Introduction

The use of natural language as a means of communication between man and machine requires that the computer be able to (a) understand the user's message, (b) know how to reply to this message and (c) formulate a reply in the language of the user. The first task involves automatic analysis, the second a reasoning module, and the third automatic generation (see Fig. 1). However, automatic analysis and generation are not limited only to dialogue. Any system that produces a formal representation from a text is in the domain of automatic analysis, whereas any system that produces a text from a formal representation is in the domain of automatic generation.

Historically, research on automatic analysis preceded research on automatic generation by many years. Many programs for analysis have been written since the 1950s whereas the development of automatic generation is less than ten years old. In actual practice, computers have been producing messages in natural language for a long time. Consider, for example, error messages like *There is no file FOO–BAR*, following a command to print the file FOO–BAR. Such messages are pre-recorded in the machine in the form of sentences containing variables (e.g. *There is no file* F_x). When the

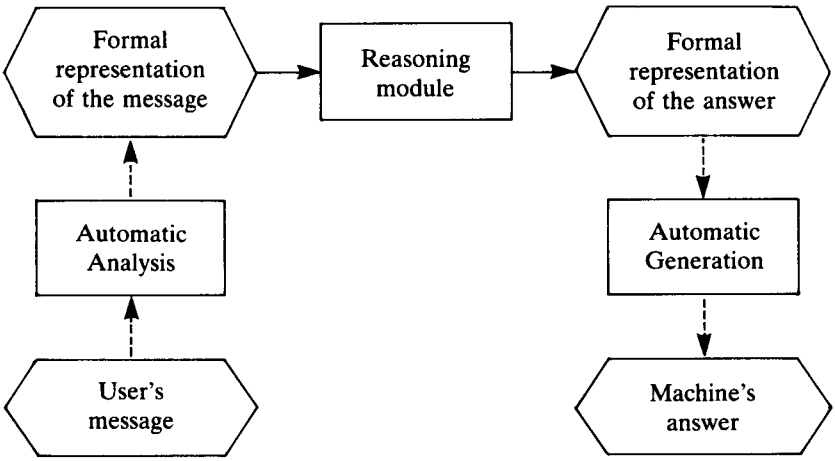

Fig. 1 Man-machine dialogue

messages are to be printed, the variables are given a value and the completed sentences are displayed on the screen; none of this involves any linguistic computation. The simplicity of the method of pre-recorded sentences may have given rise to the impression that generating messages in natural language is a simple matter. However, this method is not very practical when more elaborate messages are required. Suppose that we want to create a file management system that produces error messages such as the following, after a command to print the file FOO-BAR:

(1) *There is no file FOO–BAR, but there is a file FOO–BAT.*
Do you want to print FOO–BAT? (answer Y/N)
(2) *There is a file FOO–BAR.LCL and a file FOO–BAR.UCL.*
Do you want to print FOO–BAR.LCL? (answer Y/N)
Do you want to print FOO–BAR.UCL? (answer Y/N)
(3) *The file FOO–BAR is not in your current directory, but it is in your directory ⟨Danlos.R⟩.*
Do you still want to print it? (answer Y/N)

Such messages presuppose a search algorithm for finding files that might correspond to the one the user wants printed. Producing such messages requires first determining what information is to be conveyed to the user. Messages (1)–(3) do not indicate only those files that might correspond to FOO–BAR. They begin with an explanation of the error and end with one or more questions. Once the information that is to appear in the messages has been determined, one can envisage using a pre-recorded sentence for each item of information and then juxtaposing these pre-recorded sentences without carrying out any linguistic computation. With the following pre-recorded sentences:

There is no file F_x
to explain the error;
The file F_y *is in your directory* D_i
to indicate the existence of a file related to the one to be printed;
Do you want to print F_y*?*
for the question,

we obtain the following messages for the cases that correspond to (1)–(3):

(1′) *There is no file FOO–BAR. The file FOO–BAT is in your current directory.*
Do you want to print FOO–BAT?
(2′) *There is no file FOO–BAR. The file FOO–BAR.LCL is in your current directory. The file FOO–BAR.UCL is in your current directory.*
Do you want to print FOO–BAR.LCL?

Do you want to print FOO–BAR.UCL?

(3′) *There is no file FOO–BAR. The file FOO–BAR is in your directory ⟨Danlos.R⟩.*
Do you want to print FOO–BAR?

These messages are clumsy, and one might wish to improve on them by introducing some kind of semantic computation. For example, the following two sentences could be used to explain the error:

The file F_x is not in your current directory.
There is no file F_x.

The choice between these two forms depends on whether there is a file whose name is identical to the one indicated by the user. This refinement results in the following message for the case (3):

(3″) *The file FOO–BAR is not in your current directory. The file FOO–BAR is in your directory ⟨Danlos.R⟩.*
Do you want to print FOO–BAR?

Although less clumsy than (3′), it is hardly any better. Thus, an approach based on pre-recorded sentences with some semantic computation still results in the production of unsatisfactory messages.

This approach could be improved if the messages underwent syntactic operations such as pronominalization, coordination and complement deletion. However, this would entail annotating the pre-recorded sentences with linguistic information. For example, replacing a noun phrase with a pronoun implies knowledge of the syntactic function of the noun phrase (subject, object, etc.) and of the gender and number of the head noun in order to determine the form of the pronoun (*he*, *she*, *it*, *him*, *her*, etc.). Moreover, performing syntactic operations has none of the simplicity of the method of pre-recorded sentences. The conditions under which such operations can be applied are highly complex, as will be made clear in the course of this book. Finally, even though the quality of the texts obtained can be considerably improved by applying syntactic operations to pre-recorded sentences that have been annotated with linguistic information and whose use is governed by some semantic computation, results of high quality cannot be expected. The formation of a text conveying the items of information $I_1, I_2, \ldots, I_n$ cannot be reduced to a straightforward juxtaposition of sentences $S_1, S_2, \ldots, S_n$ that respectively express $I_1, I_2, \ldots, I_n$. Phenomena such as the existence of a sentence S_i that simultaneously expresses I_k and I_l or the fact that S_j can be inferred from S_m must also be taken into account. In other words, a more global assessment has to be made of the information that is to be conveyed and of the ways of expressing it. The production of well-formed texts thus entails abandoning the method of pre-recorded sentences and adopting

techniques that fall in the domain of automatic generation. The synthesis of error messages such as (1)–(3) will be examined in Chapter 1. This concrete example will allow us to present the main problems encountered in automatic generation and outline a method for resolving them.

The aim of this work is to produce texts from semantic representations of the information to be conveyed. The texts generated are of 'good quality'. From the standpoint of meaning, they are easily understood, unambiguous and give an accurate translation of the information contained in the semantic representations. From the standpoint of form, they employ a varied syntax, a rich vocabulary and a consistent style. Chapter 2 will describe various kinds of research currently under way in automatic generation. In particular, it will examine research that is not oriented towards the generation of 'good quality' texts from semantic representations; for example, research on the determination of the information that is to be conveyed or on the simulation of a speaker. This chapter also includes a discussion of the various generator inputs.

The generation of 'good quality' texts that translate semantic representations requires at least two kinds of decision to be made:

> **conceptual decisions:** in what order should the information appear? What information should be expressed explicitly and what should be left implicit?
>
> **linguistic decisions:** how is the text to be divided up into sentences? What words are to be chosen? What syntactic constructions are to be selected?

It will be shown, first, that each of these decisions is dependent on the others, and second, that all these decisions, except lexical choice, require the setting up of a 'discourse grammar'. The interdependence of these decisions arises principally from the non-isomorphic character of the relationship between meaning and form. This feature of natural language manifests itself in lexical constraints that seem unrelated to elements of meaning. Such constraints are emphasized in the work done on simple sentences at Maurice Gross's Laboratoire d'Automatique Documentaire et Linguistique (LADL). For example, the active–passive transformation cannot be applied without taking into account the lexical items involved; on the one hand, the verb:

> *Bob (abandoned + left) the house.*
> *The house was (abandoned + *left) by Bob*[1].

and on the other, its complements:

> *(Rich politicians + Bob) inhabit(s) Manhattan.*
> *Manhattan is inhabited by (rich politicians + *Bob).*

In an extension of the work carried out at LADL, I have emphasized lexical constraints at the discourse level. For example, a discourse conjoining two sentences, the first one describing an act and the second the result of this act, is acceptable only with some verbs that express the result:

*John shot Mary in the head. He (killed + *murdered) her.*

The existence of such lexical constraints, whether at the level of the simple sentence or at the discourse level, implies that verbal forms cannot be chosen independently of syntactic constructions and/or the order of the information.

As I have already pointed out, well-formed texts cannot be produced by simply juxtaposing sentences that express the various items of information contained in the semantic representation. The inferences and the conciseness of form that a language offers must also be taken into account. Moreover, there exist other procedures for linearization into sentences apart from juxtaposition; for example, relativization, subordination and coordination. However, the use of such procedures requires not only that they be feasible from a formal standpoint but also that they yield texts with the desired semantics. Consider the causal relation between the act described in *A policeman clubbed Mary* and the result described in *A policeman wounded Mary* and let us see how these two sentences can be conjoined by varying the linearization procedure, their order of appearance and their respective syntactic constructions. The texts

(1a) *Mary was wounded. She was clubbed by a policeman.*
(2a) *A policeman clubbed Mary, who was wounded.*
(3a) *A policeman clubbed Mary and he wounded her.*

can suggest a causal relation between the act and the result. However, this is not the case for texts obtained respectively from (1a), (2a) and (3a) when *club* and *wound* are inverted:

(1b) *Mary was clubbed. She was wounded by a policeman.*
(2b) *A policeman wounded Mary, who was clubbed.*
(3b) *A policeman wounded Mary and he clubbed her.*

These texts present the act and the result as two independent events. The differences of interpretation are not specific to the sentences *A policeman clubbed Mary* and *A policeman wounded Mary*. They can also be found in the act described in *John abandoned Mary* and the result described in *John shattered Mary*. The following texts:

(1′a) *Mary was shattered. She was abandoned by John.*
(2′a) *John abandoned Mary, who was shattered.*
(3′a) *John abandoned Mary and he shattered her.*

whose structures are respectively identical to those in (1a), (2a) and (3a), can suggest a causal relation. In contrast, the texts

(1′b) *Mary was abandoned. She was shattered by John.*
(2′b) *John shattered Mary, who was abandoned.*
(3′b) *John shattered Mary and he abandoned her.*

where the structures are respectively identical to those in (1b), (2b) and (3b), do not suggest any causal relation. So far as direct causal relations are concerned (i.e. an act which directly provokes a state), we will show that the options on linearization into sentences, the order of the information and the syntactic constructions offer a combinatorics of 36 possibilities. Of these, only 15 are appropriate in English or French in that they are formally feasible and give the desired semantics. These 15 discourse structures, which are neither semantically nor syntactically predictable, constitute linguistic data of a new kind – a 'discourse grammar'. The originality of the notion of discourse grammar introduced here lies in its ability to integrate both conceptual and linguistic decisions. The discourse grammar elaborated here applies only to direct causal relations. However, analogous discourse grammars could be devised for other types of semantic relationships: indirect causal relations, logical implications, dictionary definitions, and so on. It will be shown that the sole guarantee for generating well-formed texts, whatever the situation, lies in the use of such discourse grammars. The linguistic basis of a generation system will be developed in Chapter 3, and Chapter 4 will examine how a discourse grammar can be set up and used.

The interdependence of the various decisions and the need to use a discourse grammar have led us to devise a generation system that is modularized into two components: the strategic component and the syntactic component. For a given semantic representation, the strategic component makes the linguistic and conceptual decisions mentioned above and constructs 'text templates' that indicate how the text is to be linearized into sentences and what verbal forms are to express the main concepts, as well as the syntactic constructions that are to be used. Simplified examples of such text templates are:

T1 ACTOR *club* OBJECT *who be wounded.*

T2 OBJECT *be assassinated* DATE GEO. ACTOR *shoot* OBJECT *as* OBJECT *drive past in* CAR.

The syntactic component develops the text templates into texts. For T1 and T2, this could result in the production of the following texts:

A policeman clubbed Mary who was wounded.

John Kennedy was assassinated yesterday in Dallas.

A mentally unbalanced man shot the President of the United States as he was driving past in a convertible.

Text templates are annotated with syntactic data that does not appear in the examples given above. Both the syntactic data needed for the text templates and the operation of the syntactic component are based on work carried out at LADL, where a lexicon-grammar of the French language has been developed that covers 50,000 lexical entries used in 600 constructions. Our syntactic component is therefore based on a grammar of wide coverage rather than on a grammar fragment derived from a narrow corpus; the latter type is all too often used, despite the fact that this situation has been deplored by a number of researchers (Mann 1982).

Our generation model has been implemented in two domains: newspaper reports of terrorist crimes and answers in a help system for UNIX. These two generators will be described in Chapter 5. The first one, written in LISP, generates reports of terrorist crimes in a journalistic style. To illustrate the task it accomplishes, imagine an international press agency that operates in the following way: whenever an event takes place in a given country, a reporter carries out his investigation in the traditional way. He then writes up his 'report'. This is done by filling out an appropriate form (murder form, kidnapping form, car crash form, etc.) in an abstract code. This form is then dispatched, electronically of course, to different countries in the world where local journalists 'translate' it into their own language. In this scenario, our system does the work of the journalist who translates the completed form into the local language. The completed forms, which are the input to our generator, thus constitute our semantic representations. They contain the information that can be found in a daily newspaper and which is meant to be of interest to the average reader. For example, the terrorist crime form contains information on the identity of the target(s), the terrorist(s), the condition of the target(s) after the attack, the nature of the act committed by the terrorist(s), the city or country where the attack took place and the date. To simplify the task of completing forms in an abstract code we have created a menu which asks the user questions and fills in the form according to his answers. The idea of creating forms that are automatically translated into natural languages can be extended to other domains. In office automation, for example, forms could be created for synthesizing thank-you letters, estimates, expense chits, and so on. The domain of terrorist crime, distasteful though it is, offers the advantage of implicating temporal, spatial and causal relations, which are among the most common phenomena encountered in natural language texts and are also associated with the trickier aspects of language.

The generator produces texts in both English and French, as illustrated by the examples given below:

Fr *Indira Gandhi a été assassinée hier à New-Delhi. Deux extrémistes sikhs ont tiré sur le Premier Ministre Indien alors qu'elle quittait son domicile à pied pour se rendre à son bureau.*[2]

Eng *Indira Gandhi was assassinated yesterday in New-Delhi. Two Sikh extremists shot and killed the Indian Prime Minister as she left her home on foot to go to her office.*

Fr *Des anarchistes ont fait sauter un commissariat de police aujourd'hui à Paris tuant deux policiers et en blessant dix autres. La bombe, qui était cachée dans un camion garé devant le commissariat de police, contenait vingt kilos de dynamite.*

Eng *Anarchists blew up a police station today in Paris killing two policemen and wounding ten others. The bomb, which was hidden in a truck parked in front of the police station, contained twenty kilos of dynamite.*

The 'good quality' of these texts stems from the fact that our generator is based on sound linguistic knowledge.

The generator that produces answers in a help system for UNIX is simpler than the terrorist crime report generator. This difference is inherent in the domains themselves. The answers required in a help system do not involve subtle causal, temporal and spatial relations like those encountered in terrorist crime reports. Furthermore, they can be expressed in a technical vocabulary which raises fewer difficulties than the vocabulary of everyday language used to report terrorist crimes. The comparison between these two generators will highlight the portability of our generation model.

Finally, I will describe a semantic-representation-to-speech system; that is, a system that communicates the information of a semantic representation orally. This system is an extension of our written text generation system which has been modified and completed so as to perform phonetic conversion and to process prosody.

1 Introductory example: production of error messages

In this chapter a concrete example, the production of 'intelligent' error messages, is developed. Using this example, it is possible to give a general picture of the decisions that must be taken in a system of automatic generation, and a descriptive outline of the knowledge required for the setting up of such a system.

1.1 Presentation of the problem

Let us consider a system of file management in which the file names obey the following syntax:

⟨filename⟩ = ⟨name⟩ / ⟨name⟩.⟨extension⟩[1]

⟨name⟩ is a string of characters that contains neither blanks nor periods (e.g. ⟨name⟩ = ⟨FOO–BAR⟩)

⟨extension⟩ is a string of letters (e.g. ⟨extension⟩ = LPT)

The extension is used either to indicate how the file is to be used (e.g. ⟨extension⟩ = LISP for a program file written in LISP) or to pin-point different states of the same file (e.g. the file FOO–BAR.R designates a file ready to be formatted and the file FOO–BAR.LPT designates the same one, formatted and ready to be printed). We will assume that the files are divided into directories (e.g. directory ⟨Danlos.R⟩), the 'current directory' being the one that the user is working on at a given moment. We will also assume that the procedure for deleting files is carried out in two steps: the user first asks for a file to be deleted, the file then goes into the 'directory of deleted files'. A file can be recovered from this directory until the second step of the deletion procedure, when it is irretrievably destroyed.

Instead of making do with error messages like *file FOO–BAR not found* following a command to print the file FOO–BAR, one might prefer the machine to display error messages such as the following:

(1) *There is no file FOO–BAR, but there is a file FOO–BAT.*
Do you want to print FOO–BAT? (answer Y/N)

(2) *There is a file FOO–BAR.LCL and a file FOO–BAR.UCL.*
Do you want to print FOO–BAR.LCL? (answer Y/N)
Do you want to print FOO–BAR.UCL? (answer Y/N)

(3) *The file FOO–BAR is not in your current directory, but it is in your directory ⟨Danlos.R⟩.*
Do you still want to print it? (answer Y/N)

(4) *The file FOO–BAR has been deleted, but it is in your directory of deleted files.*
Do you still want to print it? (answer Y/N)

(5) *There is no file FOO–BAR, nor any extension of this file, nor any file with a related name.*

These messages imply that the machine is functioning in an interactive mode: the first four conclude with one or more questions to which the user replies with a yes or a no (Y/N). They presuppose the existence of a search algorithm capable of carrying out the following operations:

searching for files with names spelled differently from those indicated by the user, for example, a file whose name differs by one letter from the one requested. Message (1) corresponds to the case where there is one (and only one) file whose name is related to the requested file; this file is in the current directory, which is the default value for the location of a file;

searching for extensions of the name of the file to be printed. Message (2) is produced upon finding two extensions of the name of the file to be printed;[2]

searching for the file to be printed in different directories. Message (3) corresponds to the case where the file to be printed is in a directory other than the current directory or the directory of deleted files. Message (4) corresponds to the case where the file to be printed is in the directory of deleted files.

These five possibilities are of course not the only ones, and there is no reason why the search algorithm should be limited to these cases. One could envisage

several files in the current directory with related names,
and/or files with related names in other directories,
and/or different extensions in different directories,
and/or the file to be printed may be in different directories.

Whatever the search algorithm capable of carrying out these operations may be and whatever form the results of such an algorithm may take, the results can always be reduced to a table of the following form: