henceforth NLG)-showing some of the theoretical and practical problems that linguists, computer scientists, and psychologists have encountered when trying to explain how language works in machines or in our minds.

(p. 285) 15.1 General Introduction: What is Natural-Language Generation? (Cognitive, Linguistic, and Social Dimensions)

15.1.1 Natural Language Generation—a knowledge-intensive problem

Producing language is above all a social activity. We speak to solve problems, for others or for ourselves, and in order to do so we make certain choices under specific space, time, and situational constraints. The corresponding task of NLG therefore spans a wide spectrum: ranging from planning some action (verbal or not) to executing it (verbalization). Hence, information has to be mapped from some *non-linguistic* source (e.g. raw data from a knowledge base or scene) into some corresponding linguistic form (text in oral or written form) in order to fulfil some non-linguistic goal(s). This ‘transformation’ is neither direct nor straightforward, and bridging the gap between the non-linguistic ‘scene’ and its linguistic counterpart involves many decisions or choices: these include determination and structuring of *content* (i.e. choice of the message) and *rhetorical structure* at various levels (text, paragraph, sentence), choice of the appropriate *words* and *syntactic structures* (word order, morphology, and grammatical constructions), and determination of text layout (title, headers, footnotes, etc.) or acoustic patterns (prosody, pitch, intonation contour). Providing architectures in which all of these decisions can be made to coexist, while still allowing the production of natural sounding/reading texts within a reasonable amount of time, is one of the major challenges of NLG. Avery simple, first view of this process and its stages is suggested in Fig. 15.1..

Another challenge is ascertaining just what the decisions involved in NLG are. NLG requires many kinds of expertise: *knowledge of the domain* (what to say, relevance), *knowledge of the language* (lexicon, grammar, semantics), *strategic rhetorical knowledge* (how to achieve communicative goals, text types, style), etc. Moreover, building successful NLG systems requires *engineering knowledge* (how to decompose, represent, and orchestrate the processing of all this information) as well as *knowledge* about the *habits* and *constraints* of the end user as an *information processor* (sociolinguistic and psychological factors).


PROBLEM
Why communicate? (Reason–motivation)
What to say? (Content–message)
How to say it? (Linguistic resources)
Say it! Formulation
SOLUTION
Situation
have (X, z) not have (Y, z)
Why?
Goal
want have (Y, z)
What?
Conceptu-alization
(ask (give (X, z, Y)))
How?
Expression
Could you please give me a cigarette?

Click to view larger

Fgure 15.1 **NLG: a simple view**

While everybody speaks a language, not everybody speaks it equally well. There are substantial differences concerning its speed of learning, and its ease and success of use. How language works in our mind is still a mystery, and some consider the construction of NLG systems as a methodology for helping to unravel that mystery. Others see NLG as an approach to solving practical problems-such as contributing (p. 286) to the synthesis side of *machine translation* (cf. Chapters 27, 28), to *text summarization* (cf. Chapter 32), and to *multilingual* and *multimodal presentation of information* (cf. Chapters 36 and 38) in general. That our understanding of the process remains so poor is largely due to the *number*, the *diversity*, and the *interdependence* of the choices involved. While it is easy to understand the relationships between a glass falling and its breaking (causal relationship), it is not at all easy to understand the dependency relationships holding in heterarchically organized systems like biology, society, or natural language. In such systems, the consequences of a choice may be multiple, far reaching, and unpredictable. In this sense, there are many points in common between speaking a language well, hence communicating effectively, and being a good politician. In both cases one has to make the right choice at the right moment. Learning *what* the choices are, and *when* they should be made, is then a large part of the task of building NLG systems.

15.1.2 What is language? A functional perspective

Much of the complexity of NLG arises out of the fact that producing language is a knowledge-intensive, flexible, and highly context-sensitive process. This context sensitivity reveals itself best when we consider connected texts rather than isolated sentences. Consider the following example. Suppose you were to express the idea: [LEAVE (POPULATION, PLACE)]. It is instructive to watch what happens if one varies systematically the expression of the different concepts *leave, population*, and *place* by using either different *words (abandon, desert, leave, go away from* in the case of the (p. 287) verb, and *place* or *city* in the case of the noun), or different *grammatical resources:* e.g. a *definite description* ('the + *N'), possessives* ('yours', 'its'), etc. Concretely, consider the alternatives given in (15.1).

(15.1) 'X-town was a blooming city. Yet, when hooligans started to invade the place, [insert one of: *(a)-(e)* below]. The place was not livable any more.'

(a) the *place* was abandoned by (its/the population)/them.
(b) the *city* was abandoned by its/the population.
(c) was abandoned by its/the population.
(d) its/the population abandoned the *city*.
(e) its/the population abandoned *it*.

The interested reader may perform all the kinds of variations mentioned above and check to what extent they affect *grammaticality* (the sentence cannot be uttered or finished), *clarity* (some pronouns will create ambiguity), *cohesion*, and *rhetorical effect*. In particular, while all the candidate sentences we offer in *(a)-(e)* are basically well-formed, each one has a specific effect, and not all of them are equally felicitous. Some are ruled out by virtue of poor textual choices (e.g. in (a) *'the place'*is suboptimal, since it immediately repeats a word), others because of highlighting the wrong element, or because of wrong assignment of the informational status (given-new) of some element (e.g. in *(d) 'the city'* is marked as'minimal' *new* information, while actually it is known, i.e. *old* information). Probably the best option here is *(c)*, since this preserves the given-new distribution appropriately, without introducing potentially ambiguous pronouns (see Chapter 14 for detailed discussion of anaphora).

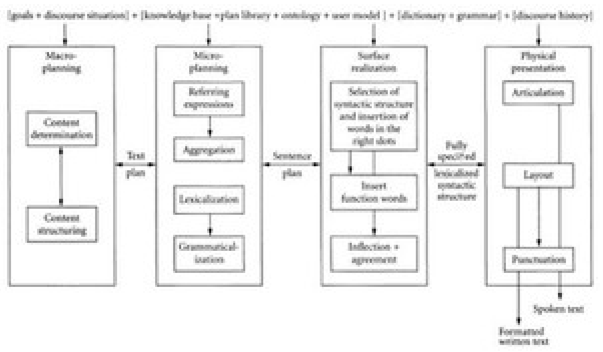
Getting a text ('right' is therefore a major problem. Central here is the notion of the *effects* of individual linguistic choices-not only locally, but also globally for the text to which they are contributing. Both choices and effects have to be orchestrated so as to achieve collectively the speaker's overall goals. The notion of 'grammaticality', central to formal approaches of language (see, particularly, Chapters 4 and 8) is by no means sufficient and many other factors, such as social, discourse, and pragmatic constraints (Chapters 6 and 7), have to be taken into account in order to achieve *successful communication*. Many of these latter factors have traditionally been at the heart of functional theories of language (e.g. Halliday 1994), and this is one of the reasons why such theories have been used widely by the NLG community—far more widely than has been the case for natural language understanding.

15.1.3 Decomposing the NLG task: Architectures for NLG

Architectures deal with the functional relations between components (dependencies, control of information flow) or the problem of *what* to process *when* (order). The sheer range of interdependencies found in language has motivated many different kinds of architecture (sequential, parallel, integrative, revision based, blackboard, **(p. 288)** connectionist). They have also led to differing conceptualizations of what NLG is, resulting in at least three kinds of definition:

1. NLG as a *mapping problem*
2. NLG as a *problem of choice*
3. NLG as a *planning problem*

All of these involve some decomposition of the problem space into different kinds of representational layers. The most common are those shown in Fig. 15.2, which divides the whole process into four broad tasks (macroplanning, microplanning, linguistic realization, presentation) and a series of subtasks. At the top of the figure there are recurring situational constraints and knowledge sources (e.g. dictionary, grammar) that can apply at various stages in processing; the boxes below represent the (sub) tasks at the various levels. These four main tasks may be described as follows:



Click to view larger

Fig.15.2 **Commonly assumed components**

• Macroplanning. Given some goal and knowledge sources (raw data), choose among them and organize them to build a text plan. This generally yields a tree composed of a set of leaves (messages to be expressed, generally sentences or clauses) and a set of nodes and arcs expressing the type of *rhetorical relation* (cause, sequence, etc.) and the textual status of the segment *(nucleus, satellite:* see below).

(p. 289) • Microplanning. Decide on how to describe any given event or entity, i.e. give enough information to allow its *discrimination from possible alternatives* (reference); group closely related material by *factoring out redundant elements* in order to build better integrated, more concise text (aggregation); and *select appropriate words* for expressing the chosen content and for achieving the necessary degree of cohesion (lexicalization, pronominalization, and choice of cue words). In practice (and perhaps in theory too), it is not possible to separate cleanly selection of lexical items and commitments to particular grammatical organizations.

• Surface realization. Impose selected *grammatical constructions* (entailing selection of syntactic functions: subject, direct object, etc.), *linear order* (constrained either by the grammar or by information status as a language requires), *part of speech* (noun, verb, etc.), *sentence complexity* (simple/ compound), insert *function words* (e.g. prepositions), and determine the *final form* of the words (morphology).

• Physical presentation. Perform final *articulation, punctuation*, and *layout operations*.

15.2 Content Selection and Discourse Organization: Macroplanning

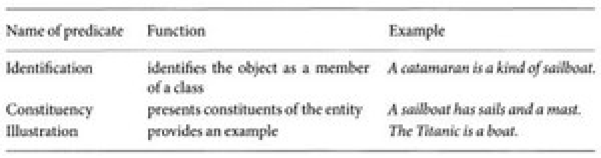
15.2.1 Content planning

Before starting to talk, one should have something to say. The crux of the problem is to find out how this 'something'—the conceptual input to be expressed by the surface generator—is determined. Obviously, given some topic, we will not say everything we know about it, neither will we present the messages in the order in which they come to mind; we have to make some choices and we have to perform certain rearrangements. But what guides these operations?

Suppose you had the task of writing a survey paper on NLG or a report on the weather. You could start from a knowledge base containing information (facts) on *authors, type of work, year*, etc. or *meteorological information* (numerical values). Since there is no point in mentioning all the facts, we have to filter out those that are relevant. *Relevance* here means to be sensitive to the writer's and reader's goals, their knowledge, preferences, and point of view. And, of course, there are still many other factors that have to be taken into account: for example external constraints like space (number of pages), time (deadline), mode (spoken vs. written), etc. Moreover, the knowledge to be expressed (message) may be in a form that is fairly remote from its (p. 290) corresponding linguistic expression (surface form). For example, the information may consist of raw, numerical data or qualitative representations. In both cases, data has first to be *interpreted* in order to specify its semantic value: e.g. a sequence of decreasing numeric values might be semanticized as *a drop, a decrease*, or *the falling of temperature*. Finally, although knowledge bases are generally structured, it is less often the case that this structure can be directly used for organizing a text. Data can be organized alphabetically or chronologically, but most types of texts exhibit other structures—their information is organized *topically* (e.g. for our survey paper task: deep generation, surface generation, lexical choice, etc.; or for the weather report: the state of the weather today, predictions about changes, statistics, etc.). This requires further information, either explicitly—via, for example, an *ontology* stating the components of an object or a task (Chapter 25) and some account of appropriate text structures—or implicitly, that is, via inference rules (which in that case are needed in addition to the knowledge base).

An important function of this organizational knowledge is to enable the text producer to select and structure information according to criteria not explicitly mentioned, for example, in our survey, according to *pioneering work, landmark systems*, etc. Such criteria then need to be operationalized before they can playa role in an NLG system: for example, one could define the notion of 'landmark' as work that has changed the framework within which a task is being carried out (e.g. the move from *schemata* to *RST* in discourse planning as discussed below), or as a paradigm shift that has since been adopted by the field, etc. This close link between *content selection* and *discourse structure* is a natural one: information is generally only relevant for a text given some particular communicative goals. For this reason, these two tasks are often addressed by the same component.

15.2.2 Text planning: the schema-based approach and the TEXT system



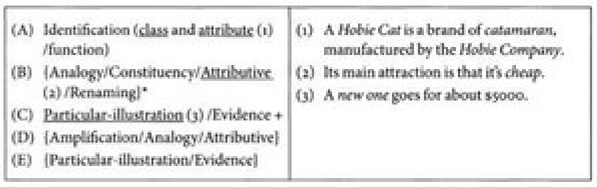
Name of predicate	Function	Example
Identification	identifies the object as a member of a class	*A catamaran is a kind of sailboat.*
Constituency	presents constituents of the entity	*A sailboat has sails and a mast.*
Illustration	provides an example	*The Titanic is a boat.*

Click to view larger

Fig.15.3 **Examples of rhetorical predicates**

TEXT (McKeown 1985) was one of the first systems to automatically produce paragraph- length discourse. Having analysed a large number of texts and transcripts, McKeown noticed that, given some *communicative goal*, people tended to present the *same kind of information* in the *same order*. To capture these regularities, McKeown classified this information into **rhetorical predicates** and **schemata**. The former describe roughly the semantics of the text's building blocks; examples of predicates, their function, and illustrative clauses are shown in Fig. 15.3. The latter describe the legal combination of predicates to form patterns or text templates. Fig 15.4(a) gives several possible decompositions of the *Identification* schema, using the notation for schema definition McKeown proposes. Fig 15.4(b) then gives an example text; the connection between the sentences of the text and the responsible text schema **(p. 291)** elements are indicated by the sentence numbering. Schemata are thus both *organizational devices* (determining *what* to say *when)* and *rhetorical means*, i.e.discourse strategies for achieving some associated goals.

Once the *building blocks* and their *combinations* are known, simple text generation becomes fairly straightforward. Given a goal (define, describe, or compare), the system chooses a schema which stipulates in abstract terms *what* is to be said *when*. Whenever there are several options for continuing—for example, in the second sentence one could have used *analogy, constituency*, or *renaming* instead of the predicate *attributive* (cf. Fig 15.4(a), schema B)—the system uses **focus rules** to pick the one that ties in best with the text produced so far. Coherence or text structure is thus achieved as a side effect of choosing a goal and filling its associated schema.



(A) Identification (class and attribute (1) /function)	(1) A *Hobie Cat* is a brand of *catamaran*, manufactured by the *Hobie Company*.
(B) {Analogy/Constituency/Attributive (2) /Renaming}*	(2) Its main attraction is that it's *cheap*.
(C) Particular-illustration (3) /Evidence +	(3) A *new one* goes for about $5000.
(D) {Amplification/Analogy/Attributive}	
(E) {Particular-illustration/Evidence}	

Click to view larger

Fig.15.4(a) **Identification schema**

Fig.15.4(b) **Corresponding text**

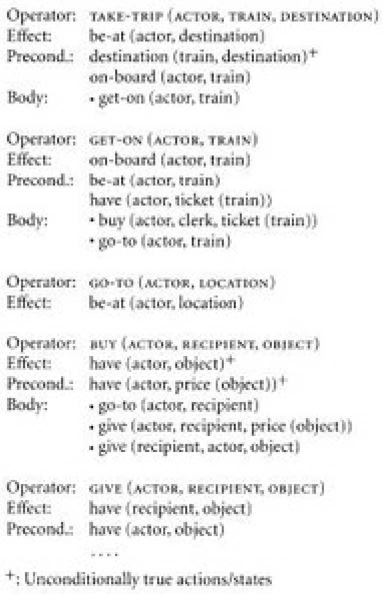
Key:{} optionality;/alternative; * optional item which may appear 0 to n times; + item may appear 1 to n times. The underlined predicates are the ones that are actually used in the text produced *(see 15.4b)*

Not all information stipulated by the schema must appear in a final text; actually only schema parts (A) and (C) are compulsory. A given predicate may also be expanded and repeated. Optionality, repetition, and recursion (not illustrated in our example) make schemata very powerful, but they still have some recognized shortcomings—the most significant of which is the lack of connection between the components of an adopted schema and the goals of a text. The goals associated with a schema specify the role of the schema as a whole—Leo why to use it—but they do (p. 292) not specify the roles of its parts: why use a specific rhetorical predicate at a given moment? Hence, in case of failure (for example, the information given being considered as insufficient by the user), the system cannot recover in order to offer an alternative solution because there is no linking between schema-parts and subgoals for the text as a whole: the TEXT system will invariably produce the same answer, regardless of the user's expertise, informational needs, or problems.

15.2.3 Text planning and rhetorical structure theory

Rhetorical structure theory (RST) was adopted in NLG partly to overcome the problems of schemata and to provide a more flexible mechanism that links communicative goals and text structure more closely. According to RST (cf. Mann and Thompson 1987), any coherent text is decomposable into a recursive structure of text ‘spans’ (usually clauses): the relations between text spans construct, often implicit, propositions involving common rhetorically relevant predicates such as *cause, purpose, motivation*, and *enablement*; these relations generally divide a text span into at least two segments: one that is of most importance for the text—the **nucleus**—and others which have more a supportive role—the **satellites**. The existence of a single overarching rhetorical structure for a text then accounts for its perceived coherence. RST received its initial computational operationalizations in Hovy 1988 and Moore and Paris 1988 and has since been incorporated in a wide range of NLG systems. Probably the most widespread version of operationalized RST is that presented in Moore and Paris 1993.

RST in this form makes essential use of the *planning paradigm* from AI (Sacerdoti 1977). **Planning** here means basically *organizing actions rationally* in order to *achieve a goal*. Since most goals (problems) are complex, they have to be decomposed. High-level, global goals are thus refined to a point where the actions associated with them are primitive enough to be performed directly. Planning supposes three things: a *goal* (problem to be solved), a *plan library* (i.e. a set of plan operators each of which allows the achievement of a specified goal), and a *planning method* (algorithm). Each *plan operator* combines the following information: an *effect* (i.e. a state which holds true after the plan's execution, i.e. the goal); zero, one, or more *preconditions* (conditions which must be satisfied before the plan can be executed); and a *body* (i.e. a set of actions, which are the means for achieving the goal). Somewhat simplified, then, a hierarchical planner works by decomposing the top-level goal using plan operators whose effects match the goals given and whose preconditions are true. These plan operators may introduce further subgoals recursively.



```
Operator:  TAKE-TRIP (ACTOR, TRAIN, DESTINATION)
Effect:    be-at (actor, destination)
Precond.:  destination (train, destination)+
           on-board (actor, train)
Body:      • get-on (actor, train)

Operator:  GET-ON (ACTOR, TRAIN)
Effect:    on-board (actor, train)
Precond.:  be-at (actor, train)
           have (actor, ticket (train))
Body:      • buy (actor, clerk, ticket (train))
           • go-to (actor, train)

Operator:  GO-TO (ACTOR, LOCATION)
Effect:    be-at (actor, location)

Operator:  BUY (ACTOR, RECIPIENT, OBJECT)
Effect:    have (actor, object)+
Precond.:  have (actor, price (object))+
Body:      • go-to (actor, recipient)
           • give (actor, recipient, price (object))
           • give (recipient, actor, object)

Operator:  GIVE (ACTOR, RECIPIENT, OBJECT)
Effect:    have (recipient, object)
Precond.:  have (actor, object)
           ....
+: Unconditionally true actions/states
```

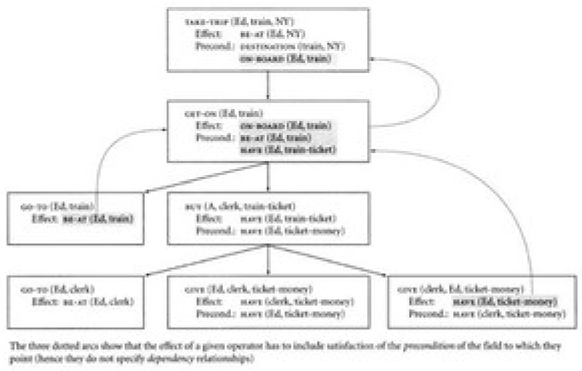
Click to view larger

Fig.15.5 **Plan library of action types**

Let us take an example. Suppose'Ed'wanted to go to 'NewYork'. Weassume that this can be represented in terms of a goal such as: [BE-AT (ACTOR, DESTINATION)], with the arguments instantiated as required for our particular case. Now, rather than building (p. 293) a plan that holds only for one specific problem, one generally appeals to a generic plan—e.g. a set of plans and solutions for a related body of problems. In Fig. 15.5, for example, we show a library of plan operators appropriate for the planning of trips for different people to go to different places. Given our present starting goal of wanting to get Ed to New York, the planner looks for operators to achieve this, i.e. operators whose effect fields match the goal. In our case the top-level goal can partially be achieved in a number of ways: for example via the TAKE-TRIP or the GET-ON operator; however, there are additional dependency relationships and states of affairs that need to be fulfilled and these restrict the ordering of applicability of the plan operators. Thus, we cannot take a trip unless we get on a train, etc. (cf. Fig. 15.5). Assuming that the text planner has got as far as proposing the taking of a trip, then the body of the operator is posted as a further subgoal: GET-ON. Associated with this further operator are two conditions, [BE-AT(ACTOR, TRAIN)] and [HAVE(ACTOR, TICKET)], which can be satisfied respectively via the GO-TO and BUY operators. The first one is unconditional, it can be directly achieved, while the second one decomposes into three actions: *go to the clerk, give him the money*, and *receive the ticket*. It is this last action that ensures that the traveller (actor) has finally what is needed to take the trip: the ticket (precondition of the GET-ON operator). The process terminates when all goals (regardless of their level) have been fulfilled, which means that all effects or preconditions hold true, either unconditionally—the action being primitive enough (p. 294) to dispense with any further action (see the GO-TO operator)—or because there is an operator allowing their realization. Fig. 15.6 shows the partially completed hierarchical plan for achieving the final goal.

Within operationalized RST, RST relations are modelled as plan operators and the effects are communicative goals. To illustrate this use of RST in text planning, we take a slightly adapted example from Vander Linden (2000). The application scenario is providing help or documentation for a computer program. Knowledge concerning user interactions with the program has been modelled, and help-information is to be automatically generated. Several NLG systems have been built for this kind of scenario. Suppose that a user would like to know how to save a file. For the NLG system to generate an appropriate text, it is first given a top-level communicative goal; in this case:

(COMPETENT H (DO-ACTION SAVING-A-FILE))

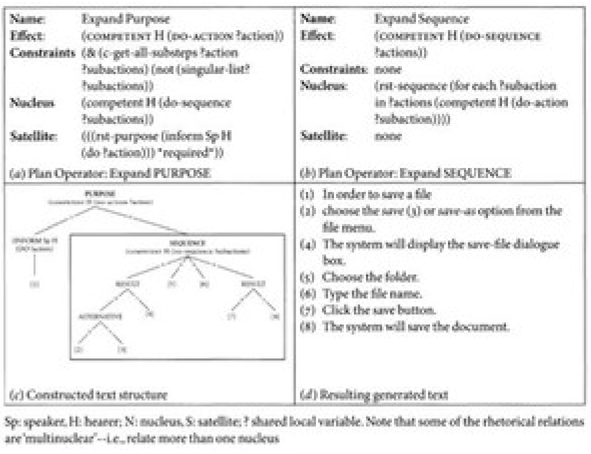

TAKE-TRIP (Ed, train, NY)
Effect: BE-AT (Ed, NY)
Precond.: DESTINATION (train, NY)
ON-BOARD (Ed, train)
GET-ON (Ed, train)
Effect: ON-BOARD (Ed, train)
Precond.: BE-AT (Ed, train)
HAVE (Ed, train-ticket)
GO-TO (Ed, train)
Effect: BE-AT (Ed, train)
BUY (A, clerk, train-ticket)
Effect: HAVE (Ed, train-ticket)
Precond.: HAVE (Ed, ticket-money)
GO-TO (Ed, clerk)
Effect: BE-AT (Ed, clerk)
GIVE (Ed, clerk, ticket-money)
Effect: HAVE (clerk, ticket-money)
Precond.: HAVE (Ed, ticket-money)
GIVE (clerk, Ed, ticket-money)
Effect: HAVE (Ed, ticket-money)
Precond.: HAVE (clerk, ticket-money)
The three dotted arcs show that the effect of a given operator has to include satisfaction of the precondition of the field to which they point (hence they do not specify dependency relationships)

Click to view larger

Fig.15.6 **A partial plan built to achieve the goal 'get to NY'**

That is: achieve a state in which the hearer (H) is competent to do the action 'saving-a- file'. An RST-based hierarchical text planner then has to access information in the knowledge base and structure this into an appropriate text plan. Domain knowledge (p. 295) about the program to be documented is generally stored as a collection of goals and methods for achieving them. Thus, there will be a high-level goal 'save a file' and a sequence of actions corresponding to a method for it.

The simplest actions a hearer (H) is able to perform *(primitive planning actions)* are those expressed by individual messages associated with simple speech acts, such as 'inform', 'question', etc. When all goals have been expanded to plan steps involving primitive (and therefore directly achievable) subgoals, the planning process halts, as the text structure is then complete.



Name:	Expand Purpose
Effect:	(COMPETENT H (DO-ACTION ?action))
Constraints	(& (c-get-all-substeps ?action ?subactions) (not (singular-list? ?subactions))
Nucleus	(competent H (do-sequence ?subactions))
Satellite:	(((rst-purpose (inform Sp H (do ?action))) *required*))

(a) Plan Operator: Expand PURPOSE

Name:	Expand Sequence
Effect:	(COMPETENT H (DO-SEQUENCE ?actions))
Constraints:	none
Nucleus:	(rst-sequence (for each ?subaction in ?actions (competent H (do-action ?subaction))))
Satellite:	none

(b) Plan Operator: Expand SEQUENCE

(c) Constructed text structure

(1) In order to save a file
(2) choose the save (3) or save-as option from the file menu.
(4) The system will display the save-file dialogue box.
(5) Choose the folder.
(6) Type the file name.
(7) Click the save button.
(8) The system will save the document.

(d) Resulting generated text

Sp: speaker, H: hearer; N: nucleus, S: satellite; ? shared local variable. Note that some of the rhetorical relations are 'multinuclear'—i.e., relate more than one nucleus

Click to view larger

Fig.15.7 **Illustration of 2 plan operators, the resulting text plan (tree) and corresponding text**

One plan operator that matches our example top-level goal is that given in Fig. 15.7(a): 'expand purpose'. In order to see if this operator may be applied, the preconditions/ constraints must first be checked. To do this, the variable ?ACTION of the effect field is bound to the corresponding action in our top-level goal 'saving a file'; then the constraints slot specifies that this action must be decomposable into some substeps and that this sublist should not consist of a single action. If we assume that (p. 296) this is the case for our present example—

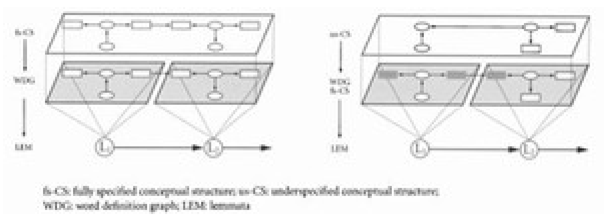
saving a file will typically require several substeps—then the operator will apply. Note that this process not only checks constraints but also retrieves relevant information (via matching and binding), thereby combining content selection with its ordering. The subgoals specified in the nucleus and satellite slots are then posted and must both in turn be satisfied for planning as a whole to succeed. The satellite subgoal succeeds immediately, since it calls for the system, or speaker (Sp), to inform (a primitive act) the hearer (H) about some propositional content (‘DO “save a file”’). Moreover, this entire content is embedded as a satellite under an RST‘purpose’ relation, which then constrains its possible linguistic realizations; one possible realization of this branch of the text plan, provided by the surface generator on demand, is given in (1)in Fig. 15.7(d). The goal posted under the nucleus requires further expansion, however, and so the planning process continues (cf., e.g. Fig. 15.7(b): ‘expand sequence’). The result of the process is a tree whose nodes represent the rhetorical relations (effect), and whose leaves represent the verbal material allowing their achievement. This result is then handed either to microplanning (e.g. for aggregation, which may eliminate redundancies) or to the surface generator directly. Fig. 15.7(C, d) show a final text plan constructed by the planning operators and a possible corresponding text.

15.3 Microplanning and Realization

We sketched above (section 15.1) some of the subtasks of microplanning; we will address here only one of these: *lexicalization*.

15.3.1 Lexicalization

Somewhere in a generation system, there must be information about the words that would allow us to express what we want to convey, their meanings, their syntactic constraints, etc. This kind of information is stored in a **lexicon**, whose precise organization can vary considerably (see, for examples and discussion, Chapters 3 and 25). Dictionaries are static knowledge, yet this latter has to be used: words have to be accessed and, in case of alternatives (synonyms), some selection has to be made. All this contributes to what is called *lexicalization*.

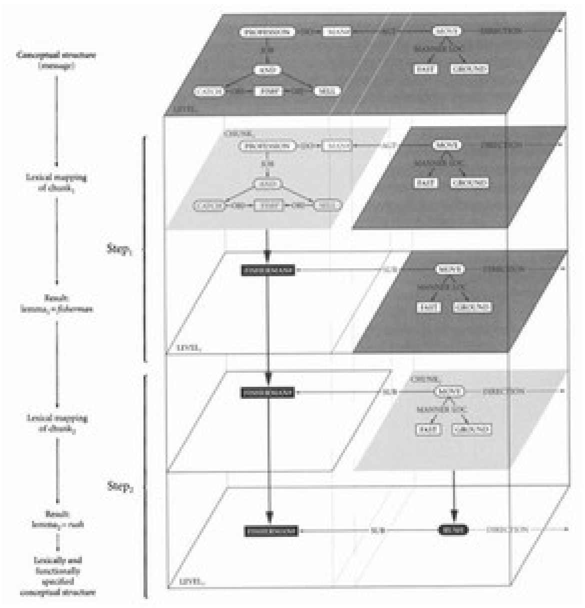

fs-CS
WDG
LEM
us-CS
WDG fs-CS
LEM
fs-CS: fully specified conceptual structure; us-CS: underspecified conceptual structure;
WDG: word definition graph; LEM: lemmata

Click to view larger

Fig.15.8(a) **The lexicon as a resource for compacting a given conceptual chunk**

Fig.15.8(b) **The lexicon as a resource for refining an underspecified message**

There are two broad views on lexicalization: one can conceive it either as conceptually driven (meaning) or as lexicon driven (see Figure 15.8*a* and 15.8*b*). In the former view everything is given with the input (i.e. the message is complete), and the relationship (p. 297) between the conceptual structure and the corresponding linguistic form is mediated via the lexicon: lexical items are selected provided that their underlying content covers parts of the conceptual input. The goal is to find sufficient mutually compatible lexical items so as to completely cover the input with minimal unwanted additional information. In the latter view, the message to be expressed is incomplete prior to lexicalization, the lexicon serving, among other things, to refine the initially underspecified message (for arguments in favour of this second view see Zock 1996). One begins with a skeleton plan (rough outline or gist) and fleshes it out progressively depending on communicative and speaker needs (for example, by providing further information necessary for establishing reference).


Conceptual structure (message)
Lexical mapping of chunk₁
Result: lemma₁ = fisherman
Lexical mapping of chunk₂
Result: lemma₂ = rush
Lexically and functionally specified conceptual structure
Step₁
Step₂
PROFESSION
MAN
ACT
MOVE
DIRECTION
MANNER LOC
FAST
GROUND
SUB
AND
CATCH
FISH
SELL
CHUNK₁
CHUNK₂
FISHERMAN
RUSH
LEVEL₁
LEVEL₂
LEVEL₃

Click to view larger

Fig.15.9 **Progressive lexical specification**

To illustrate the first view (conceptually driven lexicalization) we will use an adapted example taken from Zock 1996; suppose your goal were to find the most precise and economic way for expressing some content, let us say 'the fisherman rushed to the canoe'; the problem is to decide how to carve out the conceptual space so as to cover everything planned. We will show here only how the first two open-class words *(fisherman* and *rush)* are selected, suggesting by the direction link that more is to come. Lexicalization is performed in two steps. During the first only those words are selected that pertain to a given semantic field (for example, movement verbs). In the next step the lexicalizer selects from this pool the term that best expresses the intended meaning, i.e. the most specific term (maximal coverage). Thus we need to map the conceptual chunks (input) 'a man whose profession consists in catching and selling fish' and 'move fast on the ground' into their linguistic counterparts: *fisherman* and *rush*. The process is shown in Fig. 15.9; for more details see Nogier and Zock (1992) and Zock 1996; for a sophisticated proposal of what to do if one cannot find a complete covering, see Nicolov 1999. There are many open questions concerning how such a process may be controlled effectively when faced with more alternatives. (p. 298)

15.3.2 Surface generation

The final stage of NLG consists in passing the results of the microplanner to the **surface generator** to produce strings of properly inflected words; while some surface generators produce simple text strings, others produce strings marked up with tags (p. 299) that can be interpreted for prosody control for spoken language, for punctuation, or for layout. Unlike in situations of theoretical grammar development, the primary task of a surface generator is certainly not to find all possible ways to express a given propositional content, but rather to find a way of expressing the given communicative content that is in line with the rules of the language, the speakers' goals, and, often most crucially, the rest of the text that has been generated so far and is to come (the co-*text*).

Surface generation is nevertheless the area of NLG that overlaps most with the traditional concerns of linguistics—particularly structural linguistics concerned with accounts of sentences (syntax: Chapter 4; semantics: Chapter 5). However, although nearly all linguistic theories have been implemented in trial generation and analysis systems, there are essentially three theoretical frameworks that have emerged as being particularly well suited to the needs of NLG. These are accounts based on systemic-functional grammars (SFG: Halliday 1994), tree-adjoining grammars (TAG: Danlos 2000), and Mel'čuk's meaning-text model (MTM: Mel'čuk 1988).

SFG has been adopted by those interested in the systematic representation, *functional motivation*, and consequences of the choice of particular linguistic resources—e.g. what impact do the selections of particular grammatical patterns (and hence indirectly of particular grammatical Subjects, Themes, etc.) have on the coherence and cohesion of a text? Systemic-functional grammars can also be viewed as 'feature-based' systems, since feature selection is their main operation for describing and choosing a linguistic unit. As a result, some systemically based generators are directly implemented using feature matching, or *unification*, as their main operation (e.g. FUF/SURGE: Elhadad and Robin 1997), others adopt special-purpose implementations—typically for the more efficient generation behaviour necessary when dealing with large-scale grammars (e.g. PENMAN: Mann and Matthiessen 1986; KPML: Bateman 1997).

TAG has typically been adopted by those who are seeking **incremental generation**—i.e. a generation architecture where information can be consumed by the available grammatical constructions of a language as soon as that information becomes available. To achieve this, TAG splace their main descriptive work on setting out extensive libraries of partial syntactic structures, which nowadays are increasingly indexed according to their so-called *lexical anchor—that* is, particular families of syntactic structures are linked with particular lexical items (cf. Chapter 26 and Harbusch, Finkler, and Schauder 1991)—and are organized hierarchically.

MTM has been adopted principally by researchers interested in issues of lexical variation (cf. Iordanskaja, Kittredge, and Polguère 1991) *and restricted lexical co-occurrence* (**collocations**). The latter are units where the form for expressing one part of the meaning (the *collocate)* depends on the other (the *base)*, the one to which this meaning is applied. For example, suppose you wanted to express the idea that some entity has the property [SIZE-big]: surface generation here is complicated by the fact that lexicalization is not a local choice—depending on the lexeme selected (p. 300) for the entity being described, i.e.depending on the base (e.g. *man* vs. *mountain)*, we may need to select different *collocates* (e.g. *tall* vs. *high* respectively). This is because it is not appropriate (in English) to say 'he is a high man' or 'this is a tall mountain'. Such collocational restrictions are widespread in all languages and are difficult, if not impossible, to 'predict' reliably on semantic grounds.

15.4 Open Issues and Problems

Among the different open issues and problems, we will mention here only three: architectural, inferential and control.

(i) *Architectural*. Architectural decisions revolve around the problems of *order* (uni- vs. bidirectional); *completion* (one pass vs. several, revisions), *control* (central vs. distributed), etc. Many systems designers have—purely on engineering grounds (e.g. ease of implementation, speed of generation)—opted for a unidirectional flow of information between the modules (the **pipelined** view). This poses a number of problems; most prominently, there is a danger of not being able to find resources (e.g. words) to express the message planned. Hence, one reaches a deadlock; this is often discussed as one of the problems of the '**generation gap**' (Meteer 1992) between different levels of representation in NLG architectures. For a thorough discussion of architectures in general, see de Smedt, Horacek, and Zock 1996 and for attempts at architecture standardization take a look at Cahill et al. (1999) and the website mentioned below.
(ii) *Inferences*. Next to schemata and RST, **inference-driven generation** has been proposed to deal with the problem of content determination; for some pioneering work in this area, each very different from the others, see Cawsey 1991, Kölln (1997), Stone and Doran 1997. This is a particularly promising area; at the same time, it is known that one of the hardest problems of inferencing is to have good criteria for making the process stop.

(iii) *Control*. As the number of possibilities to express a given content grows, it becomes increasingly necessary to find good criteria for choosing among the alternatives. While it is relatively easy to find criteria and tests for deciding on well-formedness, it is a lot harder to settle the issue of adequacy. When should one use a grammatical resource, such as passive voice, a relative clause, etc.? What to do in case of conflict? When to stop? The problem of termination is a delicate issue in general: when is a text optimal? (ever?) And finally, what holds for the text in general, holds also for each of its components (what is a perfect input, the best organization, etc.). There is, therefore, still much to do both in the fundamental design of NLG systems and theory and in its practical application.

(p. 301) Further Reading and Relevant Resources

There are now several excellent introductory papers to NLG that take a more detailed view of both the tasks of NLG and the approaches that have attempted to deal with them; see, for example: Bateman (1998), Hovy 1996, Vander Linden (2000), and the many references cited there. In addition, for further pointers, descriptions of systems, etc., see Zock and Adorni 1996 and Reiter and Dale 2000. The latter include a particularly detailed comparison of the SFG, FUF, and MTM-syntactic approaches to surface generation. For more on grammars, see Teich 1998, while both Paiva 1998 and Gal et al. (1991) are good NLG sources for the practically minded. Reviews of the state of the art in approaches to text planning using rhetorical relations are given by Maybury 1990, Hovy 1993, and Paris 1993. They all make a very readable introduction to text planning and the latter shows well its control by *user-modelling*. For work on aggregation see Dalianis 1999. Two excellent survey articles covering work on lexical choice are Wanner 1996 and Stede 1999. In order to get a feel of the methods psychologists use at both the sentence and discourse level, Levelt 1989, Bock 1996, and Andriessen, de Smedt, and Zock 1996 can all be used as starting points.

For access to recent publications, reference lists, free software, or to know *who is who*, the best thing to start out with is the website of the ACLs Special Interest Group for Generation (SIGGEN); this can be accessed via the ACL pages at http://www.cs.columbia.edu/~acl/. There are also a number of freely available generation systems and generation grammars that can be used for hands-on exposure to the issues of NLG. The three most extensive are Elhadad's FUF/Surge system at http://www.cs.bgu.ac.il/fuf/index.htm, which includes a very wide-coverage generation grammar for English; the MTM-influenced syntactic generator RealPro at http://www.cogentex.com/technology/realpro/index.shtml; and the KPML multilingual generation system at http://purl.org/net/kpml, which includes an extensive grammar development environment for large-scale systemic-functional grammar work and teaching, introductory tutorials, as well as a growing range of generation grammars: e.g. grammars for English (very large), German, Dutch, Czech, Russian, Bulgarian, Spanish, French, and others. There are also several quite extensive on-line bibliographies including substantial NLG-related references—for example, several of the bibliographies under Alf-Christian Achilles's Collection of Computer Science Bibliographies at the University of Karlsruhe (http://liinwww.ira.uka.de/bibliography/index.html, subsection Artificial Intelligence) are worth bookmarking—particularly http://liinwww.ira.uka.de/bibliography/Ai/nlg.html (Kantrowitz) and http://liinwww.ira.uka.de/bibliography/Ai/bateman.html (Bateman). In addition, the following sites contain a variety of useful information about diverse aspects of NLG: the Reference Architectures for Generation (RAGS)project website describes an attempt to provide a general scheme for describing architectures for 'practical NLG' (www.itri.brighton.ac.uk/projects/rags); the Scottish and German NLG sites (www.cogscLed.ac.uk/~jo/gen/snlg/nlg.html (p. 302) and http://www.ling.uni-potsdam.de/~stede/nlg-ger.htm respectively) have a variety of useful pointers to systems, literature, and researchers in their respective countries; and the RST site offers a wide range of information, both computational and linguistic, concerning all aspects of rhetorical structure theory and its application (http://www.sil.org/linguistics/RST). A range of papers are available on-line at http://www.iro.umontreal.ca/~scriptum/PublicationsMembres.html, and for pointers on AI in general http://www.norvig.com/ is very useful. Finally, we also maintain an almost complete list of NLG systems with linked bibliographical data and pointers to further web-based information at http://purl.org/net/nlg-list-to which additions are welcome at any time!

References

Adorni, G. and M. Zock (eds.). 1996. *Trends in Natural Language Generation: An Artificial Intelligence Perspective*. New York: Springer Verlag.

Andriessen, J., K. de Smedt, and M. Zock. 1996. 'Discourse planning: empirical research and computer models.' In T. Dijkstra and K. de Smedt (eds), *Computational Psycholinguistics: AI and Connectionist Models of Human Language Processing*, London: Taylor and Francis.

Bateman, J. 1997. 'Enabling technology for multilingual natural language generation: the KPML development environment'. *Journal of Natural Language Engineering*, 3(1), 15–55.

—— 1998. 'utomated discourse generation'. In A. Kent (ed.), *Encyclopedia of Library and Information Science*, New York: Marcel Dekker, Inc.

Bock, K. 1996. 'Language production: methods and methodologies'. *Psychonomic Bulletin and Review*, 3, 395–421.

Cahill, L., C. Doran, R. Evans, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper. 1999. *Towards a reference architecture for natural language generation systems: the RAGS project*. Technical Report: ITRI-99–14, Brighton.

Cawsey, A. 1991. 'Using plausible inference rules in description planning'. *Proceedings of the 5th Conference of the European Chapter of the ACL* (Berlin), 119–24.

Cole, R., J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, A. Zampolli, and V. Zue. 1997. *Survey of the State of the Art in Human Language Technology*. Cambridge: Cambridge University Press.

Dalianis, H. 1999. ‘Aggregation in natural language generation.’ *Journal of Computational Intelligence*, 15(4).

Danlos, L. 2000, ‘G-TAG: a lexicalized formalism for text generation inspired from tree adjoining grammar: TAG issues’. In A. Abeille and o.Rambow (eds.), *Tree-Adjoining Grammars*, Stanford, Calif: CSLI, 343–70.

de Smedt, K., H. Horacek, and M. Zock. 1996. ‘Some problems with current architectures in Natural Language Generation’. In: Adorni and Zock (1996), 17–46.

Elhadad, M. and J. Robin. 1997. *SURGE: a comprehensive plug-in syntactic realization component for text generation*. Technical Report, Computer Science Dept., Ben-Gurion University, Beer-Shiva, Israel.

Gal, A., G. Lapalme, P. Saint Dizier, and H. Somers. 1991. *PROLOG for Natural Language Processing*. Chichester: J. Wiley and Sons.

(p. 303) Halliday, M. A. K. 1994. *An Introduction to Functional Grammar*, London: Arnold.

Harbusch, K., W Finkler, and A. Schauder. 1991. ‘Incremental syntax generation with tree adjoining grammars’. In W. Brauer and D. Hernandez (eds.), *Verteilte Kuenstliche Intelligenz und kooperatives Arbeiten*, Berlin: Springer, 363–74.

Hovy, E. 1988. ‘Planning coherent multisentential text’. *Proceedings of the 26 th Annual Meeting of the ACL (ACL* '88) (Buffalo), 179–86.

—— 1993. ‘Automated discourse generation using discourse structure relations’. *Artificial Intelligence*, 63, 341–85.

—— 1996. ‘Language generation’. In Cole et al. (1997), 159–67.

Iordanskaja, L., R. Kittredge, and A. Polguère, 1991. ‘Lexical selection and paraphrase in a meaning-text generation model’. In Paris et al. (1991), 293–312.

Kölln, M. 1997. ‘Textproduktion als intentionaler, benutzerorientierter Prozess’ St. Augustin: Infix: DISKI, 158.

Levelt, W. 1989. *Speaking: From Intention to Articulation*. Cambridge, Mass.: MIT Press.

McDonald, D. 2000. ‘Natural language generation’. In R. Dale, H. Moisl, and H. Somers (eds.), *A Handbook of Natural Language Processing Techniques*. New York: M. Dekker Inc., 147–79.

McKeown, K. 1985. ‘Discourse strategies for generating natural-language text’. *Artificial Intelligence*, 27, 1–41.

Mann, W. and C. Matthiessen. 1986. ‘Demonstration of the Nigel text generation computer program’. In J. Benson and W Greaves (eds.), *Systemic Functional Approaches to Discourse*, Norwood, NJ: Ablex Publ. Company.

—— and C. Matthiessen and S. A. Thompson. 1987. ‘Rhetorical structure theory: description and construction of text structures’. In G. Kempen (ed.), *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics*. Dordrecht: Kluwer Academic Publishers, 85–96.

Maybury, M. 1990. *Planning multisentential English text using communicative acts*, Ph. D. thesis, University of Cambridge.

Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice*. New York: State University of New York Press.

Meteer, M. 1992. *Expressibility and the Problem of Efficient Text Planning*. London: Pinter.

Moore, J. D. and C. L. Paris. 1988. ‘Constructing coherent texts using rhetorical relations’. *Proceedings of the 10th Annual Conference of the Cognitive Science Society (COGSCI-88)*, 199–204.

—— 1993. ‘Planning text for advisory dialogues: capturing intentional and rhetorical information’. *Computational Linguistics*, 19(4).

Nogier, J. E, and M. Zock. 1992. ‘Lexical choice by pattern matching’. *Knowledge Based Systems*, 5(3), 200–12.

Nicolov, N. 1999. ‘Approximate text generation from non-hierarchical representation in a declarative framework’. Ph.D. thesis, University of Edinburgh.

Paiva, D. 1998. *A survey of applied natural language generation systems*. Technical Report ITRI98-03, Brighton. **http://www.itri.brighton.ac.uk/techreports**.

Paris, C. L. 1993. *User Modelling in Text Generation*. London: Frances Pinter.

—— W. Swartout, and W.Mann (eds.). 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Boston: Kluwer Academic Publishers.

Reiter, E. and R. Dale. 2000. *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press.

Sacerdoti, E. 1977. *A Structure for Plans and Behavior*. Amsterdam: North Holland.

(p. 304) Stede, M. 1999. *Lexical Semantics and Knowledge Representation in Multilingual Text Generation*. Dordrecht: Kluwer.

Stone, M. and C. Doran. 1997. 'Sentence planning as description using tree adjoining grammar'. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL* '97) (Madrid), 198–205.

Teich, E. 1998. *Systemic Functional Grammar in Natural Language Generation: Linguistic Description and Computational Representation*. London: Cassell Academic Publishers.

Vander Linden, K. 2000. 'Natural language generation'. In D. Iurafsky and J. Martin (eds.), *Speech and Language Processing: An Introduction to Speech Recognition, Computational Linguistics and Natural-Language Processing*. New Jersey: Prentice-HalL.

Wanner, M. 1996. 'Lexical choice in text generation and machine translation'. *Machine Translation*, 11(1–3), 3–35.

Zock, M. 1996. 'The power ofwords in message planning'. *Proceedings of the 16th International Conference on Computational Linguistics* (Copenhagen), 990–5.

—— and G. Adorni. 1996. 'Introduction' to Adorni and Zock (1996), 1–16.

Notes:

(1) Authorsin alphabetical order.

John Bateman

John Bateman is Professor of Applied Linguistics at the University of Bremen, Germany. Since obtaining his Ph.D. from the Department of Artificial Intelligence at the University of Edinburgh in 1985, he has worked in natural language generation and functional linguistics, applying the latter to the former, on projects in Scotland, Japan, California, and Germany, as well as in a variety of European cooperations. His main research focuses are multilingual NLG, multimodal document design, discourse structure, and the application of all areas of systemic-functional linguistics.

Michael Zock

Michael Zock is Emeritus Research Director of the CNRS and is currently affiliated with University of Marseille, France (LIF-AMU). He holds a Ph.D. in experimental psychology. Having initiated (1987, Royaumont) the European Workshop Natural Language Generation workshop series, he has edited five books on language generation. His major research interests lie in the building of tools to help humans to produce language (speaking/writing) in their mother tongue or when learning a

foreign language. His recent work is devoted to the navigation in electronic dictionaries, aiming at overcoming the tip-of-the-tongue problem, and facilitating the access, memorization, and automation of words and syntactic structures.