Computer Assisted Language Learning

Program Structure and Principles

Edited by
Keith Cameron
University of Exeter

intellect books
BSP Blackwell Scientific Publications Ltd
Osney Mead, Oxford OX2 0EL

VII

TOWARDS AN INTELLIGENT SYNTAX CHECKER

J.E.Galletly and C.W.Butcher, with J.Lim How

University of Buckingham

This chapter contains two principal parts : the first aims to present an extremely wide overview of the directions in which we feel Computer Assisted Language Learning should perhaps be moving in the future; and the second, to report on a small project in this field carried out at Buckingham, using PROLOG on an Orion super/minicomputer, and designed to check a small area of French syntax. While it would be temerarious to claim that this project, of a very limited scope, is any sort of real pointer to the future, we do feel that certain of its unconventional aspects may indicate a possibility for new lines of research.

1. The Next Generation: What Future for CALL?

Taking the widest of overviews, it is possible to argue that the processing of natural language by computers, and, with it, CALL, is at present at a crossroads. It is our feeling that the various initiatives within the subject, and the various constraints without, whether in hardware, finance, or public expectations, have reached a 'cusp point'. It will, in our view, either all tend to run out of steam or else begin finally to make a number of major breakthroughs.

On many levels, CALL may be considered to have existed long enough now to have had the chance to acquire a clear *modus vivendi*. Whatever the vicissitudes of funding at the moment, many UK universities have sufficient numbers of semi-dedicated machines, in most cases Acorn BBCs, for normal-sized teaching groups to gain individual hands-on experience. Within the particular field of French, there are several score programs available for use on these machines. A very broad categorisation of them might consist of saying that one major area is demonstration and testing of simple grammatical points within the framework of multiple choice or correct/incorrect question/answer sessions. The other main area is more or less based on games: for instance, anagrams, cloze type exercises, or adventure type situations, where, if the situation itself may be relatively open-ended, the language

elements themselves are again comparatively limited.

Programs for language teaching are not, however, limited to language teaching programs. Essay writing may be assisted by use of word processing, with or without spell-checkers, and this often leads to considerable gains in both accuracy and creativity. The teaching of translation, in establishments where this is considered a constructive activity, is greatly enhanced when compared with the model, or rather counter-model, of machine translation. More generally, any activity at all on a computer, whether or not specifically designed for teaching and/or language purposes, may well contribute to language use: for instance constrained or open-ended communication with Minitel services or other machine users; or indeed any activity whatsoever with computers which provides a pretext for discussion in the foreign language concerned.

The potential benefits of all these methods are indisputable. The main one, from the all important view of the student her/himself, is that the (micro-)computer normally provides immediate, individual, uncritical, and unambiguous feedback about some aspect of language performance. Whereas human views on language are often ill informed, evasive, contradictory, or even wrong, the mere fact of being informed by the machine leads the student to believe that error and obscurity are minimised, if only because of the process of formalisation, and s/he is often right.

Nevertheless, we believe that many of the existing initiatives may well prove difficult to sustain. One of the problems is that of the commercial world outside. The educational market represents perhaps 1% of the total national market for hardware and software, and educational software hardly crosses national boundaries at all. The result, then, is that the business world, with which students will increasingly be making comparisons, is apparently in a better position than educational establishments to produce sophisticated and well presented products within a minimal lapse of time. Another problem, in the UK at least, is that of standards. Until now the *ipso facto* standard provided by the BBC machines, at least in language departments, has proved an inestimable advantage for communication, despite the limited memory capacity of these venerable devices. In our view, the future will however be marked by a period of competition between even the Archimedes, with its capacity to operate on PC-DOS, and pure IBM compatible micros.

But the final problem is that of the very methodology, and this, we believe, is where the next few years may well prove crucial. It would seem probable that the degree of complexity of language 'processed' by computers will increase markedly. The evidence from other areas of almost quantum leaps is here indicative. After

draughts, where a computer was of world champion level as early as 1959 (1), microcomputers have, after many false starts, reached average club level at chess, can prove theorems in geometry, can do questions from IQ tests. In other words, some element of intelligence has convincingly been demonstrated, often even on the humble micro, and the severest critics have thus been forced to repeatedly reduce the area where 'a mere machine will never be as good as a human being'. Again, from a slightly different angle, expert systems, representing the transcription of human expertise in such subjects as medicine or share dealing, demonstrate behaviour comparable in some respects to that of humans. This remains true even if the methods employed are often the severest of short cuts, with the inevitable consequences of limited areas of competence and of lack of flexibility.

The implications for language are inescapable. Despite the elusiveness of many aspects of the subject, the amount of non-trivial processing of natural language will increase. At the same time, the commercial influence, if only on operating systems or programming languages, will become more and more important. In this perspective, it is impossible to overestimate the importance of word processing. Of course, the computing implementation of present day achievements cannot be considered especially difficult (and one can therefore legitimately ask why, like the walkman, they took so long to be introduced in practical form). But this lack of computing complexity, although it has led many 'pure' computer specialists to dismiss the whole area, probably has little to do with its real potential, which would seem very large indeed. It is our view then that, despite the extra impetus provided by 'desk top publishing', the full effect on many practical areas of even present stages of text processing is still to be felt. Sir Alan Peacock, for instance, has emphasised the extent to which the work of government committees is beginning to be transformed (2); and some language teachers, to bring the subject closer to home, are just beginning to assess the practical and theoretical consequences of this mini-revolution.

As one example, should spelling be taught at all in cases where much of the donkey work can be done by machines? Again, translations and essays, etc., are to be carried out by the student without any external help, goes the unwritten rule, but does this apply to help from a mere computer? The question is especially crucial in those universities where traditional, 3-hour examinations are *not* the only method of evaluation, where, as a consequence, a rich enough student may improve 'take home' work by artificial means. But the problem is not very far away from the examination hall either. Anyone who participated in the incoherent and anguished

(1) See Butcher, H.J., (1968), *Human Intelligence: Its Nature and Assessment*, p.133.
(2) In an address at the University of Buckingham, "The Future of Broadcasting", May 1987.

debate about the use of calculators in mathematics and science will understand that the problem of the use of portable language processors is urgent, and should be discussed without delay.

Such, then, was one element of our thinking about a year ago. The huge advantage of word processors and spell-checkers is that they represent real interaction between the user and the computer. Their disadvantage, of course, is that, ultimately, they represent the mere mechanical storing and reproduction of minimal units of language. The word processor itself is even language free (give or take a few diacritics), a fact which demonstrates its conceptual emptiness; and the spell-checker is, in its present avatars, nothing but a word list. The task for the future is thus that of enhancing the substantive but excessively discrete areas of natural language that computers can already cope with. One's awareness, however, of over ambitious projects in all areas of computing, and the often even less justified claims accompanying them, must incite one to a great deal of caution in predicting what can be achieved.

Our next conclusion, therefore, was that the syntax/semantics distinction might prove vital. On the one hand, semantics, with its strong links with philosophy, is a highly contentious area, and contains very few indisputable assertions indeed. The syntax of a given language, in marked contrast, represents a considerable body of accumulated knowledge, in relatively uncontroversial form. Descriptive linguists, who have often replaced the prescriptive ones in recent years, even have an ultimate court of appeal as to the 'correctness' (i.e. existence) of a given 'string' of characters: either submission to competent users of the language in question, or comparison with pre-existing performance in that language. The set of all possible utterances in a language, in other words, is a well defined set; and so is that of utterances which do *not* conform to the language. Ultimately, it may perhaps follow that the distinction between the two sets may be susceptible to rule based treatment; and therefore to treatment by machine based methods. At the same time, syntax is obviously sufficiently broad and deep to present any number of real challenges for the future, in both applied linguistics in general and its subvariety based on computers.

As far as CALL in particular is concerned, studying syntax could thus be a reasonably precise area of research, while at the same time having the interest and prestige of being a subject 'on the cutting edge of human knowledge'. But in fact, at least as important an advantage is that emphasis on 'mere' syntactic processes is of course the substance of much foreign language teaching practice, even where advanced students are concerned. Perhaps as little as half of the feedback process is concerned with what the students 'really' wished to say, or, especially, write; and

perhaps as much as half with 'mistakes' on the 'surface' level of spelling, grammar, etc. Many of these errors are in fact on a surprisingly elementary level (3).

Our next piece of heart-searching took us into the more technical area of considering the choice of tools available.

2. Choice of Programming Language

Another reason why CALL and, more generally, artificial intelligence applied to languages may be considered at a crossroads is the use of programming language.

BASIC is of course at present the lingua franca in many areas of both CAL and CALL in the United Kingdom. The principal reason is accessibility: the language itself is relatively easy to learn, and easy to use; and it is often included with the micro-computer on sale. Amongst the many dialects, Acorn's BBC-BASIC is universally recognised as being second to none, to such an extent as to have been adopted by at least one notorious arch-rival.

It was of course inevitable that computer purists, or puritans, should decree that more necessarily meant worse, that making the arcane knowledge of the boffins available to the masses was necessarily to adulterate it. The language community, on the other hand, took the eminently sensible view that its interests did not always coincide with those of other users, a view often encapsulated in disdain of 'mere number crunching'. Whatever the underlying reasons, BASIC has in fact proved of inestimable worth to linguists as the standard language, and one can point to such highly creditable achievements within it as Kenney and Kenney's *A Vous la France!* (1986) or Farrington's *Littré* (1987) (4).

The disadvantages of BASIC have also been well rehearsed: in particular, its unstructured nature which, even in the BBC dialect and in not untutored hands, can sometimes lead to unwieldy programs which are difficult to read, and therefore

(3) Hares,R. and Elliott, G., (1982), *Compo! French Language Essay Writing.* Designed for both secondary and university level students, *Compo!* reveals how often, in the view of Hares and Elliott, essays are marred by elementary mistakes: the examples quoted (pp. 28 – 30) would seem to be approximately 40% pure spelling, 40% pure agreement problems, and only 20% cases requiring further explanation. The inescapable implication is that, in an overwhelming majority of cases, this formal aspect is an area where computers are likely to be very shortly encroaching on that of students' competence.

(4) Kenney, M-M. and Kenney, M., (1986), *A Vous la France!*, BBC Publications; Farrington, B, *Littré*, 1987 version, Scottish Computer Based French Learning Project, University of Aberdeen.

also difficult to alter without the whole edifice beginning to crumble over one's head. Other disadvantages can be slowness of reaction time, and, as we have seen, limitations of memory in the machines on which it is normally implemented.

Amongst the alternatives we considered, therefore, was Icon, which is a modern derivative of the string processing language SNOBOL. As such it was clearly suited to language processing work. On the other hand, its use is at present largely limited to the United States. Even if work is presently being carried out to 'port' Icon compilers to popular microcomputers available on the British market, we thought it better to play safe, and avoid excessively eccentric choices.

This left as main contenders, amongst those programming languages used by workers in the fields of Artificial Intelligence and Knowledge Engineering, LISP and PROLOG. Both languages are essentially different from BASIC, in that they are very 'high level' ones. In BASIC, a great deal of effort is expended giving detailed instructions to the computer as to how to go about solving the tasks required. LISP and PROLOG, in marked contrast, are 'declarative' languages: they merely state, in duly standardised form, the nature of the task. In this way, the donkey work of specifying the steps for solving the task is delegated to the compiler. The result is much shorter programs, and, hopefully, more elevated and clear sighted programming.

Both languages are, again, suited to string handling; but here this built-in capability exists on many different levels, in a way that models remarkably certain features of natural language. Thus the main structure in both languages is the 'list': a word may be defined as a list of characters; but then a sentence may be defined as a second level list, a list of words; and so on. This sort of recursive possibility is not, however, limited to such definitions. It may be invoked in general even within the procedures, allowing them notably to invoke themselves. The result, as one can imagine, is highly concise and elegant programs.

An excessive degree of elegance may here, however, be dangerous, in the sense that abuse of recurrence leads to potential difficulties. Nevertheless, it is perhaps not entirely too fanciful to imagine that this very danger is indicative of deep parallels between programming and natural languages, as brilliantly demonstrated by Hofstadter (5). In particular, he claims that a) natural language is intrinsically defined by its capacity to cope with the multi-level contradictions produced when one allows formal systems to self-refer by embodying emblematic representations of themselves and that b) this analogy between natural language and computing

(5) See Hofstadter, D.R., (1979), *Gödel, Escher, Bach: An Eternal Golden Braid*, New York; Hofstadter, D.R., (1985), *Metamagical Themas.*

languages may be very fruitful indeed for future research in such areas.

Choosing between LISP and PROLOG comes down to a number of possibly ancillary factors. It is not our intention to arbitrate the fierce debate currently going on amongst 'pure' computer scientists as to the intrinsic merits of each. But LISP has the advantage of being more widely available, with more researchers proficient in it, and more existing programs. Against that, it suffers, in our view, from a slightly cumbersome syntax, a proliferation of brackets, which makes programs difficult to read and to adapt.

PROLOG, on the other hand, is a more recent language. It was chosen by the Japanese as the base language for their fifth generation computer projects. This is possibly a sign of its inherent worth; but also a knock-on effect may be produced in the future, and PROLOG may thus become one of the standard languages in artificial intelligence.

More particularly, one can point to two particular advantages of PROLOG. First, it has built-in pattern matching routines, clearly invaluable in the context of repeated searches for given patterns of letters within words, and given words within the text as a whole. Secondly, it has intrinsic modularity. It is therefore especially suitable for not only building prototypes of systems quickly, but also, should this seem useful, adding successive new stages to existing systems (6).

Ultimately the choice of language is determined by its ease and pleasantness of use for a given purpose. (Literary trained scholars may therefore be more convinced by appeal instead to Barthes's 'pleasure of the text', and his insistence on *scriptabilité* ('write-ability') and *lisibilité* ('read-ability').) For us, whatever the reason, it was PROLOG, by half a head.

3. A Brief Introduction to PROLOG

[The aim of this section is to give some of the flavour of the PROLOG programming language, by presenting a few concepts and examples. It is not, however, essential to the understanding of the next section, which describes the project itself in essentially practical terms. Some of the details of the PROLOG implementation itself are, in addition, briefly described after the project.]

(6) PROLOG has a very strong compatibility with natural language processing. As just one example, the syntax of some PROLOG compilers has even been extended to enable a particular class of parsers, called 'Definite Clause Grammars', to be written easily (Pereira, C.N., and Warren, D.H.D., (1980) "Definite Clause Grammars for Language Analysis", *Artificial Intelligence*, Vol. 13, p.231.).

Any programming language for AI or expert systems must necessarily have some internal means of representing knowledge. Ideally, a knowledge system will include the following features:

1. a knowledge base, a set of facts and rules;

2. an inference engine, a system to reason with the given facts and rules;

3. an explanation facility, to explain to the user why the system has adopted a particular line of reasoning;

4. user interface, to provide easy-to-use access; and

5. a knowledge acquisition system, a method for acquiring and encoding new knowledge.

In PROLOG, the inference engine is explicitly provided, but great freedom is accorded to the programmer in instituting the others!

The name 'PROLOG' means 'Programming in Logic'. Basically, the programmer's task is to state the problem in terms of defined facts and rules, these rules being expressed as a 'logical' sequence of statements. A PROLOG program, then, comprises a set of known 'facts' (the 'database'), and a set of rules or relations governing the facts, the two together being called the 'knowledge base'. The system solves a problem expressed in terms of a goal by attempting to prove the 'validity' (positive truth value) of this goal on the basis of the given facts and rules. Normally sub-goals will be defined by the system, and then proved separately.

A very simple example may make this much clearer. At a first stage of sophistication, we simply wish to communicate to the system the present indicative conjugation of the verb *avoir*:

```
avoir  (ai).
avoir  (as).
avoir  (a).
avoir  (avons).
avoir  (avez).
avoir  (ont).
```

These, then, are PROLOG facts, with *avoir* being called the 'predicate', and *ai*, *as*, etc., the 'argument'.

If we wish to add further information, then we could write the following PROLOG facts:

```
verb  (avoir, ai).
verb  (avoir, as).
        .
        .
        .
verb  (avoir, ont).
verb  (être, suis).
verb  (être, es).
        .
        .
        .
verb  (être, sont).
```

[Read: 'there exists a verb including parts *avoir* and *ai*',etc.]

Here we have defined a new predicate, called 'verb', and included the infinitive *avoir* or *être* as a second argument to this predicate.

Having given the system a reasonable number of similar facts, like other verb conjugations and tenses, one can then interrogate the system. A question such as

verb (Inf, sommes).

asks the system to find an Inf (infinitive) such that *sommes* is part of the same verb. (The system does not *know*, of course, that Inf means anything: for it, Inf is just an unknown variable. The capital I on Inf, incidentally, marks it as being a variable rather than a constant (which would begin with a small letter).) The pattern matching facility of PROLOG is then invoked, the database is searched, and the solution

Inf = être

duly appears on the screen.

Turning now to an example of the *rules*, let us assume that some regular verb stems and verb endings have already been read in, as follows:

```
reg_stem     (parl).
reg_stem     (port).
                .
                .
                .
reg_stem     (aim).
reg_ending   (e).
reg_ending   (es).
                .
                .
                .
reg_ending   (ent).
```

If we wish to tell the system now that a verb is in fact made up of a stem plus an ending, we simply write the *rule*:

```
reg_verb (Stem, Ending):-  reg_stem (Stem),
                           reg_ending (Ending).
```

[:- is read 'such that', and , is read 'and' (the logical operator).]

A rule, in other words, enables the system to generalise, to cope, in the present example, with *any* regular verb. More generally, a rule is always of the form

Head:- Body.

where Head is what is being defined, and Body is what is already known, being comprised of a predicate or predicates.

The power of PROLOG is of course that this process may be repeated as many times as one wishes, so as to build up knowledge bases of indefinite complexity. But even within the simple database of verb conjugations, one *can* imagine non-trivial problems which could be quickly solved. Assuming that 'all' French conjugations have been read in, one could then ask which verbs have an identical present and *passé simple*. Ask the average human user, and you might receive the response '*dit*'.

Let us assume that the facts have been entered, for all verbs, in the form:


```
                .
                .
                .
verb   (present, dit).
                .
                .
                .
verb   (passé-simple, dit).
                .
                .
                .
```

and that a general rule has been indicated, of the form:

```
find (Tense1, Tense2, Part):-  verb (Tense1, Part),
                               verb (Tense2, Part),
                               Tense1 \= Tense2.
```

(where \= is the inequality operator).

Then a query of the form:

find (present, passé-simple, X).

would elicit the response:

X = dit

But then as many further instances as wished may be obtained by repeatedly typing a semi-colon (;), which will give

X = finit
X = choisit, etc.

Again, to enquire which *different* verbs have an identical part, and assuming that the facts have been entered in the form:

```
.
.
.
verb  (past-subjunctive, crusse, croître).
.
.
.
verb  (past-subjunctive, crusse, croire).
.
.
.
```

together with a rule:

```
find (Inf1, Inf2) :-  verb (Tense, Part, Inf1),
                      verb (Tense, Part, Inf2),
                      Inf1 \= Inf2.
```

Then a query of the form:

find (A, B).

will elicit the response:

B = croire
A = croître

In sum, PROLOG is a highly flexible and elegant language, one which is perfectly adapted, we believe, to processing natural language.

4. The Project

The project was a final year undergraduate Computer Science one undertaken by J. Lim How, and supervised by J. Galletly (School of Sciences) and W. Butcher (School of Humanities).

It was principally a pilot study into the development of tools for computer assisted teaching of French language at the University of Buckingham. Besides students taking French to degree level, Buckingham has numerous students taking French as a supporting course from beginner's to post A-level standard. The project was not intended however to be a *pure* CALL project, for two reasons: 1. students are not especially orientated towards the theory or practice of teaching; and 2. it

was thought that the results of the project would be of more interest if the system exhibited some aspects of the general, open-ended use of language characteristic of human communication, rather than simply leading the user through a predefined and closed teaching situation. In other words, the system should demonstrate some small degree of machine intelligence or expert knowledge. For the reasons explained above, it was considered that the specific area of French syntax was sufficiently wide as to allow a large number of interesting possibilities.

5. General Requirements of a CALL System

We list below some important general features which we believe any CALL system should possess. The list is not exhaustive nor original. Barchan *et al.* (7), for example, have expressed similar views: 1. the system should have some pedagogic value and provide an interesting environment in which to learn; 2. the system should provide quasi-immediate responses, users should not be kept waiting unduly for system responses; 3. the error reporting should be helpful to the user, the messages should be meaningful; 4. the system should correct user errors wherever possible; 5. the system should accept *free* input, the system should be wide ranging enough to cope with arbitary sentences and not confine the user to a narrow range of input; 6. the system should be robust, user errors or unexpected answers should not make the system crash; 7. the system should be capable of expansion, new ideas, new approaches, new areas should be readily accommodated; 8. consequently, it must not be idiosyncratic: it must use methods that are both transparent and reproducible.

In the following sections, we provide a mainly practical and linguistic description of the project.

6. Area of Investigation

The human method of constructing sentences in a foreign language, at least at the elementary and intermediate level, includes applying, implicitly or explicitly, certain rules of grammar. It is this notion which we decided to use: instead of following the traditional approach of parsing, we based the 'syntax checker' on various heuristics about French grammar in certain selected domains. These heuristics or rules form the 'knowledge base' of our system, with rules being applied to a French sentence to see if the sentence conforms to them or not. Our method, then, is slightly reductionist, but no more so than many accounts in textbooks.

Two closely related areas of French syntax which seemed compact enough for

(7) Barchan, J., Woodmansee, B., and Yazdani, M., (1986), "A PROLOG Based Tool For French Grammar Analysis", *Instructional Science*, vol. 14, pp.21 – 48.

this project suggested themselves. These are

negation

and

object pronoun order in verbal phrases.

Both areas are regular enough to allow some sort of systematic treatment and are also sufficiently different from their parallels in English to offer interest to non-native speakers of French. Barchan *et al.* (1986) have pointed out research evidence for the necessity of putting bounds on the learning area, only specific points of grammar should be dealt with.

Negation

Negation has the advantage that the nine main operative words

ne ... pas, point, jamais, rien, plus, personne, nullement, guere

are morphologically invariant, with the exception of *n'* being a variant of *ne*. On the other hand, there are major disadvantages. Although normally *ne* and one of *pas*, *point*, *jamais*, etc., must *both* be present in the sentence, there are certain exceptions. These include *ne* on its own, *pas*, *point*, etc., on their own, and cases involving *ne* and *ni* in combination. There is also the situation where these words are used as nouns or other parts of speech, for more than half of them, *pas*, *point*, *rien*, *plus*, *personne*, are not necessarily negation words at all. In the event, due to time constraints, we adopted the practical expedient of bypassing these problems, and requesting the user not to be so perverse as to introduce such sentences as *un plus n'est plus plus qu'un rien*!

The basic rules implemented in the program are as follows:

1. *ne* can be followed (but not immediately) by any negation word in a sentence. There has to be at least one word (including a verb) in between. For example, *Je n'entends personne* ('I hear nobody') is accepted, but *Je ne personne* is rejected.

2. *pas* and *point* in the same sentence are considered ungrammatical, as is a combination of *pas* or *point* with *jamais*, *rien*, *plus*, *personne*, *nullement* and/or *guere*. But a combination of two or more of this last list *is* allowed, provided that the same negation word does not appear twice in the sentence. e.g. *Je n'ai pas point vu Paul*

is rejected, but *Je n'ai jamais rien vu de pareil* ('I have never seen anything like it') is accepted. On the other hand, as explained above, 'perverse' sentences like *Rien n'est plus beau que rien* are technically correct but are treated as errors.

3. *rien* and *personne* are the only negation words which can precede *ne* but in that case they must do so immediately. e.g. *Rien ne va plus!* ('No more bets please!') is accepted, but *Rien va ne plus* is rejected.

While these few rules are of course far from a complete description of negation in French, they were found in practice to be sufficient to 'trap' many learner errors.

Object Pronoun Order

Verbs and their preceding pronouns, with optional negation, present a complex but well formed structure in French, and one which conveniently supplements the above.

Sequences of up to eight words can be dealt with by the system, which can thus on occasion seem quite impressive. At the same time, seven of the words are from very well defined categories, and there is little possibility of intervening words: two advantages making the implementation much easier.

The various combinations possible are summarised in the following table, which covers all indicative tenses, together with negative imperatives:

ne	me	le	lui	y	en	*VERB*	pas
	te	la	leur				point
	se	les					jamais
	nous						rien
	vous						plus
							personne
							nullement
							guère

Of course, almost any or all of these words could be absent. The only necessary element in the sequence is in fact the verb. Accordingly, our analysis of pronoun order starts by trying to identify the verb in the sentence entered and, only when this has been successfully carried out, examining the pronoun order.

This identification is a major problem. Various lines of attack might have been possible here, including checking words against an existing dictionary, looking at

the context of words in their surroundings and examining the endings of words. The first approach was used by Barchan *et al.*: trailing characters are stripped off a word until a morphological root is recognised in the dictionary. But the word-ending approach looked the most interesting to us: the program would attempt to locate a verb in a sentence by examining the endings of all the words. In the event, we adopted the opposite method to Barchan's, stripping off *leading* characters until a recognisable *ending* appeared. Given that the longest endings were thus searched for first, this had the advantage of identifying *-tes* as distinct from *-es*, as distinct from *-s*.

Another problem is that, in some tenses, French verbs are in two main parts: the auxiliary *avoir/être* plus the past participle. Also other words, such as *même* or *indubitablement* may intervene before the participle. The solution adopted was to consider the finite part of the verb as the operative part and to ignore the participles. This decision is in line with speakers' subjective impressions that the auxiliary is the vital part, and it also obviates the problem of agreement of the past participle (8).

As a first step, some highly simplistic rules for identifying verbs by their endings were identified. We adopted the practical expedient of accepting the affirmative, negative and imperative forms of the verbal phrase, but not interrogatives. (Infinitives may, of course, be present but are in any case ignored by the program, which simply identifies the finite verb.)

The basic French verb endings may be summarised as follows:

1. words with endings *-it*, *-ai*, *-as*, *-ez*, *-ais*, *-ait*, *-ent*, *-est*, *-ons*, *-ont*, *-iens*, *-ient*, are *probably* verbs

and

2. words with endings *-a*, *-e*, *-s*, *-es*, *-is*, *-tes* are *possibly* verbs.

It was decided, however, that these rules were of limited usefulness on their own: many words which are not verbs have endings in *-e*, *-s*, *-es*, etc. Also, the distinction possibly/probably would be very difficult to implement in practical terms. As one way of alleviating the problem, the program was given some more information:

(8) Or past participles, as in the various forms of *surcomposé*, e.g. *il a eu fait*.

1. a small dictionary containing some common non-verbs with the above endings is searched before the verb rules are applied. A successful matching is then ignored as a verb;

2. a dictionary containing the complete conjugation of three of the most common irregular verbs, *avoir*, *être*, *aller*, is also searched before the verb rules are applied. A word matching with a dictionary entry is taken to be a verb.

If the use of these two dictionaries does not work, the situation clearly becomes more problematic. For example, there is the homonym problem: *porte* is possibly a verb, *le porte* certainly is, *la porte* just possibly is, *je la porte* certainly is, and so on.

In the event, we recognised that there is limited knowledge in the system and, due to time constraints, instead of trying to identify the verb via further rules and facts based, for instance, on the immediate grammatical context, resorted to user interaction. Of course appealing to the human user reduces the autonomy of the program. It does, nevertheless, increase the user's involvement, which may be an important consideration in an educational context. In these 'awkward' cases, we have to assume that the user has some minimal knowledge of French syntax, i.e. can identify whether a given word is a verb or not.

At this point, a major methodological problem became apparent. A given sentence must, for our purposes, contain a verb, but it may contain, in fact, any number of different verbs, and thus it is hard to know when to stop looking for them. The solution adopted was to assess each word in the sentence in order, and not to attempt to define a 'main' verb. Some examples may make the different cases clearer:

J'ai déjà donné ('I have already made a contribution')

Nous allons gagner la Coupe ('We are going to win the Cup')

Vivre est souffrir ('To live is to suffer')

La musique adoucit les moeurs ('Music makes for gentler manners')

The first three are accepted as such, since the program recognises *ai*, *allons*, and *est* as definite verbs. In cases like *adoucit*, however, the program announces to the user that the word is possibly a verb, and ask her/him to confirm it. In other words, by means of progressively less elegant and autonomous, but more complete methods,

a verb is always identified. In theory, there are no situations where the machine simply 'gives up'.

The analysis of the pronouns proved considerably easier to implement. Use of PROLOG means that the system can identify with relative facility the two negation words and the various combinations of up to five pronouns. It can then check whether the canonical order is respected. It finally either notifies the user of any errors detected in the order of the words, or confirms that it has not detected any errors of this sort. Thus

N'y va pas ('Don't go there')

Il n'y en avait plus ('There weren't any left') and

Je le lui donnai ('I gave it to him')
are accepted.

Contrariwise,

Je lui le donnai

J'ai lui donne
are not accepted (9).

To sum up, then, what happens in terms of screen presentation: once a prompt mark appears on the screen, the user can enter a French sentence. In certain cases, he will be asked, successively, if certain words are verbs or not. Finally, the machine issues a verdict as to its assessment of grammaticality (covering both the verb(s) and the other appropriate elements of the sentence). It finally produces a prompt, inviting the user to enter another sentence.

7. *Further Details of Implementation*

PROLOG is ideal for expressing the heuristics and dictionaries which form the intelligence of this system.

The dictionaries are written as PROLOG facts, e.g. the individual negation words are written as:

```
negation (ne).
negation (pas).
.
.
.
negation (guère).
```

When a sentence is read in by the program, individual characters are combined to form words and the words are stored in a PROLOG list structure. Each word in the sentence is then inspected in turn using PROLOG's pattern matching facility to access the dictionary until a word is recognised.

1. In the negation part, if a negation word is found, then the predefined negation rules are invoked, to examine each of the remaining words in the sentence to see if the sentence conforms or not.

2. In the object pronoun part, each word in the sentence is checked to see whether it is one of the three irregular verbs. Otherwise leading characters are stripped off the word one at a time and the resultant 'stub' compared with the verb endings in the facts database. If a verb ending is recognised, then the user is prompted that the word is either probably or possibly a verb. Once a verb has been asserted, then the object pronoun rules are invoked to analyse the preceding words so as to check that any object pronouns before the verb are both correctly formed and correctly placed. Finally, either correction or congratulation messages are shown on screen.

8. *Conclusion*

This final year student project posed real and interesting problems; it also generated a great deal of cooperation between the departments involved, and even produced interest from other members of the University.

It was deliberately pitched at a relatively high level, since marketing the result was not an aim and it was felt that the project might as well therefore tackle some substantive area of French grammar. As such, it clearly required a highly heuristic approach, one that may even seem to some people non-conventional, in contrast with, for instance, approaches based on parsing. Also, some of the obstacles encountered could not entirely be removed within the time available, but had to be

(9) It may be noted that the system does not pronounce on whether a given combination makes sense. But this, in our view, is perfectly sensible. Cases like *Je le lui en donne* or even *Je ne le lui y en ai pas donné* are at least perverse. The reason why native or other competent speakers hesitate is doubt as to what nouns all the pronouns could refer to. The problem is ultimately therefore on the semantic rather than the syntactic level, and would thus seem counter productive for treatment by computer based methods at the present moment.

detoured around. Nevertheless, the fundamental aim was certainly met: that of constructing a program which could accept a very wide range of input, and could analyse it in terms of certain well defined grammatical constraints. Without resorting to the brute force method of storing a large bank of predefined questions and answers, a 'semi-intelligent' response is effectively obtained. More precisely, the program provides the user with very quick, appropriate, and reasonably accurate information in an interactive fashion, and this must surely be considered an achievement in the notoriously slippery world of natural language.

This is not to say that the project does not have room for further improvement and extension. It would be very useful, obviously, if sentences containing more than one negative structure could be read in, like *je ne marche plus et je ne cours jamais*. More generally, it is conceivable to use fuzzy logic to deal with cases of 'possibly/probably a verb' in a less cut-and-dried fashion. This would have the additional advantage of being closer to what humans actually do when presented with an ambiguous structure like *sanctionner* or *je suis*: they seem normally to suspend final judgement, and seek further information in the subsequent words, before 'backtracking' to the source of ambiguity.

In fact, certain cases of lexical ambiguity, if on a simple level, may be a fruitful area for further computer based work. Cases quoted earlier, like *porte* vs *le porte*, etc., are extremely context based; and, even for human users, often in practice pass through a stage of hesitation, involving something like fuzzy logic. It would, nevertheless, be relatively easy for a given pair of homonyms, such as *porte-porte*, *pas-pas* or even *manoeuvre* (m.) – *manoeuvre* (f.), to undergo a process of 'disambiguisation' by means of certain key context pointers. Indeed, this is the most obvious failing, and perhaps the easiest remedied, of the current generation of spell-checkers: their limitation to single word analysis. The word itself for this, 'syntax checker', has unfortunately been trivialised by American programs that do little more than check for odd brackets or typing errors like *the the*. Perhaps the next stage forward, for both CALL and the business world, is programs carrying out semi-intelligent analyses of language: real syntax checkers dealing with real linguistic problems.

We will be the first to buy!