HAL
archives-ouvertes.fr

Understanding a technical language.A schema-based approach

P. Falzon

▶ To cite this version:

P. Falzon. Understanding a technical language.A schema-based approach. RR-0237, INRIA. 1983. inria-00076321

HAL Id: inria-00076321
https://hal.inria.fr/inria-00076321

Submitted on 24 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P 105
78153 Le Chesnay Cedex
France
Tél (3) 954 90 20

Rapports de Recherche

N° 237

UNDERSTANDING A TECHNICAL LANGUAGE

A schema-based approach

Pierre FALZON

Octobre 1983

UNDERSTANDING A TECHNICAL LANGUAGE

A schema-based approach

Pierre Falzon

June 1983

This research was supported by a grant from the Institut National de Recherche d'Informatique et d'Automatique (Rocquencourt, FRANCE), and completed at NASA - Ames Research Center (Moffett Field, CA, USA). This report has been published by both INRIA and NASA. The author wishes to thank Charles Billings, Renwick Curry and Everett Palmer for their help and support in this work.

RESUME

Les communications verbales en situation de travail amènent les opérateurs a élaborer un langage technique, restreint par rapport à la langue naturelle; la connaissance de ces restrictions spontanées peut être utiles pour la conception de langages de commande homme-machine. Dans cette perspective est étudie un langage de travail, utilisé par les contrôleurs aériens dans leurs communications avec les pilotes. Une méthode d'analyse est présentée, permettant la mise en évidence des schémas correspondant aux différentes catégories de messages, et de leurs éléments. Cette connaissance "schématique" est utilisée par des programmes qui font l'hypothèse que le caractère opératif des langages techniques (et notamment la restriction de l'univers du discours) permet de limiter les processus et les données nécessaires a la compréhension (monosémie, absence de prise en compte de certains mots, évocation des schémas par des mots-commandes, absence de syntaxe). Les programmes sont capables d'interprêter, et de traduire en séquences d'actions à entreprendre, les messages émis par les contrôleurs.

Mots-clefs: langage technique, schéma, compréhension du langage, planification.

ABSTRACT

Operators in work situations tend to develop technical languages, which are restricted subsets of natural language. A better knowledge of these restrictions could provide guidelines for the design of the restricted languages of interactive systems. In that perspective, a technical language (used by Air Traffic Controllers in their comunications with the pilots) is studied. A method of analysis is presented, allowing the identification of the schemata underlying each category of messages. This schematic knowledge has been implemented in programs, which assume that the goal-oriented aspect of technical languages (and particularly the restricted domain of discourse) limits the processes and the data necessary in order to understand the messages (monosemy, limited vocabulary, evocation of the schemata by some command words, absence of syntax). The programs can interpret, and translate into sequences of action, the messages emitted by the controllers.

Key-words: technical language, schema, language understanding, planning.

PAPIER RÉCUPÉRÉ ET RECYCLÉ

TABLE OF CONTENTS

INTRODUCTION

The instructions on the shampoo bottle read: "For best results, wet hair with warm water. Gently work in the first application. Rinse thoroughly and repeat". Hill (1972) was struck by the ambiguity, lack of precision, fuzziness, etc... of this text: repeat from where? he wondered. He then readily proposed a "much clearer" version:

```
for best results
BEGIN
wet hair with warm water
FOR j:=1,2 DO
  BEGIN
  gently work in application (j); rinse thoroughly
  END
END
```

Hill adds that, although he does not expect to see that on a shampoo bottle in his lifetime, he thinks that "it is something to be desired, far more than desiring to write plain English for computers". The author even goes farther, stating that "In my own Utopia, we shall be able to write instructions to people in programming languages, just as we do for computers".

One decade later, where do we stand? Much of the literature focuses now on the design of user-oriented systems, the language of which should be closer to natural language, although still restricted. The reason for that interest in the user is not to be found in a sudden improvement in precision and clarity of natural language, but rather in two major facts. The first one is the

modification of the user population: the users are often inexperienced with computers, and quite unwilling to learn anything about them. Computer experts now form only a small fraction of the prospective users of a system. The second one is the problems that may arise if the interface is not easy to use. This is especially true in situations where the main task to be accomplished is not the interaction with computers, but some other task in which some interaction with computers has become necessary. Think for example of the task of an aircraft pilot in a modern cockpit, or of the operator of a nuclear power plant.

The fundamental obstacle to a widespread use of computers, which once was their size and cost, has gradually become their lack of friendliness to the user. Not so long ago, Hill could assert that people had to adapt to computers. He would now meet much adverse reaction.

Nevertheless, the criticisms of natural language need to be taken into account. Some of them have long ago been stressed in a pioneer paper by Chapanis (1965). Language is currently being investigated by more and more Human Factors specialists, very much because of the reason already mentioned: the growing use of computers in our daily lives.

A first line of research has considered the possibility of using natural language as an interactive language. Although the last decade has seen impressive results in that field, the use of unrestricted natural language for computer interaction faces

serious problems.

The first one is that powerful natural language understanding systems are major undertakings, and need a considerable amount of computer memory; this is a serious obstacle for small computer systems. But we will not argue on this point; technological progress could change it.

The second problem is that, very often, these natural language systems look like a bulldozer trying to destroy a house of cards: there is often a huge difference between the complexity of the tool and the triviality of its application. For instance, and with all the respect due to Winograd's very sophisticated system (Winograd, 1972), it is somewhat disappointing to see that it can only function when applied to the very small world of block manipulation! And the worst is (according to Petrick, 1976) that the bulldozer sometimes cannot manage to destroy the house of cards!

A second line of research took a different approach: at least in the foreseeable future, man-machine interfaces will not use natural language, but some restricted dialect. Is it then possible to restrict the possibilities of natural language without greatly constraining the user? A number of authors have explored this question, studying the effects of different imposed restrictions of syntax or vocabulary (cf for example Ehrenreich, 1981; also Kelly and Chapanis, 1977). A corollary of this approach is the definition of appropriate vocabularies (cf the studies on

naming by Scapin, 1981 & 1982; also Rosenberg, 1982), and appropriate syntactic structures (Hammond et al, 1980).

However, another approach is possible. Instead of studying specific restrictions of natural language, why not study the natural restrictions of specific languages? In any work situation where the operators have to communicate verbally, the language they use is not unrestricted natural language; the operators tend to build a specific language, molded by the characteristics of the task and its objective. These task-oriented languages transform natural language into a dialect, totally obscure to a non-specialist of the domain, although it will be entirely clear to any expert. Consider the following communication, emitted by an Air Traffic controller to a pilot: "Intercept the 1-3-5 of Point Reyes and resume the SID and with the restrictions". This message is total nonsense for a non-specialist hearer: first because of the abbreviations (what can be a SID ?), secondly because of the technical meaning of some words (what restrictions are you talking about?), and thirdly because of the concepts involved (what can be a "1-3-5 of Point Reyes"? I guess it is some object, since I am supposed to intercept it. I'd better watch out, someone is probably going to throw it to me). Linguistic competence is not enough to understand a technical language.

However, one may argue that this is mainly a question of vocabulary, and that, given an appropriate technical dictionary, the communications are just a sample of natural language. But there is more here than a vocabulary specialization. Several

factors contribute to the modification of natural language in work situations:

- workload first, which tends to make the operators restrict the length of the messages and concatenate several messages into one (Sperandio, 1969);

- the necessity to avoid ambiguity, which restricts the meanings of the words and the form of the emitted messages. This is especially true in complex situations where the risks involved are important, and often leads to the recommendation of a specific phraseology;

- the influence of a common field of work, making the reference worlds and the goals of the actors the same. The restriction of the domain of discourse has two important consequences. First, given a sufficient knowledge of the domain, the possible topics are highly predictable. In Air Traffic Control (ATC) for example, it is of very low probability to hear "Would you mind passing me the salt?", but one is prepared to hear about levels, headings, and other flight parameters. The operators are only interested in some of the properties of reality. Second, these topics are seen under a specific, distorted point of view (cf Dupre, 1981, for an illustration of this point). Restrictions on the domain of discourse limit not only the number and the type of possible topics, but also the viewpoint from which these topics are considered.

Goal-oriented languages can then be thought as being

restricted, as compared to natural language, in a number of domains: vocabulary, syntax, field of discourse, dynamics of the dialogue are a few. A better knowledge of these "spontaneous" restrictions and of the way they are built could provide guidelines for the design of computer interfaces.

Very little work has been done in the above perspective in the Human Factors area (with the notable exceptions of Thomas, 1976 & 1978, and of the series of studies conducted by Chapanis and his colleagues: cf Chapanis, 1978, for a summary). In the psycholinguistic field, the research is rarely relevant; in fact, most of it has focused on a-contextual situations, trying to find general characteristics of language. However, there seems to be a recent change in this trend, with a growing interest in the influence of specific situations on the type of emitted expressions (Gibbs, 1979, 1981; Clark & Lucy, 1975; Hupet & Costermans, 1982).

The research presented here follows these premises. It focuses on a very specific language, used by air traffic controllers in their communications to the pilots. In the first part of the text, a method of analysis will be presented. The results of this analysis are evaluated through the development of a language understanding system, presented in the second part of the text.

SOME RESULTS AND AN HYPOTHESIS

A first study (Falzon, 1982) has focused on the vocabularies (French and English) and on the forms of expression used by the controllers (if two messages are not composed of the same words in the same order, they are two different forms of expression).

Different measures on the use of words and of messages have allowed the design of restricted vocabularies, subsets of the total vocabularies the controllers use. The use of these restricted vocabularies allow the recognition of a large number of messages ("Recognition" is given here a very simple definition: a message is said to be recognized if all of its words belong to the restricted vocabulary).

This result stresses a first "natural" restriction made by the operators, concerning the vocabularies (similar results have been observed by Michaelis et al, 1977, in laboratory experiments). The interesting point is that the vocabularies, though restricted, nonetheless allow much flexibility in the form of expression of the messages, since 60% of the different forms of expression in French (i.e. 528 different utterances), and 73% in English (i.e. 380 different utterances) are recognized.

However, the recognition performances of the restricted vocabularies vary according to the category of messages, and are very poor for some categories. The reasons for this seem to be linked first to the length of the messages ("rare" words are more likely in a long message than in a short one), second to the frequency

of use of the category (in order for "conventional" expressions to appear, the category must be used frequently). Consider for example the "traffic information" category (the controller warns a pilot of the presence of another aircraft in his vicinity). The messages of this category are not very frequent, and tend to be lengthy. The restricted vocabularies are not large enough to include all the words used in these messages; according to the definition of recognition that has been given, these messages then cannot be recognized.

This last result has led to a change in our approach. Although the "Traffic Information" messages are not recognized, they are easy to spot: most of them include the two words "traffic" and "information". Thus, they can easily be categorized. Moreover, the type of information they mention is highly predictable: the pilot expects to hear the altitude, heading, relative position of the other aircraft. A similar analysis can be applied to all categories of messages: each category can be characterized by some "command" words and by a sequence of possible constituents. The messages of a category may be considered as a list of different instances of a common schema, as different actualizations of a single schema. As will be seen, the analysis of the different forms of expression provides the compulsory and optional elements of the messages, and the default values that are assumed in some cases.

The understanding of a message can then be (extremely roughly) divided into two phases. In a first phase, the

understanding process is data-driven: the words heard (or read) are processed and activate a previously stored schema. In a second phase, the process becomes conceptually-driven: schemata are pre-defined representations which ask for specific information. Thus, the input is checked to see whether it can fill in the information slots of the schema. If this process is successful the schema is validated. A more detailed presentation of schema theory is given in the next chapter.

Our hypothesis is that the universe of discourse is so restricted that syntax can often be neglected, provided that a sufficient knowledge of the task domain is available. This does not mean that we assume that the pilots never use their linguistic competence to understand the messages they receive; our point is only that it is possible to understand the messages without using much syntactical knowledge. The objective of this work is to test whether this assertion holds true, and to what degree. Since we are not dealing here with language in general, but with a specific dialect in use in a very specific context, we may feel free to use whatever analysis seems convenient, keeping in mind that we are not using (and even less building) any theory of grammar. In fact, it could even be said that a "standard" linguistic approach (if such a thing exists), would not be relevant: the interesting point is to see how a technical language differs from natural language, and not to try to analyze it through the methods which are proper to the study of natural language.

The approach we will follow is then:

* for each category of messages, the underlying schema (or schemata) must be constructed. A method of schema abstraction will be presented. A word dictionary is built.

* in order to test our hypothesis, this schema knowledge is implemented in understanding programs, which must be able to understand messages in the technical language under study. Although the programs will be provided with typed input, there will be no punctuation whatsoever, in order to remain as close as possible to spoken language.

The assumptions we make are the following:

* Syntax: the system will use as little syntax as possible. There will be no grammatical parsing. The only clue that will be used is the order of the words within a message, and the order of the messages within a communication. We assume that messages begin with a command word, and that the missing elements of a message are to be found in the immediately preceding message.

* Words: not all of the emitted words will be found in the dictionary. On top of that, each word will be given a single definition: we assume that the restricted domain of discourse forbids polysemy.

LANGUAGE UNDERSTANDING AND SCHEMA THEORY

Schemata are a fundamental issue for the study of memory organization. They do not stand alone in the field of cognitive science: schemata have close links with Minsky's frames (1974) and Schank and Abelson's scripts (1977). However, for the sake of clarity, I will only use the word schema. A thorough discussion of the schema theory can be found in Alba & Hasher (1983). The following presentation will focus on the application of schema theory to language comprehension, borrowing from Rumelhart (1978).

A schema is a data structure for representing the generic concepts stored in memory. Schemata can represent objects, situations, events, actions and sequences of actions. A schema contains variables which can take different values: the values a variable can take are limited by variable constraints. These constraints are specified for each variable of the schema; however, the set (or the range) of possible values of a variable may also depend on the value of another variable. The constraints have two important functions: first they allow the correlation between the input data and the variables of the schema (this is referred to as the slot-filling process), and second they may be used as default values when the input does not specify any. This process of filling-in the slots of the schema is generally called (in the Artificial Intelligence literature) the instantiation of the schema; Piaget uses the word "accomodation" to refer to a similar process.

Once a schema has been activated, it can be accepted or rejected: this depends on the quality of its fit to the data. This evaluation is necessary, because we do not wait until the end of a sentence to initiate a schema: we pick up the first elements and make an hypothesis, i.e. we activate a schema; the following data allow us to test the hypothesis we made (in this respect, language understanding can be considered as a problem-solving activity).

Each schema is a network of sub-schemata: these sub-schemata are the conceptual components of the general concept being represented.

The elements of a schema must not be identified with the words of a sentence, for several reasons; one of them is that the units of the conceptual level are in a way "smaller" than words. A given word may include several units of a schema (cf for example Abrahamson, 1975, analysing verbs of movement). An illustration of this point is given below.

The processing of paraphrases is a quite important issue in language understanding research: in fact, many authors see it as one of the fundamental criteria in the evaluation of a language understanding system or theory (for example, Norman & Rumelhart, 1975; Anderson & Bower, 1973; Schank, 1975). The semantic representations must be invariant under paraphrases of the same information.

Paraphrases (and near-paraphrases) are important in that they

can give insightful clues on the way information is stored. I will here borrow some examples from Rumelhart & Norman (1975). Consider sentences A and B:

A - Henry went to a store
B - Henry drove to a store

A and B are not paraphrases, but we feel they are closely related. In fact, the meaning of A seems to be included in the meaning of B. This means that this meaning of "drove" contains this meaning of "went". For example, we could say that the meaning of "drove" is [CHANGE OF LOCATION BY MEANS OF AN AUTOMOBILE], and the meaning of "went" [CHANGE OF LOCATION]. Compare now B and C:

C - Henry ran to a store

B and C are not paraphrases either, we feel they are related, but not in the same way as A and B. Neither of them is included in the meaning of the other. This points out that "ran" and "drove" probably share some common semantic elements. For example, we could say that the meaning of "ran" is [CHANGE OF LOCATION IN A QUICK PEDESTRIAN WAY]. The common elements are then [CHANGE OF LOCATION].

In A, nothing is said about the way the action is performed; nevertheless, this does not mean that we have no idea about it. Many American readers would probably think that Henry took his car, i.e. they would assume a <u>default value</u> to that unspecified slot of the schema. It is worth noticing here that this

assumption could be different in other situations: supposing the action took place in Paris, people would rather think that Henry took the subway, or that he walked to the store. The default values are context-dependent.

METHODOLOGY

THE CORPUS

Two different sources of pilot-controllers communications have been available. The first one is a set of recordings in a preceding study (Falzon, 1982). This corpus represented 20 hours of pilot-controller communications, and a total of 7700 messages, already categorized. The second source consisted of transcripts of 4 one-hour flights (Los Angeles-San Francisco), recorded in the cockpit of an aircraft. These two sets of recordings differ in several ways:

- the first recordings were made in an Air Traffic Control Center: they focus on specific sectors of control, crossed by different flights; the second recordings focus on specific flights, crossing different sectors.

- the first recordings were made in France, the second in the USA: in the first case, the controllers used either French or English in their communications (according to the language of the pilot); in the present study, only the messages emitted in English have been considered. The language used in the two recordings may differ for two reasons: first because English is not the native tongue of the controllers recorded in France, second because, although the domain of discourse is the same, the linguistic habits of French and American controllers may differ.

The two sources have been used, with a bias towards the util-

ization of the American transcripts. Many messages of a single category are necessary in order to define the schema of the category: the characteristics of the corpus prevents some categories from being sufficiently exemplified (for instance, there is little data on taxiing, taking-off, landing and final approaches). However, the transcripts are sufficient to allow the analysis of the other phases of the flight.

SCHEMATA AND CATEGORIES

Previous work on pilot-controller communications has led to a categorization of the messages (cf Janet, 1981; Falzon, 1982; Hunter et al, 1974): this categorization can be seen as an attempt to classify the messages according to the schemata they evoke. Each category thus represents a schema, and the different forms of expression are different actualizations of the schema. In some cases, however, a further classification is necessary inside the category. For example, one category deals with instructions related to changes in the heading, route or course of the aircraft (horizontal movements). There are four possible actions concerning the horizontal movements of an aircraft (speed excluded): maintain (heading, track, etc ...), modify (same), intercept, depart. Four different schemata are necessary to account for these four different horizontal actions.

Each category (or sub-category) of messages may be seen as a set of paraphrases or near-paraphrases, exemplifying a single schema. The study of paraphrases and of meaning overlaps can pro-

vide us with an experimental tool in the analysis of the underlying representations. The successive steps of this analysis will now be described.

SCHEMATA AND THEIR ELEMENTS

For each schema, specific information appear. For example, a "depart" action will always mention a "from" position and a direction; an "intercept" action will mention a radial and a VOR (a navigation aid), or, more generally, a track, e.g. "intercept the ILS course"; a "maintain" action may very well mention nothing: the value to be maintained (and sometimes even its nature) will have to be found in context. This means that the expectations are different for each schema, i.e. that we are able to specify the slots we will need to fill.

The approach will be illustrated by an example of analysis for one category of messages, dealing with modifications of the level of an aircraft. Consider these simple messages:

1 - climb level 330
2 - descend level 330
3 - leave 290 for 330

The schema will be built step by step. First, we know that all the messages of this category deal with actions in the vertical dimension, and more specifically with a change of level (as opposed to a change in the rate of climb or descent). These elements must be specified in the schema:

CHLVL : ((Act: VE)(Nat: LVL))

CHLVL (for: CHange LeVeL) is the name of the schema. VE means that the action (Act) taking place concerns the vertical (VE) dimension. LVL (level) indicates the nature (Nat) of the action. All schemata are composed of a list of pairs of items; in each pair, the first element (in lower case) is the role, the second (in capitals) is the filler. As we will see, the filler can include several elements, among which a choice must be made.

If we now consider message 1, we see that a first element is missing in the schema: the level to be reached. Moreover, we see that messages 1 and 2 do not indicate the same type of relation between the present level and the level to be reached: the schema then needs to include these elements.

CHLVL : ((Act: VE)(Nat: LVL)(Rel: (+ -))(To: P))

Rel stands for relation: + indicates that the level to be reached is above the present level, - that it is below. "To" indicates the level to be reached, P stands for parameter.

In message 3, a new information appears: the present level. This element needs to be taken into account by CHLVL, but what about messages 1 and 2 ? In these messages, the present level is not mentioned, but message 1, for example, could be re-written as "from your present level, climb level 330". We then also need to specify that, if the level that is left is not mentioned in a message, it must be the present level.

```
CHLVL : ((Act: VE)(Nat: LVL)
         (Rel: (+ -))(From: (PV P))(To: P))
```

From indicates the level that has to be left. PV stands for "Present Value". Consider now message 4:

4 - climb level 330 at pilot's discretion

The expression "pilot's discretion" has several implications: without giving too many details now, it indicates that the action can be delayed (the pilot may wait before changing his altitude), and that the pilot has more latitude in the accomplishment of the instruction. Anyway, this information must be taken into account by the schema.

```
CHLVL : ((Act: VE)(Nat: LVL)
         (Rel: (+ -))(From: (PV P))(To: P)
         (Time: (def NOW PD)))
```

PD stands for "pilot's discretion". NOW is a default value, assumed when the message does not mention the "Time" information. Messages 1, 2 and 3 for example do not specify when the action has to take place. In that case, a default value is specified in the schema: "def" stands for "default", and indicates that the next filler is to be chosen if the message does not specify any. "def" is written in lower case because it is not really a filler, but only a flag pointing to the filler next to it.

A brief remark: "Time" is probably not the best way to name this role: as we have seen, "pilot's discretion" indicates not only when, but also how, the action is to be executed. Another example can be found in comparing "now" to "immediately". Both words mean that the action must begin upon reception of the

instruction, but "immediately" implies also a specific way to perform the action: it includes a notion of urgency, meaning that, for example, maximum thrust should be used.

In some cases, the default value will be nil. For example, compare these two communications (dealing with modifications of the horizontal movements of the aircraft), each composed of two messages:

(5) a - fly heading 230.
    b - receiving Avenal proceed direct.
(6) a - fly heading 230 until receiving Avenal.
    b - then proceed direct.

From 6a, we can infer that this type of message (heading change) may mention a limit (until...). But we also notice that 5a does not mention it. The same limit information is to be found in fact in the next message 5b. In the same way, message 5b mentions a condition whereas message 6b does not (the condition is to be found in 6a). Because of these phenomena, we need to know that a "heading change" schema may, or may not, include a limit, and that a "route change" schema may, or may not, include a condition. This is the reason why we need the possibility for the default values to be nil.

The same analysis, applied to all categories of messages, provides a dictionary of schemata.

WORDS DEFINITION

Compare the following expressions:

1 - climb level 330
7 - climb flight level 330
8 - climb to the flight level 330
9 - climb 330

These four messages are paraphrases. From their examination, we can infer that some of the words that are used are not needed for the understanding of the messages. For example, the information given by "to", "the", "flight" and even "level" is not useful: when hearing "climb", the pilot immediately knows that the instruction refers to a modification of the flight level, and a parameter is expected. Thus, much of what is said can be discarded. Therefore, the number of words that the system will have to know is only a subset of the different words that are used: the dictionary does not need to include the words mentioned above (in fact, "level" is defined in the dictionary: it is needed to understand other messages).

The words are defined using the same elements that have been used in the definition of the schemata. It is interesting here to compare the definition of "climb" and "leave" to the corresponding schema definition (CHLVL):

```
CHLVL : ((Act: VE)(Nat: LVL)
         (Rel: (+ -))(From: (PV P))(To: P)
         (Time: (def NOW PD)))

climb : ((Act: VE)(Nat: LVL)
         (Rel: +)(From: PV))

leave : ((Act: VE)(Nat: LVL))
```

The definitions of "climb" and "leave" differ in two ways: first, climb expresses a relation (Rel: +), whereas leave is

neutral in that respect; second, a message using "climb" will not mention the present level of the aircraft: this is made clear by the presence in the dictionary definition of (From: PV), which indicates to the system that no present level is to be expected.

THE PROGRAMS

AN OVERVIEW

The system is composed of two sets of programs:

- "understanding" programs - These programs are provided with ATC communications, composed of one or several messages. They "translate" the input, first finding an appropriate pre-defined schema (among several others), then filling in the slots of the schema. There are two programs: Schematch and Dicolisp. Schematch is the main processing program: it processes the words of a communication, matching them to evoked schemata, and then stores the instantiated schemata in memory. Dicolisp includes a dictionary of words and schemata and specialized subprograms adapted to the different schemata. In fact, each schema cannot be considered apart from its subprogram: it is a data structure plus a set of operations, a representation.

- "planning" programs - These programs are provided with the filled schemata which are the output of the "understanding" programs. Their job is to transform these schemata into sequences of actions. The only processed schemata are those related with movements in the horizontal plane (the program is "Planho") or in the vertical dimension ("Planve"). Other schemata are produced by the "understanding" programs (for example, schemata related to frequency changes, "report" orders, politeness, questions), but are not taken into account in the planning programs. The outputs of

the two planning programs differ: "Planho" produces a sequence of legs, "Planve" produces a single frame, divided in three parts: the core (the fundamental action), the rules (i.e. descent at pilot's discretion until level X) and the constraints (i.e. cross a specified VOR at a specified level).

In fact, to call Schematch and Dicolisp "understanding programs" is somewhat inappropriate: the system can be said to understand, since it exhibits an "intelligent" behavior (through the production of plans of action), but what can we say about the output of Schematch ? To call Schematch and Dicolisp "parsing programs" would be misleading too. For lack of a better name, though, we will here stick to the name "understanding programs".

All programs are written in LISP, and implemented on the UNIX system at NASA Ames Research Center.

THE UNDERSTANDING PROGRAMS

Schematch and Dicolisp

This presentation will try to avoid going into too much detail. We will focus on one particular example, and leave aside some minor aspects of the programs.

Schematch is given as input a communication, i.e. a set of messages with no punctuation, and processes it word by word. There are four different kinds of words:

- unknown words: these are words which cannot be found in

Dicolisp and which are not numbers. The processor drops them and goes to the next word.

- numbers: numbers are not referenced in the dictionary. They are directly recognized as parameters by the general processor. They are given in their numeric form, not spelled.

- names of "places": these can be VORs, Airports, ATC centers, etc... These names can actually be composed of several words (Los Angeles, Santa Monica). The system transforms those into single words (Los-Angeles, Santa-Monica). They are referenced in Dicolisp as "Places".

- dictionary words: these are the words that can be found in Dicolisp. Each word has a single definition (no polysemy). This definition is composed of a list of role-filler pairs; a definition may be a single pair. For example, "now" is just (Time: NOW). Other words have more complex definitions (cf "climb" and "leave" in the preceding chapter).

Some of the dictionary words have a special property: they evoke a specific schema. Words like climb, descend, fly, turn, contact, intercept, etc... are schema-associated. When the processor finds one of those, it knows that it must open a new schema. Let us consider a simple case: the communication is "climb level 230".

The first word is schema-associated: the processor loads the appropriate schema, called CHLVL, and opens an empty list, called

Bindings, which will receive the instantiated elements of the schema; the first element of Bindings is the name of the activated schema:

```
SCHEMA: ((Act:(VE))(Nat:(LVL))(Rel:(+ -)).
         (From:(def PV P))(To:(P))
         (Time:(def NOW ANY)))
BINDINGS: (CHLVL)
```

The rest of the process consists in the creation of an instantiated schema (Bindings) using both the activated schema and the information given by the words of the message. First, the definition of "climb" is called for:

climb: ((Act: VE)(Nat: LVL)(Rel: +)(From: PV))

A pattern matcher compares the definition of climb and the schema. Each corresponding element is written in BINDINGS, and deleted from SCHEMA. We obtain:

```
SCHEMA: ((To:(P))(Time:(def NOW ANY)))
BINDINGS: (CHLVL (Act: VE)(Nat: LVL)(Rel: +)(From: PV))
```

All the pairs of the definition of climb have been processed and no discrepancies have appeared: the processor considers then the next word, "level". It is not a schema-associated word, its definition is then loaded:

level: ((Nat: LVL))

This pair cannot find its match in SCHEMA. BINDINGS is then checked to see if this absence is caused by redundancy. This is

the case, the information given by "level" already existed in "climb". The next word is then considered. It is "230", a number. By definition, a number is never schema-associated, and will not be looked for in the dictionary: it is internally coded as P (for parameter). The program processes numbers (and places) in a specific way (the matching process differs). Anyway, P is found in the schema, yielding the following result:

```
  SCHEMA: ((Time:(def NOW ANY)))
BINDINGS: (CHLVL(Act: VE)(Nat: LVL)(Rel: +)
                (From: PV)(To: 230))
```

Fine, what is the next word ? Oh, there is no next word !! We must conclude. The point is, we still have some information in SCHEMA and must do something about it (this is a rule: a schema may not be abandoned unless it is empty). There is no problem, since we are provided with a default value for Time. Let's do that:

```
  SCHEMA: .....(nil).....
BINDINGS: (CHLVL(Act: VE)(Nat: LVL)(Rel: +)
                (From: PV)(To: 230)(Time: NOW))
```

Are we finished with the process ? Not yet. Each schema has specific procedures attached to it, allowing different checks. For example, we could have been unable to empty SCHEMA, because some information was missing: the procedures would have tried some heuristics to find the missing information. In our present example it is not the case, but consider the message: "leave 230 for 290". As we have already seen, "leave" does not specify the

type of Relation (+ or -), but this relation can be inferred from the values of the parameters: here, the schema procedure would infer a (Rel: +) from the pairs (From: 230) and (To: 290).

Another role of these specialized sub-programs is to format the output in order to facilitate its use by the planning programs. In the present example, though, the procedures do not help much, and the instantiated schema is stored as it is in Memory.

In the example we have studied, the schema was closed because there were no more words to process. Another reason to close a schema is when the processor finds a schema-associated word; two cases may occur. Consider (1) and (2):

(1) climb level 230 turn left heading 150
(2) climb and maintain level 230

In (1), the processor finds "turn", which is schema-associated, and tries (and manages) to close the active schema. A new schema is then loaded (CHHDG, for: change heading), and the process continues. In (2), the processor finds "maintain", which also is schema-associated, but cannot manage to close the schema: it needs a parameter. In this case, a second schema is opened in parallel, and the following words are processed for both schemata. It is here interesting to consider another situation:

(3) climb level 230 and maintain

In (3), when the processor finds "maintain", it tries to close the schema and succeeds. It then stores the instantiated

schema in Memory, loads the schema associated with "maintain", and tries to process the message. But "maintain" is the last word; the problem is not so much that the processor is waiting for a parameter: obviously a default value could be used; the problem is that the processor does not know what is to be maintained (speed, level, heading,...?). In this case, the schema-associated procedure will explore the Memory, assuming that the last property being talked of is the one the value of which is to be maintained.

Discussion

It is difficult at this point to give an evaluation of the "understanding" programs in terms of percentage of understood messages: in order to do this, a larger sample of communications would be necessary. However, as far as the corpus we have access to is concerned, the programs are quite successful. They are able to recognize much of Air Traffic Control instructions, and this despite the facts that the system has no syntactical knowledge, a limited dictionary, and a single definition for each word of the dictionary.

The simplicity of the understanding programs is of great interest: how is it that the programs are able to understand so much of ATC communications, being so simple?

The reason is that the messages are rarely elaborate. A previous study (Falzon, 1982) has shown that, especially for the categories of messages which have a high frequency, the

controllers tend to stick to some standard (and simple) forms of expression. Although variations do occur, they tend to be organized along common general patterns. For these reasons, the syntactical blindness of the system is not an obstacle, because of the syntactical simplicity and stereotypy of this technical language. Most messages begin with some sort of "command" word, and this could even be considered as a characteristic of restricted natural languages (or at least of this particular one). In this respect, it may be more fruitful to compare the ATC language to a computer command language rather than to natural language, i.e. to describe its syntax in terms of operators and operands, rather than in terms of generative rules.

The monosemy of the words can be a characteristic of technical languages, because of the restricted domain of discourse, and because of the necessity to avoid ambiguity. In any case, the fact that each word only has a single meaning does not seem to be a problem in decoding the ATC communications. In the same way, the fact that we are able to pay no attention to words that are not defined in Dicolisp is an interesting feature.

The very simplicity of Schematch and Dicolisp is then in itself a result. It proves something about the "natural" (i.e. user-originated) restrictions of natural language. Some of these restrictions have already been pointed out: other characteristics will be found when implementing the planning programs.

Still, the present programs are quite certainly not enough to

understand all possible Air Traffic Control messages. Although the human operators are willing, because they are operators, to restrict themselves to some standard phraseology most of the time, they still have the possibility, because they are human, to switch back to the use of natural language, when they want or need to do so (for example, in case of a low workload, or when the situation is rare, so that there is no adequate usual phraseology). In such cases, a more elaborate system would be necessary in order to understand the communications. Does this mean that the programs are useless? Certainly not.

The language of the controllers is not homogeneous. Most of the time, the language they use is a technical dialect, which vocabulary and syntax are highly restricted. However, they may use other expressions and a different vocabulary in less usual situations. The fact that the programs are not able to understand all of what is said is a consequence of the use of two different modalities in the communications to the pilots, for which different analyses must be performed.

THE PLANNING PROGRAMS

Again, a complete description of the two planning programs (Planho and Planve) will not be given; we will only outline their main characteristics and differences. The two planning programs consider that a single aircraft is dealt with: this restriction has been introduced to limit the number of flightplans the system has to know of, but it can easily be changed, provided that

information is given about the different aircraft. The meanings of the messages are based on the interpretation given in the Airman's Information Manual (FAA, 1982).

Planho

Planho is the program dealing with the horizontal plane. It needs two types of input:

- a set of instantiated schemata (messages processed by the "understanding" programs). Planho only processes those schemata dealing with horizontal movements. In order to do that, it filters the Memory, retrieving only the relevant schemata.

- the present flightplan (before updating). The flightplan is a list of legs, the first leg is assumed to be the active one.

Each leg is composed of three elements: a Trigger, a Direction and a Limit. The trigger is the position at which the action begins, the direction indicates the heading, or radial, to be followed, and the limit indicates the end of the leg (if A and B are two successive legs, the limit of leg A is then the trigger of leg B).

A word or two about the definition of directions, limits and triggers. Limits and triggers are positions. They can be defined in several ways.

* First, by a VOR, a radial and a distance on that radial: for instance, (D-30 R-160 Avenal) means 30 miles from Avenal on its 160 radial (an interesting case occurs when the distance and

radial are not mentioned, as in "receiving Avenal proceed direct": in this case, the point will be defined as (D-any R-any Avenal), meaning the first position meeting this condition).

* Second, by the name of the VOR itself: this is in fact a special case of the preceding definition, "Avenal" for instance meaning in fact "at a minimum distance from Avenal on any radial", i.e. (D-min R-any Avenal).

* Third, by the intersection of two directions: for example, ((H-130 Pesca) (R-160 Avenal)) is the intersection between heading 130 from Pesca and the 160 radial of Avenal.

* Fourth, by the present position of the aircraft (on which the program has very little information): it is then coded "H&N", for "here and now" (or for "hic et nunc", which makes it more intellectual).

We have seen how directions are coded: they always mention a reference point (VOR or not); this reference point may be H&N: (H-120 H&N) is then "heading 120 from wherever you are now".

Planho creates a sequence of legs according to the input messages, and then compares this sequence of legs to the flightplan, creating new legs and inserting them in the flightplan, deleting some legs from the flightplan when necessary. The output of Planho is then a new flightplan, which is considered valid only if it is a continuous sequence of legs (i.e. if the limit of each leg is the trigger of the following one): the validity is checked by the program.

It should not be inferred from this short summary that a

single leg is created for each message: some messages imply the creation, or the modification, of several legs ("intercept" messages, for example).

Planve

Planve, like Planho, filters the output of Schematch in order to process only the messages dealing with vertical information. The result of this processing is not a sequence of steps, but a single step, which includes three parts:

* the core: this indicates the fundamental general action of the step. It mentions the type of action, its limits, and its conditions. For instance, the core:

(Act: -)(From: 330)(To: 200)(Cond: PD)

means that the general action is a descent from the level 330 to the level 200, and that the action may begin at pilot's discretion (PD). The conditions may be of several different kinds: they may specify a specific position (if the "from" level is to be maintained until this position before descent), a specific speed (if the aircraft is supposed to reduce its speed before descent),... The conditions can be compared to the triggers of the legs (but they are not exactly the same thing).

* the rules: they indicate the degree of freedom of the pilots during the different substeps of the core. If a descent instruction mentions "at pilot's discretion", it means first, as we have seen, that the action may be delayed, but also that the

way to conduct the descent is not submitted to the standard rules of descent: for example, the pilot is allowed to level during descent if he wants to, to vary the rate of descent, etc. But "pilot's discretion" does not mean no rule at all: for instance, the pilot is not allowed to climb (i.e. the rate can be zero, but cannot be positive). The "PD" rules may be more lenient, but they still are rules. Each rule mentions the type of rule and its limit, e.g. ((Rule: PD)(To: 200)).

* the constraints: they specify the altitude restrictions the aircraft has to comply with. The constraints mention a position and its associated restrictions. For example, (Fillmore =" 240) means that the aircraft must cross Fillmore at or above the level 240.

Let us examine an example: suppose the instruction was "cross Fillmore at or above 240 maintain 200". The communication is first analyzed by the "understanding" programs, then processed by Planve. Supposing the aircraft was previously steady at 330, the output of Planve is:

```
(CORE (Act: -)(From: 330)(To: 200)(Cond: PD))
(RULES ((Rule: PD)(To: 240))
       ((Rule: ST)(To: 200)))
(CONSTRAINTS (Fillmore =" 240))
```

Implied in the "cross" schema are several elements: first, there is an implicit action: the aircraft is supposed to change its flight level (the fact that it is a descent is inferred from the values of the parameters). The target level (i.e. the "To"

level) is temporarily set at 240. Second, if not otherwise specified, "cross" implies that the action may begin at pilot's discretion: this is why the Condition is PD. Third, the rules to apply to the descent are also at pilot's discretion, until the critical level is reached. And fourth, obviously, the constraint is to reach Fillmore at or above 240.

Implied in the "maintain" schema is an instruction to... descend, to the level 200. The "To" element of the core is then changed from 240 to 200. We need then to know the rule to follow between 240 and 200. Nothing is specified in the communication: in that case, the default value is "ST" (for "standard"), meaning another set of rules of vertical movements (no levelling, constant rate, etc.). A new rule is then added to RULES.

Discussion

The two planning programs can be analyzed using the schema theory framework: both programs are composed of a data structure and a set of operations. The following paragraphs exemplify this assertion.

Planho has a standard structure (Trigger/Direction/Limit) for composing each leg of the flightplan and different operations to process the sequence of legs. For example, it knows it has to link the successive legs (through Limit and Trigger). When this link is not specified in the input, the program uses a characteristic of language, i.e. that, unless otherwise specified, the emission order of the messages of a communication corresponds to

the order in which the actions have taken (or will take) place. Then, if message A is given before message B, Planho assumes that the legs implied by A are antecedent (and probably immediately antecedent) to the legs implied by B, so that the missing trigger or limit can be inferred, and the legs linked. Of course, this is not always the case: some messages (like the "depart" messages) specify the position where the action will take place. These messages are processed accordingly.

Planve also has a standard structure (Core/Rules/Constraints) to describe the vertical step, on which different operations are applied.

For instance, the program knows that rules must be specified for all levels belonging to the core. If no rule is specified, an inference is constructed to bridge the gap, assuming a default value ("standard" rules).

In the same way, if a substep is missing, it will be built (e.g. one cannot maintain a given level unless this level has before been reached: as has already been explained, Planve will infer the missing action).

Another operation specifies that, whatever is the order of the messages, the constraints must be organized in the order in which they have to be met, i.e. following the vertical movement of the aircraft.

A last example: the program simplifies the rules, by concatenating some of them. If rule A is to be obeyed between the levels 300 and 230, and then between the levels 230 and 200, the

rules will be simplified by applying rule A from 300 to 200, creating a single substep. This does not mean that the pilot will apply only one of the possible modes of descent during this substep, but that any mode he will choose will have to comply with the rule of the substep.

The main criticism of the planning programs will focus on their psychological validity. There is no evidence that the structure of the pilot's representation of the future flightplan (vertically and horizontally) is similar to the structure of the outputs of Planve and Planho. These outputs can only be considered as one of the possible results of the planning activity. A good proof of that is that several different formats have been tried during the writing of the programs: for example, in the first version of Planve, the output was not a single step, but a series of successive steps, comparable to the output of Planho. The structures that were finally adopted have been chosen only because they seemed to have a good a priori probability to reflect the structure of the pilots' plans (knowing of course that they are only structures, not complete descriptions). For example, one can think that a given leg has little influence on the following one, whereas a level constraint at a given position may have consequences on the whole vertical step, or at least on the preceding substep. This explains the structural difference between the outputs of the two programs (sequence of legs vs. single step).

A second criticism (related to the first one) could also bear

upon the fact that the two programs process separately the vertical and the horizontal instructions: one could argue that it is doubtful that the pilots have two totally separate plans in memory, that no correlations are made between the two. We have some excuses for such a criticism, in that the programs have no knowledge of the spatial relations between the different points of the flightplan (except for their order), and of the capabilities of the aircraft; this has two consequences:

- first, some actions belonging to a single domain are difficult to process: the best example is the effect of an "intercept" schema. The planned sequence will differ according to the position of the aircraft (will the aircraft intercept the radial if it keeps its present heading, or is a turn necessary before ?). Since the program does not know the precise position of the aircraft relative to the VOR, it cannot decide between the two.

- second, this makes it impossible to link the two plans: the programs cannot infer the effect of an instruction concerning one dimension upon the other one.

This spatial ignorance is of course a poor "excuse", and, if correlations are made by the pilots, they should appear in intelligent planning programs.

Another point, which we will illustrate: in some cases the output of Planho is incomplete, in the sense that the limit of the last leg of the planned sequence is not the runway. This occurs when the controller has instructed the pilot to take some

action, but has not given a complete vectoring to the runway: the pilot is then expecting some further information. It is of course quite obvious that the pilot will not passively wait several hours until something happens. First, after "some" delay, and in the absence of any message, instructions will be asked. Second, it is likely that the pilot has some default plan of action, and it may even be that the controller expects the pilot to apply this plan, defined either by the application of the formally specified default maneuvers (the "lost communications" procedures) or by... common sense. The problem is that we have no idea about this common sense knowledge. Its study seems therefore necessary.

The planning programs have proven that a simple set of programs can be effective in "translating" a technical language into sequences of actions. However, further studies of the pilot's planning activity would be necessary in order to improve the accuracy of the outputs of the programs.

What have the planning programs taught us ? The more important result is that the schemata mean more than we expected. In a way, this is partly contradictory with what we said earlier. In the discussion of the "understanding" programs, we assumed that there was some simplicity in the dictionaries, no polysemy. We assumed further that the schemata could be described by a simple set of elements. We must realize now that this apparent simplicity in surface corresponded to a real complexity in meaning. The data structure of each schema may be simple, but the processes

attached to it are complex. They include a lot of procedures, which react in different ways to different contexts, need to make inferences about missing elements, to use default values, and to build some implicit actions when necessary.

A last remark: it is very easy to speak of the "cross" schema, the "depart" schema, the "intercept" schema, and so on. Why is that ? The first reason is that there is little synonymy, so that a schema can be called by the word that evokes it (this, however, is not always true: some schemata are activated by a variety of words). The second reason is that each schema of Dicolisp (evoked by these schema-associated words) finds in the planning programs its matching procedure. Quite obviously, a "cross" schema cannot be processed in the same way as a "climb/descent" schema: specific procedures are required, both when processing the actual message ("understanding" programs) and when planning the instantiated schemata (planning programs). In fact, in many ways, the understanding programs and the planning programs share the same structure: in both sets of programs, a general processor and specialized subprograms can be found; in both sets, the general processor is independent of the type of schema (for example, the pattern matcher of Schematch applies to any schema), whereas the specialized subprograms are schema-associated (and in fact are named after the schema they are associated with).

CONCLUSION

THE NEED FOR MORE DATA

One of the main problem of the system is that the sample of communications on which it has been built is very small. The analysis of more transcripts is necessary, at least for three reasons:

- As has already been pointed out, the system does not understand all of what is said even in the limited sample we studied. Why? In some cases, because of the use of complex or rare messages by the speaker; but also (and this is more important) because, in order to define the schema for a given category of messages, we need to have a sufficient number of messages of this category: it is only in this way that the words used more frequently can be spotted, and that the elements of the schema can be defined. The problem is that the sample is not large enough to allow some categories to be sufficiently exemplified. We are aware that, for the least frequent categories, restricted vocabularies are probably difficult to build (because the conventionality of the surface form of a message is a function of its frequency of use), but some categories that are absent in our sample are probably not rare (at least from a controller's point of view). Some work needs therefore to be done in that area. A selective sampling of ATC communications may be possible.

- New questions can appear when studying a larger sample. For example, because of a lack of data, there is no schema for

the "clearance delivery" messages. In these messages, successive waypoints and airways are given by the controller. Their number is variable, and depends on the specific destination of the flight. A schema dealing with clearance deliveries would have to allow an indefinite number of elements to appear, i.e. would have to be recursive, at least partly. None of the schemata presently implemented includes such a possibility. Other improvements of the system might be necessary, concerning in particular syntax. Some parsing tools may become helpful, for instance in the identification of the conditions of action. Although these conditions are successfully processed (in the limited sample of messages we studied), some syntactical knowledge could prove to be necessary in some cases.

- Finally, a larger sample is needed in order to validate the approach that has been chosen and to analyze its limits and weaknesses. One aspect of the validation is the study of the performance of the system in terms of percentage of understood messages, but of greater importance is the analysis of what is not understood. It is only this analysis that can really provide an evaluation of the approach we followed. Important questions in that respect are: What makes a message impossible to understand by the system? Is it a matter of rarity, of syntax, of vocabulary? When and why do such messages appear? Are they necessary for the controller and the pilot, what is their function? Are some messages misunderstood? How many?

THE NEED FOR MORE KNOWLEDGE

Some of the problems that are met do not have their origin in the system's limited knowledge , but rather in our limited knowledge.

For example, we might experience some difficulties in "translating" a given input into a sequence of actions, because we do not really understand what was meant by the speaker (the controller). Of course, if we do not understand, the programs will not understand (unless we write some instructions based on "common sense", or on our own representation of the meaning of the message). In other words, we need a better knowledge of the meanings of some words (or messages) for both controllers and pilots and in different contexts, as well as a better understanding of the effect of these words/messages on the planning activity of the pilots.

An important issue is the mental "format" of the planned sequence, i.e. the mental image, built by the pilot, of the successive steps or legs of the aircraft. It has been explained that some assumptions have been made concerning the structure of the representations. The characteristics of the actual representations are fundamental not only in order to give an accurate image of the pilot's activity but also in the design of adapted machine representations of the flightplan, for example in the future on-board flight management systems: the impact of such studies is not confined to improvements of our programs, although these

would indeed profit from it.

One important question in that respect is the relations that exist between the vertical and the horizontal planning activities, i.e. how does an instruction in one domain modify the planned actions of the other domain ?

THE FLIGHTPLAN AS A HIERARCHY OF REPRESENTATIONS

Finally, there is another way to improve the system. The flightplan of an aircraft must not be considered as a linear sequence of events: it can be divided into different phases, and decomposed hierarchically. A flightplan is a plan of action to reach an overall goal. This plan of action is composed of several scripts, each script using several schemata. An example will illustrate this point. For an aircraft flying from San Francisco to Los Angeles, the goal can be described as:

GOAL: (From: San Francisco)(To: Los Angeles)

This goal can be attained using a plan specifying the different necessary scripts:

PLAN: (Taxiing (From-gate: 15)(To-runway: 01R))
(Taking-off on: 01R)
(Departure procedure: Porte 5)
(Routing: Avenal transition)
(Approach procedure: Fillmore 8)
(Landing on: 24L)
(Taxiing (From-runway: 24L)(To-gate: 23)

All the slots of the above scripts are filled, but some of them may be blank at the time of departure; default values may

exist here also. Each script can be expanded in its elementary schemata. Here is for example the expanded script of the departure procedure, Porte 5:

```
SCRIPT:

((Trigger: end of runway 01R San-Francisco)
 (Dir: (H-010 San-Francisco))
 (Limit: ((H-010 San-Francisco)(R-350 San-Francisco))))

((Trigger: ((H-010 San-Francisco)(R-350 San-Francisco)))
 (Dir: (R-350 San Francisco))
 (Limit: (4-DME-fix)))

((Trigger: (4-DME-fix))
 (Dir: (H-200 4-DME-fix))
 (Limit: ((H-200 4-DME-fix)( R-135 Point-Reyes))))

((Trigger: ((H-200 4-DME-fix)( R-135 Point-Reyes)))
 (Dir: (R-135 Point-Reyes))
 (Limit: (Pesca)))

((Trigger: (Pesca))
 (Dir: (H-090 Pesca))
 (Limit: ((H-090 Pesca)(R-116 Woodside))))

((Trigger: ((H-090 Pesca)(R-116 Woodside)))
 (Dir: (R-116 Woodside))
 (Limit: (Wages)))

((Trigger: (Wages))
 (Dir: (R-116 Woodside))
 (Limit: (Avenal)))
```

Some evidence of the relevance of such an approach can be found particularly in the "clearance delivery" communications, which we have begun to study. Here is an example:

```
"(1) United one six one,
 (2) cleared to Los Angeles,
 (3) fly a Porte five departure,
 (4) Avenal transition
 (5) as filed,
     [...]"
```

In this example, message 2 sets the general goal, and messages 3, 4 and 5 specify some scripts.

Such an approach (similar to the one proposed by Hammer, 1983; cf also Rouse et al, 1983) can provide useful tools for a better understanding of Air Traffic Control messages.

First, each script dictates the type of message that can, or cannot, appear: when taxiing, a pilot does not expect to be given a level change, for example, but is prepared to be told to "hold short of runway". These expectations help us understand how the pilots manage to make sense out of the garbled gibberish they sometimes hear. In the same way, these expectations can guide the comprehension of the messages in a language understanding system. (However, it must be understood that this can be the source of some misunderstandings, in real work as well as in a system, when the messages happen to differ from the expectations.)

Second, the effect of the controllers' instructions vary according to the level (goal, plan, script, schema) to which they apply. The modification of a flightplan may affect a single script of the plan, and have no consequence on the next or preceding script. For example, modifications of the horizontal trajectory (shortcuts) during the departure procedure will have little effect on the other scripts. On the other hand, other modifications may affect the goal itself: if the destination airport is closed because of snow, a new goal has to be set, which in turn affects the plan, the scripts and the schemata.

REFERENCES

ABRAHAMSON A.A., 1975. Experimental analysis of the semantics of movement. In : Norman & Rumelhart,1975.

ALBA J.W. & HASHER L., 1983. Is memory schematic? Psychological Bulletin, 93, 2, 203-231.

ANDERSON J.R. & BOWER G.H., 1973. Human associative memory. Washington, Winston.

CHAPANIS A., 1965. Words, words, words. Human Factors, 7, 1-17.

CHAPANIS A., 1978. Interactive communication: a few research answers for a technological explosion. 86th annual convention of the American Psychological Association. Toronto, Ontario, Canada, Aug. 1978.

CLARK H.H. & LUCY P., 1975. Understanding what is meant from what is said: a study in conversationnally conveyed requests. J. Verbal Learning and Verbal Behavior, 14, 56-72.

DUPRE J., 1981. Natural kinds and biological taxa. The Philosophical Review, XC, 1, January 1981.

EHRENREICH S.L., 1981. Query languages: design recommendations derived from the Human Factors literature. Human Factors, 23(6), 709-725.

FAA, 1982. Airman's Information Manual. Basic flight information and ATC procedures. US Department of Transportation. December 23, 1982.

FALZON P., 1982. Les communications verbales en situation de travail: Analyse des restrictions du langage naturel. Report CO 8211 R70, INRIA, Le Chesnay, France.

GIBBS R.W., 1979. Contextual effects in understanding indirect requests. Discourse processes, 2, 1-10.

GIBBS R.W., 1981. Your wish is my command: convention and context in interpreting indirect requests. J. of Verbal Learning and Verbal Behavior, 20, 431-444.

HAMMER J.M., 1983. A computerized, procedural aid for flight crews. Proc. of the 2nd Symposium on Aviation Psychology, Columbus, OH, April 26-27, 1983.

HAMMOND N., BARNARD P., CLARK I., MORTON J. & LONG J., 1980. Structure and content in interactive dialogue. Report HF034, IBM Hursley Human Factors Laboratory, Winchester, UK.

HILL I.D., 1972. Wouldn't it be nice if we could write computer programs in ordinary English - or would it? The Computer Bulletin, 16, 306-312.

HUNTER J.S., BLUMENFELD D.E. & HSU D-A., 1974. Modeling air traffic performance measures. Volume 1 : Message element analyses and dictionaries. Report FAA-RD-73-147,I. US Dept of Transportation. Washington D.C.

HUPET M. & COSTERMANS J., 1982. Et que ferons-nous du contexte pragmatique de l'enonciation? Bulletin de Psychologie, 35, 11, 759-766.

JANET E., 1981. Analyse du langage employe dans les communications d'un systeme de controle. Report CO 8201 R 69, INRIA, Le Chesnay, France.

KELLY M.J. & CHAPANIS A., 1977. Limited vocabulary natural language dialogue. Int. J. Man-Machine Studies, 9, 479-501.

MICHAELIS P.R., CHAPANIS A., WEEKS G.D. & KELLY M.J., 1977. Word usage in interactive dialog with restricted and unrestricted vocabularies. IEEE Transactions on Professionnal Communication, Vol. PC-20, 4, 214-221.

MINSKY M., 1974. A framework for representing knowledge. MIT AI Lab Memo 306. Cambridge, Mass.

NORMAN D.A. & RUMELHART D.E., 1975. Explorations in cognition. San Francisco, Freeman & Co.

ROSENBERG J.K., 1982. A psycholinguistic model for command names. Report CIS-20, XEROX-Palo Alto Research Center.

ROUSE W.B., HAMMER J.M., BROWN E.N. & MORRIS N.M., 1983. Pilot interaction with automated airborne decision making systems. Third semiannual progress report. Sept. 1982/Feb. 1983. Georgia Institute of Technology, Atlanta, Georgia.

RUMELHART D.E., 1978. Schemata: the building blocks of cognition. Report 79, Center for Human Information processing, University of California, San Diego.

RUMELHART D.E. & NORMAN D.A., 1975. The active structural network. in: Norman and Rumelhart, 1975.

SCAPIN D. 1981. Computer commands in restricted natural language: some aspects of memory and experience. Human Factors, 23, pp. 365-375.

SCAPIN D. 1982. Computer commands labelled by users versus imposed commands and the effect of structuring rules on recall. Proceedings of the Conference on Human Factors in Computer

Systems, Gaithersburg, Maryland.

SCHANK R.C., 1975. Conceptual information processing. Amsterdam, North-Holland Publ.

SCHANK R.C. & ABELSON R.P., 1977. Scripts, plans, goals and understanding. Lawrence Erlbaum Associates. Hillsdale, N-J.

SPERANDIO J.-C., 1969. Analyse des communications air-sol en controle d'approche. Report IRIA CO 6909 R21, IRIA, Le Chesnay, France.

THOMAS J.C., 1976. A method for studying natural language dialogue. IBM Research Report RC 5882, Yorktown Heights, N-Y.

THOMAS J.C., 1978. A design-interpretation analysis of natural English with applications to man-computer interaction. Int. J. Man-Machine Studies, 10, 651-668.

WINOGRAD T., 1972. Understanding natural language. Cognitive Psychology, 3, 1-191.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique