Parsing with Flexibility, Dynamic Strategies, and Idioms in Mind

Oliviero Stock

Istituto per la Ricerca Scientifica e Tecnologica, 38050 Povo, Trento, Italy

One desirable aspect of a syntactic parser is being meaningful (i.e., contributing to incremental interpretation) during the process of parsing and not only at the end of it. This becomes even more important when dealing with flexible word order languages, where the number of alternatives in parsing may grow dangerously. One such parser is WEDNESDAY 2. It is a lexicon-based parser, relying on the chart mechanism combined with a particular kind of unification, guided by the so-called Principle of the Good Clerk. The paradigm is a multiprocessing one and the experimentation with dynamic syntactic strategies (what heuristics may sensibly guide the process, for instance within a particular sublanguage) is a relevant task here. An interactive integrated environment built around the parser is described. Flexible idiom processing is one of the best features of WEDNESDAY 2. Idioms are treated as decomposable parts of speech and the treatment of idiom recognition does not differ from the "literal" process until a threshold of activation of a flexible pattern is crossed. At that point a new, idiomatic process is added.

1 Introduction

While natural language parsing has come to be a very serious and respectable field (almost as much as the theory of compilers, according to Martin Kay), a lot of new realizations that appear as variations of consolidated approaches continue to emerge in an unordered way. Although all this causes a lot of redundance and noise, we believe that it is important to maintain this experimental attitude, provided that it is complemented by a correct acknowledgment of the state of the art. In our own work we have followed a set of ideas in the belief that, at this stage, it was better to see them realized in a "closed" world (one in which we do not adopt an external formalism, but rather one in which things are conceived and realized autonomously) that still could lead to a convergence with other more widely accepted currents of lexicon privileging parsing, such as **lexical-functional grammar** (**LFG**).

The best machines that actually parse NL (human beings) are able to dynamically disambiguate a sentence and to make sense of partial parsings, and they take advantage of this ability in their normal activities. Machines that do not do that: a) are bound to have problems in communicating with the machines cited above; and b) will give limited support to our understanding of those machines. While these statements may be readily endorsed by all those working with an "integrated" or "semantics driven" approach, things are less clear when we consider research in which the emphasis is on the role of a strong syntactic component that, per se, may lead only to a shallow semantic representation. Although we belong to this latter subcommunity, we nonetheless believe that parsing as such must deal with the above statements.

In recent years work on a number of languages other than English, the renewed interest in nontransformational grammars, and the taking into account of the experience in parsing have influenced the appearance of new linguistic theories, such as lexical-functional grammar (Kaplan and Bresnan 1982), generalized phrase structure grammar (**GPSG**) (Gazdar and Pullum 1982), **functional unification grammar** (**FUG**) (Kay 1979), **definite clause grammar** (**DCG**) (Pereira and Warren 1980), and **tree adjoining grammar** (**TAG**) (Joshi and Levy 1982).

The appearance of these theories has likewise meant a strong interest in the development and refinement of some techniques that are well suited to processing data expressed within the abstract formalisms characterizing all these approaches, namely feature structures. In particular, unification has become the basic idea (Shieber 1986), even if with a large number of variations on the formal character (for instance, one may compare Prolog-based unification for DCG, specialized unification for particular data types in FUG, equation resolution for LFG), and on overall strategies.

0362-613X/89/01001-18$03.00

Often unification is combined with tabular methods derived from techniques used for parsing programming languages. These methods are naturally suited for parsing nondeterministically context-free languages, allowing for the drastic reduction of the complexity of a nondeterministic algorithm to acceptable polynomial figures (Aho and Ullman 1972). Given the present tendency to emphasize the reducibility of natural language to tractable formal languages (see, for instance, Joshi and Levy 1982), and the high level of ambiguity typical of natural language, this is a fairly natural choice. The most relevant of these techniques for natural language is chart parsing (Kay 1980, Kaplan 1973). See also another interesting and efficient technique in Tomita (1985). Shieber et al. (1983) and Karttunen (1986) introduce a neutral tool, usable with a number of different formalisms all based on chart and unification. It is worth noting that while in some of the referred works there is an explicit interest in the human dynamics of parsing, for most of them the effort is devoted to building clean and efficient mechanisms that are able to associate syntactic descriptions or mappings of structures to input sentences.

Our parser was developed with two purposes in mind: 1) understanding more fully the nature of language understanding and trying out our ideas concerning it; and 2) having an adequate tool for processing the Italian language. With regard to point 1, our view of syntax is twofold, namely as:

a. a set of specifications and constraints that, in parsing, guide the search for, and the linking together of, pieces of semantic representation, at the same time delivering an overall functional description. This is expressed basically in the lexicon.

b. imposing restrictions on the spaces of search for pieces to be linked together, possibly taking into account general criteria of linear precedence.

As far as point 2 is concerned, Italian is a freer word order language than English (e.g., subject-verb-object is only the most likely order in simple declarative sentences—the other five permutations of subject verb and object may occur as well, even in written Italian) and it has a richer morphology. Therefore the role of syntax specified in (b) above is somewhat reduced, while constraints and specifications due to the lexicon, of the kind specified in (a) above are richer. Of course, also, morphological analysis is an inescapable aspect that must be combined with syntax. In particular, in Italian, you can find words like *rifacendogliene*, which stands for "while making some (of them) for him again."

The lexicon in our system includes a large quantity of information, represented in a particular form of feature structure, one in which alternative subcategorizations are presented in a compact way. There is a simple centralized component that deals with distributional aspects of language. In our approach we can state all the variations of word ordering: from obligatory positions of a constituent in the string, to an obligatory position in relation to another constituent, to a preferred position, all the way down to simply admittable positions.

Parsing is the process of building an internal representation of the sentence, while disambiguating in local conditions of uncertainty. In this sense we follow a non-deterministic approach that results in a particular realization of the idea of chart parsing, such that the process goes bottom-up but with strict top-down confirmation. The concept of "sleeping edge" guarantees that all needed edges are introduced, whatever the order of the operations, but also that not too many superfluous edges are introduced. Chart parsing is combined with dynamic unification. The main idea is that unification is not carried on in a second phase, after chart parsing has yielded a constituent structure (as in the basic LFG implementation), but also that it must not be flatly incremental (as in PROLOG-based systems). Our strategy, embodied in the so-called **Principle of the Good Clerk**, is that unification is (asymmetrically) carried out after the main element of the constituent has been analyzed. This does not mean that unification here does not maintain its highly desirable qualities, such as order independence and monotonicity. This is true also here, only that the application of unification is not continuous; our strategy seems reasonable and advantageous especially with languages in which considerable freedom is allowed in ordering the constituents. We would like to emphasize the fact that while the fundamental role of a particular word class for a given constituent (often called the "head") is a well-established concept in many linguistic theories, a similarly privileged role within the process of parsing is not common. The parser is also designed for online interaction with a semantic context and therefore any particular data structure that it builds is a shallow semantic representation, i.e., a logical form of the analyzed portions of the sentence.

This approach is also a good starting point for recognizing idioms. Most parsers are either devoted to idiom recognition, do not treat idioms at all, or else have a very special separate device for that purpose. In the present work instead, we propose a view of the idiom recognition process as completely integrated with the parser, so that, in particular, idiom recognition can take advantage of the flexibility of the approach without redundancy. We think this is important, because idioms are on a continuum with literal language and share the "normal" characteristics, including the flexibility, of the particular language involved. The treatment of idiom recognition does not differ from the literal process until a threshold of activation of a flexible pattern is crossed. At that point a new idiomatic process is added.

Another salient point is that within our approach, given the fact that the algorithm is of a multiprocessing type, one can consider what strategies may be introduced in order to select dynamically the tasks that have better chances to lead to the (preferable) final analysis. These heuristics can also rely on measures of likelihood associated with each partial analysis. An integrated

interactive environment has been built up to experiment with these ideas. Besides aspects present in other environments, e.g., D-PATR and the LFG workbench, our environment provides scope for the cognitive scientist, or the application-minded one dealing with a restricted sublanguage, to explore and define processing behaviour.

In this paper, after an introductory example, we first outline the characteristics of linguistic knowledge for our parser and give an overview of the parser itself. Following that we describe the particular kind of unification used. The way non-determinism and disambiguation work is then discussed. This is followed by a brief description of our treatment of long-distance dependencies. After that comes a description of syntax-based idiom recognition, in our view one of the most relevant features of our work. A description of the environment built around the parser and of experiments carried on in defining dynamic strategies concludes the paper.

1.1 AN INTRODUCTORY EXAMPLE

We shall begin with an informal example to give the flavour of a basic part of our approach. Let us assume that the Italian sentence to be parsed is: *Un itinerario molto noioso ai visitatori ha prescritto la guida,* literally, "A very boring itinerary to the visitors has described the guide." The sentence is topicalized, resulting in a quite common construction. The representations for the first noun phrase (NP) and the prepositional phrase (PP) are built up as parsing proceeds. The building includes both a functional and a semantic representation of the constituents, drawn from information explicited in the lexical entries. At the level of the subject (S) constituent being built, no commitment is taken for what will be the subject, even when the auxiliary *ha* is dealt with. The verb *prescritto* instead causes the merging of information available so far. Therefore some of the functions specified with the verb will find their values (notably the oblique object and the direct object functions). The constituent *un itinerario molto noioso* fails to be chosen as a subject, by virtue of the result of the interaction of the parser with a (separate, and not described in this paper) semantic component that is questioned at the moment of building a new semantic representation. In fact here the meaning of the proposed subject would not be compatible with the agent role for the verb; therefore the only possibility becomes the goal role. Note that interaction with semantics is possible because we have partial representations available at this point. After that, the analysis is guided by the context that has been built so far. There could be two possible constituents whose construction may begin with the word *la* (this word is ambiguous—one reading is an article, the other reading a clitic): an S and an NP. But these bottom-up proposals are checked and, as only an NP is expected, only that interpretation is carried out, and is ultimately unified with the previous representation to produce a complete representation of the sentence, with *la guida* as subject.

With an algorithm that yields constituent structures at an initial stage and performs unification at a later stage, things would have been different. The dynamic interaction with semantics-based disambiguation, when the object is chosen, the selection of the correct interpretation of *la guida* later in the process would not have been straightforward; moreover, positional flexibility could have been achieved only at further cost and, finally, there would have been more redundancy. An approach based on strictly incremental unification would have needed instead either a lot of redundancy at the level of descriptive patterns or a useless tentative commitment to binding the first NP to information carried by the auxiliary.

2 ENCODING LINGUISTIC KNOWLEDGE

We shall now begin describing our parser, called **WEDNESDAY 2** and implemented in Interlisp-D on a Xerox AI-Processor. In the first place we shall give a brief description of how linguistic knowledge is encoded.

WEDNESDAY 2 is based on the assumption that most linguistic knowledge can be conveniently distributed through the lexicon. The lexicon includes in a compact form a lot of information that is normally stored in a grammar. This information includes semantics, syntactic "static" specifications such as category, mood and tense, case marking, the specification of the relation between semantic objects and syntactic objects, such as grammatical functions, but also some more detailed specifications of constraints in binding these objects to other objects that can occur in the sentence. These specifications describe in disjunctive form sets of features that include different aspects: beside the usual ones (syntactic category, generic features, such as gender, number, person, case marking), relative positions of the entities in the string combined with a measure of likelihood of these positions, cospecification of co-occurring phenomena (such as the equi-np in a subordinate clause), and measures of likelihood of the obligatoriness of the actual finding of one such entity to be unified with. A characteristic aspect of lexical representation in WEDNESDAY 2 is that semantic objects are part of the value of linguistic functions.

The only centralized component of the system, in the form of a simple (recursive) transition network, deals with aspects of distribution of constituents. The treatment of rather free word order languages becomes quite natural within this approach, and, in particular, the centralized network then becomes very simple. The important thing to note is that the network does not support operations of structure building (like in an ATN: Woods 1970) nor free annotations, including occurrence of equations (such as in LFG rules). It is the

lexicon that bears feature sets: the network merely has the role of restricting the space where unification can be carried on.

Each entry in the lexicon may include:

1. A semantic representation of the meaning of the word, called **sem-units**. The format of the latter is independent of that of other pieces of information and can be of several different kinds. In our actual implementation we use semantic networks. They can be seen as a set of propositions that use semantic predicates and arguments. Of course, every format has its own operational modality. For instance, unification in a semantic network is carried on as a node merging operation: two or more nodes in one or more net shreds collapse into a single node, producing a new net configuration, that nonetheless subsumes the previous ones. So the basic principle of unification is obeyed.
2. Syntactic data such as:

a. Specification of linguistic (grammatical) functions with indication of objects in the semantic representation as semantic values. In our implementations the objects are nodes of the semantic network. Arbitrary features and a case marking may be specified, in disjunctive form, in relation to the syntactic aspect connected to that object. A relevant aspect in this representation is therefore that grammatical functions are the point of contact between semantics and syntax.

b. A particular linguistic function that indicates a node (in the semantic net) that would be referred to as the **Main** node of the first syntactic constituent that includes the word. (Eventually every constituent, even of higher level, is provided with a Main, in the course of parsing.) It is worth noting that the Main constitutes a concept that bears some similarity with the concept of "head" in some linguistic theories. The most relevant difference, and the one that justifies the use of a different term, is that the value of the Main is an object in the semantic representation, integrated with syntactic specifications.

c. category

d. mood and tense.

Syntactic data here are supplemented by other dynamic syntactic information, kept separate from them for the sake of clarity. The latter consists of sets of constraints on variables but a good metaphor would be to view them as impulses to connect fragments of semantic information, guided by syntactic constraints. Yet, impulses are in a declarative format and are actually interpreted by the parsing algorithm. Impulses specify characteristics of an entity to be identified in a given search space. They have alternatives (for instance, the word "tell" has an impulse to merge its object node with the Main of either an NP or a subordinate clause). An alternative includes: a contextual condition of applicability, a category, features, case marking, side effects (through which, for example, coreference between subject of a subordinate clause and a function of the main clause can be indicated). Impulses may also be directed to a different search space than the normal one. Furthermore, there can be also impulses that do not involve operations on the semantic representation: they set markings or features for linguistic functions or for the Main.

```
(sem-units (n1(p-prescribe n2 n3 n4)))
(likeliradix 0.6)
(cat v)
(verbtense (ind past))
(main n1)
(lingfunctions (subj n2)(obj n3) (a-obj n4))
(uni (subj)
     (must 0.5)
     ((t np 0.9 ( nu sing) nom)))
(uni (obj)
     (must)
     ((t np 0.3 nil acc)
     (t s/sub 0.3 nil)
     (t s/prepinf 0.1 nil di (sideuni a-obj subj))
(uni (a-obj)
     (must 0.8)
     ((t np 0.2 nil a)))
```

Figure 1.

Measures of likelihood can also be specified a. for one alternative to be considered; b. for which relative position the entity sought should be in with respect to the present word; c. for the overall necessity of finding the entity. These measures processed together with other ones will give a quantitative account of the likelihood of an analysis and will dynamically play a heuristic role.

For the sake of example, a lexical entry, corresponding to the word *prescrisse*, a verb in the past tense (encountered in Section 1.1, in the form of a participle), is shown in Figure 1.

Sem-units specify the semantics; here they consist of a single proposition. As we use a semantic network format, the n's are called nodes. Besides the three arguments of the predicate, there is a node, **n1**, that stands for the instantiation of the involved predicate. Next we have the likelihood of the word reading (in our case drawn from frequency figures). Category and verb tense are self-explanatory. The Main specification refers to the instantiation node of the only proposition. Three grammatical functions are indicated, each specifying a particular argument of the predicate. Next, we find unification specifications (impulses). Each indicates a set of constraints: the subject function requires, with an obligatoriness of 0.5, the Main of a noun phrase, which, with likelihood 0.9, has to occur to the left of the present word. The number is singular and the case marking nominative. The initial *t* indicates that no further condition must be obeyed. The object function

necessarily requires one of the alternatives: a noun phrase, a simple subordinate clause, or an infinitive clause. This last alternative specifies that the infinitive occurs with likelihood 0.1 before the present word (and with likelihood 0.9 after it) and must be marked with a *di* case. Moreover the subject of the infinitive is to be made coreferent with the oblique object of the present clause, at the same time as object unification.

We have briefly referred to the only centralized data structures in WEDNESDAY 2 that handle the distribution of constituents and, in operating terms, the restriction of the space where unification can take place. We say that this part is concerned with opening, closing and maintaining search spaces, where entities sought by impulses are possibly found. It is realized with very simple **space management transition networks** (SMTNs), in which $EXP, a distinguished symbol on an arc, indicates that only the occurrence of something expected by the preceding context (i.e., for which an impulse was set up) may allow the transition. SMTNs can impose generalized linear precedence on labeled substrings. They encode information that could be expressed by rewriting rules, factorizing fragments that are common to different rules more perspicaciously. In particular for Italian, which has a substantially freer word order language than English, such networks are very simple. Only locally do they impose an ordering (for instance, prepositions come before what they mark). More often they only exclude intraposition of a constituent not belonging to the current space. It should be noted also that a set of acceptable verbal moods and tenses may be specified on entry points in the SMTNs. For instance, a (main) sentence needs a verb in a finite tense.

It is worth noting that SMTNs are converted into an internal format that includes also a table of *first transition cross references*, i.e., for each space type T, F(T), the set of initial states that allow for transition on T, or, recursively on a space type subsumed by a state in F(T). For example, F(Det) = {NP,S}, at least. This is in the same spirit with the concept of **reachability** discussed in Kay (1980).

3 The Basic Parser

WEDNESDAY 2 is a parser based on an integration of chart and unification. The parser uses linguistic knowledge of the type outlined in Section 2.

A morphological analyzer (Stock, Castelfranchi, and Cecconi 1986) is actually attached to it and unifies data included in stems and affixes. Furthermore, to be more precise, stems are put in hierarchies of prototypes: the result is that linguistic data are much less redundant than appeared in our introductory description. The process leans heavily on the idea of chart (Kay 1980, Kaplan 1973). Chart parsing is a very general concept for non-deterministic parsing (see, for instance, Thompson 1981 and Ritchie and Thompson 1984), historically

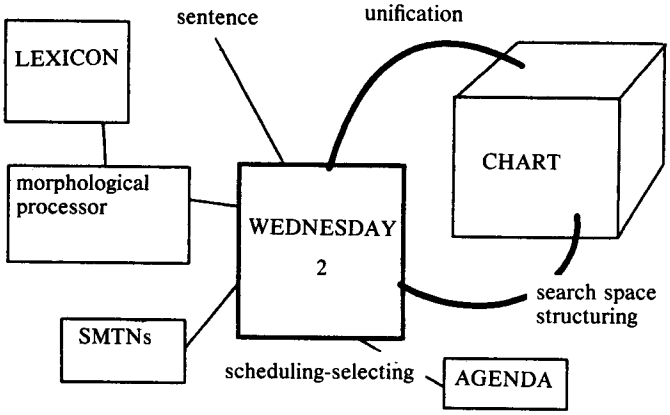


Figure 2. WEDNESDAY 2.

inspired by Earley's (1970) work on CFL parsing. We shall briefly review the main concepts here.

A **chart** is a directed graph that represents the state of the analysis. Given the input string, the junctures between words are called **vertices** and are represented as nodes in the chart. Each vertex has an arbitrary number of arcs, called **edges,** entering and leaving it. An edge is therefore a link between two vertices. In the classic chart definition there are two possible types of edge: **inactive** or **active**. An inactive edge stands for a recognized constituent (the edge spans the words that are included). An active edge represents a partially recognized constituent. For an inactive edge there is a specification of the category of the constituent. In the case of an active edge, a rewriting rule in the grammar and a position on the right hand side of that rule are provided, thus indicating what is still in order to complete the recognition of the constituent. If the rule is R: $C_0 \rightarrow C_1 \ldots C_n$ there are specifications R and *i*, with $0 < i < n$, where *i* is the position on the right hand side of rule R. A word in the string is itself represented as an inactive edge connecting two adjoining vertices. An empty active edge is an active edge that spans no words and is therefore represented in the chart as a link cycling over one vertex. It means, at least in top-down parsing, a prediction of the application of a rule.

It is important to note that edges are only added to the chart, never removed. A new edge may be added in the following ways:

1. Given an active edge A spanning from V_a to V_b and an inactive edge I spanning from V_b to V_c, where A refers to rule R and to position *i*, and the category of I is just C_{i+1}, i + 1-th symbol of the right hand side of R, then a new edge E can be added to the chart, which will span from V_a to V_c, and, if C_{i+1} was the last symbol in R, E will be an inactive edge with category equal to the symbol on the left hand side of R; if not, it will be an active edge with rule R and position i + 1.
2. Empty active edges are placed at particular points in the chart, according to the general strategy used. If

the parser is a top-down one, when, given an active edge with rule R and position *i*, that has reached the vertex V, there is a rule R′ with left hand side equal to C_{i+1}, i + 1-th symbol of the right hand side of R, an empty active edge is introduced on the vertex V, with rule R′, provided that one such edge is not already present on that vertex. If the parser is a bottom-up one, when, given an inactive edge with category C that has reached the vertex V, there is a rule R′ that has C as the first symbol of its right hand side, an empty active edge is introduced on the vertex V, with rule R′, provided that one such edge is not already present on that vertex.

The whole process of parsing aims at getting one or more (if the sentence is ambiguous) inactive edges to span the whole string, with category S the distinguished initial symbol in the grammar.

One of the great advantages of chart parsing is that, whatever the strategy adopted, work is never duplicated. For example, if, after backtracking, the analysis is continued by extending a different active edge and a vertex is reached from where an inactive edge starts, and if the edge addition rule 1 can be applied, then the analysis takes advantage of previously done partial analysis. Another advantage is that the mechanism is perfectly suited for both bottom-up and top-down parsing, depending only on the form of the addition rule 2.

Another point worth mentioning is that the input relation with other levels of analysis is coherent: lexical ambiguity results in the very simple fact that more than one inactive edge is introduced for one ambiguous word. Furthermore, the basic top-down and bottom-up control schemata can be sophisticated in a number of ways, resulting in better performance (smaller number of unnecessary edges introduced in the chart). Wiren (1987) compares a number of proposals.

In WEDNESDAY 2 the chart is the basic structure in which search spaces are defined. An active edge defines an operational environment for unification activity. Some notable overall aspects in the WEDNESDAY 2 chart are:

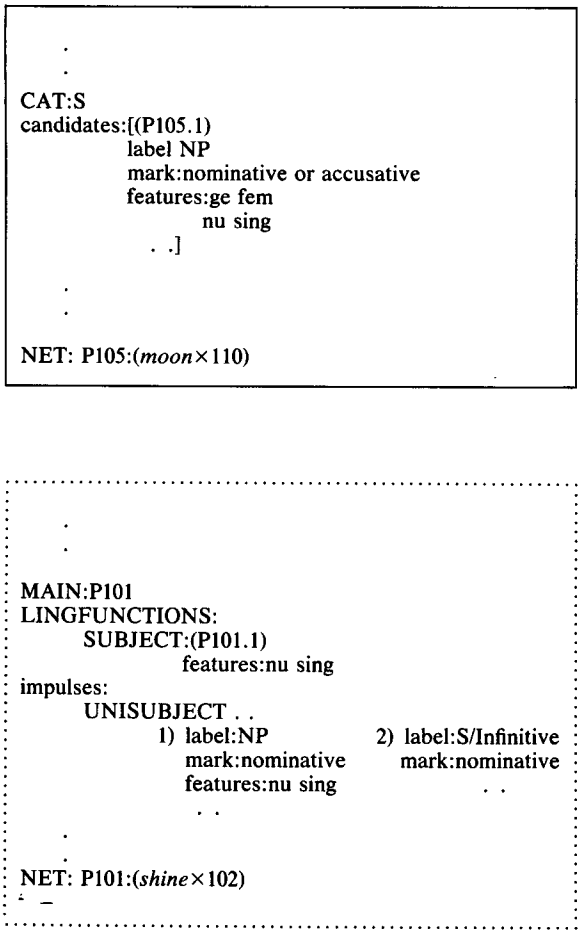
- Instead of referring to a set of rewriting rules, edges refer to positions that are states in SMTNs. The case of the $EXP arc will be discussed in further detail in Section 4.
- Parsing goes basically bottom-up, with top-down confirmation, with an improvement of the so-called **left corner** technique, a version of which is referred to in Wiren (1987). When a word edge with category C is added to the chart, its **first left cross references** F(C) are fetched and, for each of them, an edge is introduced in the chart. These particular edges are called **sleeping edges.** A sleeping edge S at a vertex V_S is "awakened," i.e., causes the introduction of a normal active edge iff there is an active edge arriving at V_S that may be extended with an edge with the category of S. If they are not awakened, sleeping edges play no role at all in the process.
- An agenda is provided. Tasks can be added to the agenda and at every moment a scheduling function can decide the order in which tasks should be performed, in a multiprogramming fashion. The scheduling function can very easily implement depth-first control and breadth-first control, but any kind of control can in principle be included. Tasks are of several kinds, including **lexical tasks, extension tasks, insertion tasks,** and **virtual tasks.** A lexical task specifies a possible reading of a word to be introduced in the chart as an inactive edge. An extension task specifies an active edge and an inactive edge that can extend it (together with some more information). Insertion tasks will be explained in Section 4.1. A virtual task consists in extending an active edge with an edge displaced to another point of the sentence, according to the mechanism treating long-distance dependencies, which is explained in Section 5.

4 Unification

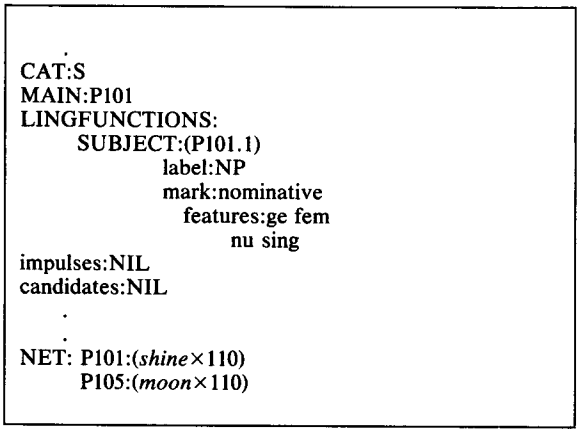
Let us now focus on the actual problem of unification. The sentence is *La luna splende* ("The moon shines"). We consider an active edge with category S, that spans the fragment *la luna,* and an inactive edge with category V, that spans the fragment *splende.* Unification is performed simultaneously with the application of the basic edge extension rule. The first frame in Figure 3a shows part of the situation in the preexisting active S edge, before the V edge contribution (shown in the dotted frame). In the first frame it is emphasized that there is a candidate for unification. P101.1 denotes the first argument of proposition P101. An unsatisfied impulse is emphasized in the dotted frame. Two alternative sets of constraints are actually indicated. The frame in Figure 3b shows the situation within the new S edge after execution.

Let us informally describe the aspect of an edge in WEDNESDAY 2. An edge is composed of a. a structural aspect, that includes a provenience vertex, a destination vertex, a category (corresponding to a space type) and a state in an SMTN; b. a set of syntactic specifications of various kinds; c. a measure of likelihood for this analysis, resulting from the likelihood of partial interpretations and the likelihood of choices performed at the present level; and d. the semantic representation of the fragment, in the form of a semantic network.

The syntactic specifications include, among others, the Main, other functions, sets of constraints to be satisfied (impulses), and candidates for satisfying these sets of constraints. An explicit list of candidates is useful because the unification activity is not continuous, and therefore a memory of pending situations is needed. Candidates emerge from included edges: a typical case is that a Main of an included edge may be taken as a



(a)



(b)

Figure 3. WEDNESDAY 2 Unification.

candidate in the present edge, if the SMTN arc transited for including the edge specifies it. A candidate is composed of a complex data structure, including syntactic features and a reference to a node in the semantic network.

In the actual implementation there are specialized edge slots for different kinds of impulses, in relation to their range: if they modify a structure in a different search space, or if they "fill an argument" for the present frame, etc. Introducing a new edge also means that these data structures are set consistently.

The contents of a word edge are established and instantiated by the lexicon (or, more exactly, by the morphological processor and the lexicon). When a word edge W is used to extend a given edge A, to produce the edge A′ based on the configuration of A, then (without considering the further processing described below) its syntactic and semantic contents are inserted in the corresponding slots of A′. When any other inactive edge I is used to extend an active edge A, then (without considering the further processing described below) some of its contents are inserted in A′ in this way: if I "plays an argumental role" in that space (information that comes from the SMTN arc transited for accepting I), then the value of its Main is placed among the pending candidates of A′. The likelihood figure of A′ is the result of the application of a numeric function to the likelihoods of A′ and I. The semantic net shred of I is copied to A′.

What we shall call the **Principle of the Good Clerk** governs the unification activity:

> Before the arrival of the Main (the "Boss") the processor adopts a lazy attitude, when the Main arrives all possible (unification) activity is done, when the Main is already there an (unification) action is performed as soon as a new event occurs.

Accordingly, before the Main arrives, an extension of an edge causes very little more than the copying actions indicated above. The only check is for mood and tense to be in accordance with what was possibly expected.

On the arrival of the Main, all the present impulses that either have only a syntactic aspect (like the case marking introduced by a preposition) or that are imported from subsumed edges (impulses that bind modifiers, in the linguistic sense) must be satisfactorily carried on and, for each candidate an expectation matching its characteristics must be found. If all this does not happen then the new edge A′ supposed to be added to the chart is not added: the situation is recognized as a failure.

After the arrival of the Main, each new candidate must find an impulse to merge with, and each incoming impulse must find satisfaction. Again, if all this does not happen, the new edge A′ will not be added to the chart. The example in Section 1 informally shows the effect of the principle.

What are the effects of unifying? The value of the linguistic function gets more precise with a reference to the filler node, a category (if not yet specified), a mark as prescribed by the considered alternative in the impulse and features of the two contractors merged together. If, for one feature, there is a set of values for one or both contractors, then the intersection of the values

is considered as the value. Of course, if the intersection is empty, and/or if any of the characteristics do not match, the merger is precluded, and unification fails.

An implementation note: changes are always performed in the active edge, but nonetheless, using specialized data types only selected data need to be copied from underlying edges.

The actual semantic unification happens in the semantic network, represented as a set of labelled propositions. With every entity x (variable or proposition label), R_x, a list reference pairs (proposition and argument number), is associated. In this way, merging x with y causes the substitution of all the occurrences of x, inferred from R_x, with y and the assignment to R_x of the union (without repetitions) of R_x and R_y. Should there be any specified side effects of a merging type (see Section 2), they are carried out in the same way.

In languages with a variable order or, for instance, if subject gapping is admitted, possible mergings are not univocally determined by structure or position. Therefore there may actually be more than one complete merging combination for a tentative new edge A'. In this case more new edges must be added to the chart, one for each alternative unification. This establishes the second kind of task mentioned before—an insertion task, with self-explanatory execution behavior.

By way of example, let us consider the simple sentence: *Il saggio ama la ballerina,* i.e., "The wise man loves the ballerina". When the "loves" edge is taken care of in the S active edge that has already included the NP "the wise man", it happens that two cases must be considered: in one case "the wise man" is the subject, in the other case it is a topicalized object. This is specified in the lexicon with the word, "loves", and different likelihoods are associated with the two possibilities. Two different edges are therefore proposed: the proposals are in the form of insertion tasks specifying the measure of likelihood of the new proposed edges. Up to that point a single path had been followed, with no commitment before the verb was analyzed. Only as a consequence of the verb introduction are the two hypotheses set up, with all the possible commitments involved in each of them and resulting in stronger constraints in the following analysis.

4.1 NON-DETERMINISM AND DISAMBIGUATION

Various kinds of ambiguities can be present in the linguistic knowledge: a word may be semantically ambiguous; an impulse may have a number of alternatives; a case marking may be ambiguous. On top of this there may be idiomatic interpretations competing with the literal ones (see Section 6). And then, of course, the SMTNs express an infinite set of configurations. When a particular set of nondeterministic unifications are to be carried out, an insertion task is introduced in the agenda. The execution of the task leads to the inclusion of a new edge in the chart. Dynamically, apart from the general behavior of the parser, there are some particular restrictions on its nondeterministic behavior which bring syntactic dynamic disambiguation to bear.

1. The $EXP arc allows for a transition only if the configuration in the active edge includes an impulse to link up with the Main of the proposed inactive edge.
2. The sleeping edge mechanism prevents spaces not appropriate to the left context from being established.
3. A search space can be closed only if no impulse specified as having to be satisfied remains. In other words, if in a state with an outgoing EXIT arc, an active edge can lead to the establishment of an inactive edge only if there are no obligatory impulses left.
4. A proposed new edge A' with a verb tense not matching the expected values causes a failure, i.e., A' will not be introduced in the chart.
5. As set out in greater detail in the previous paragraph, failure is caused by inadequate mergings, with relation to the presence, absence, or ongoing introduction of the Main.

Making a comparison with the criteria established for LFG for functional compatibility of an *f*-structure (Kaplan and Bresnan 1982), the following can be said of the dynamics outlined here. **Incompleteness** recognition performs as specified in (3), and furthermore, there is an earlier check when the Main arrives, in case there were obligatory impulses to be satisfied at that point (e.g., an argument that must occur before the Main). **Incoherence** is avoided completely after the Main has arrived by means of the $EXP arc mechanism. Before this, it is recognized as specified in (5) above, and causes an immediate failure. **Inconsistency** is detected as indicated in (4) and (5). As far as (5) is concerned, though, the attitude is to activate impulses when the right premises are present and to look for the right thing and not to check if what was done is consistent.

One hitherto fundamental, but only partially implemented, aspect is a mechanism for semantic disambiguation that would interact with WEDNESDAY 2. In fact, some aspects of the parser have been designed precisely in view of such interaction.

5 Long-Distance Dependencies

Sentences like "The programmer with the brother of whose colleague John said that Mary danced has left" require that something positioned far away in the sentence (in this example, "with the brother of whose colleague") plays the role of filler for a certain gap (in this case bound to "dance"). Constructions of this type are called **long-distance dependencies.** It is generally difficult to relate them back to a context-free mechanism unless some particular device, such as metarules in GPSG, is introduced. In LFG, double arrows are used to denote this and are combined with particular rewriting rules in the grammar that derive *c*-structures.

These rules have the empty element, *e*, on the right-hand side, and therefore gaps are foreseen by the grammar writer.

We take a different approach. First, we do not want the parsing process to be overwhelmed by the appearance of spurious constituents, where a rule that prescribes the introduction of the empty symbol was applied. Of course, the use of empty rewriting rules is attractive to the linguist, because he/she can then give a clear description of the structures. But in parsing we have a problem of the explosion of possible constituents. Second, we want the lexicon to control also the problems and idiosyncrasies involved in limiting the allowed dependencies. To illustrate this point, the solutions given by **Government Binding** theory and by lexical-functional grammar are worth recalling. The former approach considers wh-movements as constrained by 'among other thing' the principle of subjacency, which is a universal structural principle that imposes bounding categories. The latter approach allows imposition of **bounding nodes,** which is a more flexible concept (see Bresnan 1982).[1]

We shall give here a very concise description of how relative clauses are dealt with in WEDNESDAY 2. Let us begin with lexical data. Entries like wh-words carry the information of being the focus of a relative clause, complemented by two kinds of constraints: a. constraints on the path that may be followed upwards before reaching the top-level space of the relative clause. The effect of this in the above example is to allow precisely the following structure: $[_{S'}[_{PP}$ with $[_{NP}$ the brother$[_{PP}$ of$[_{NP}$ whose colleague]]]]. . . .] and cause the main node of the NP "the programmer" to be referenced as the second argument of the relation "colleague"; b. constraints on the path that may be followed downwards from the implied S' space. The effect of this is that the impulse bound to "with" will be allowed to operate in the above example, modifying either "said", or "danced", while irregular sentences will be detected.

We have so far discussed the effects before mentioning the other data implied in the matter: SMTNs. Let us introduce **virtual arcs** to the network. This is a concept that may sound reminiscent of ATNs (Woods 1970). Actually, the role is similar but the concept is quite different. In ATNs a **virt** arc (a.k.a. **retrieve**) specifies the particular kind of phrase that should be retrieved from an implicit register called **hold.** Of course, that particular kind of phrase (for instance, NP or PP) is specified in the central network and it becomes quite unnatural to deal with word idiosyncrasies. The grammar writer must describe all kinds of complex gappings that can occur with different words. WEDNESDAY 2's philosophy reverses the procedure. A virtual arc has nothing specified on it: it can be transited if there is an impulse or a candidate that, although displaced, has been made available by the parser's mechanism, and now is compatible with the situation determined by the configuration of the present particular words. The virtual arc is transited if unification can be carried out. As an example of a fragment of SMTN that includes a virtual arc, see Figure 4.

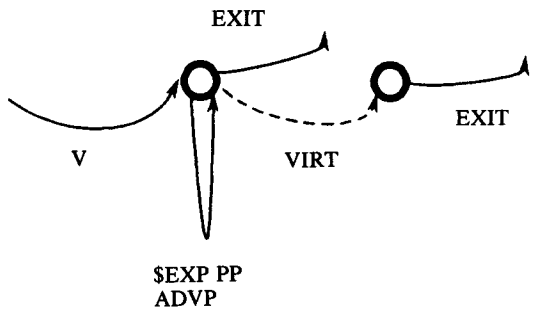


Figure 4. Virtual Arc in an SMTN.

The cycling arc specifies that transitions can be made on whatever is expected by the left context, plus modifiers such as PPs or adverbials. The virtual arc indicates that a proposed displaced element compatible with present functional expectations may be used. Note that virtual arcs are basically inserted in the SMTNs after the Main has been taken care of and therefore unification is immediate. The implementation of this aspect of the parser requires the alteration of the basic chart with the inclusion of topicalized edges, i.e., edges that have a displaced candidate or impulse in evidence, which, by obeying the constraints discussed above, may find its place within that particular edge. If this happens and the parser manages to build an inactive edge, it will be an inactive edge that satisfies the topic. If the inactive edge, and all the inactive edges subsumed by it, do not make use of the topic, the topic simply does not appear at all in the edge. Of course, a necessary condition for the introduction of an inactive top-level topicalized edge (e.g., an edge corresponding to the search space S') is that it satisfies the topic.

In our example, given that in the fragment $[_{S'}[_{PP}$with the brother of whose colleague$][_{S}$John said that$[_{S}$Mary danced]] . . . . , the inactive edge of $[_{S}$Mary danced] satisfies the topic, also $[_{S}$John said that$[_{S}$Mary danced]] satisfies the topic, and therefore an S' inactive edge can be happily introduced in the chart. The mechanism discussed above guarantees one aspect that we have stressed throughout our work: the dynamic unification of partial interpretations. Furthermore, it seems to work quite well with Italian.

6 Idioms

Idioms are a pervasive phenomenon in natural languages. Linguists have proposed different accounts for idioms, which are derived from two basic points of view: one point of view considers idioms as the basic units of language, with holistic characteristics, perhaps including words as a particular case; in the other point

of view the emphasis is instead laid on the fact that idioms are made up of normal parts of speech that play a definite role in the complete idiom. An explicit statement in this approach is the **Principle of Decomposionality** (Wasow, Sag, and Nunberg 1982): "When an expression admits analysis as morphologically or syntactically complex, assume as an operating hypothesis that the sense of the expression arises from the composition of the senses of its constituent parts". The syntactic consequence is that idioms are not a different thing from "normal" forms.

We hold the latter view. We are aware of the fact that the flexibility of an idiom depends on how recognizable its metaphorical origin is. Within flexible word order languages the flexibility of idioms seems to be even more closely linked to the strengths of particular syntactic constructions.

Let us now briefly discuss some computational approaches to idiom understanding. Applied computational systems must necessarily have a capacity for analyzing idioms. In some systems there is a preprocessor delegated to the recognition of idiomatic forms. This preprocessor replaces the group of words that make for one idiom with the word or the words that convey the meaning involved. In ATN systems, specially if oriented towards a particular domain, there are sometimes instead sequences of particular arcs inserted in the network, which, if transited, lead to the recognition of a particular idiom (e.g., PLANES, Waltz 1978). LIFER (Hendrix 1977), one of the most successful applied systems, was based on a semantic grammar, and within this mechanism idiom recognition was easy to implement, without considering flexibility. Of course, there is no intention in any of these systems to give an account of human processing. PHRAN (Wilensky and Arens 1980) is a system in which maximum emphasis is placed on the role of idioms. A particularly interesting aspect is that the system had also a twin generator, called PHRED (Jacobs 1985), that reversed the process, as both are based on the same linguistic representation. Idiom recognition, following Fillmore's (1979) view, is considered the basic resource all the way down to replace the concept of grammar-based parsing. PHRAN is based on a data base of patterns (including single words, at the same level) and proceeds deterministically, at least in its basic implementation, applying the two principles "when in doubt choose the more specific pattern" and "choose the longest pattern". The limits of PHRAN lie in the capacity to generate various alternative interpretations in case of ambiguity (though it must be reported that a nondeterministic implementation was also eventually realized) and in running the risk of having an excessive spread of nonterminal symbols if the data base of idioms is large. A recent work on idioms with a similar perspective and with particular attention to the problem of learning idioms is Dyer and Zernik (1986).

The approach we have followed is different. The goals pursued in our work must be stated explicitly: 1. to yield a cognitive model of idiom processing; and 2. to integrate idioms in our lexical data merely as further information concerning words (as in a traditional dictionary). Idiom understanding is based on normal syntactic analysis with word-driven recognition in the background. When a certain threshold is crossed by the weight of a particular idiom, the latter starts a process of its own, that may eventually lead to a solution. Some of the questions we have dealt with are: How are idioms to be specified? When are they recognized? What happens when they are recognized? And what happens afterwards?

Note that a morphological analyzer, WED-MORPH (Stock et al. 1986), linked to WEDNESDAY 2, plays a substantial role, especially if the language is Italian.

6.1 SPECIFICATION OF IDIOMS IN THE LEXICON

Idioms are introduced in the lexicon as further specifications of words, just as in a normal dictionary. They may be of two types: a. canned phrases, that just behave as several-word entries in the lexicon (there is nothing particularly interesting in that, so we shall not go into detail here); b. flexible idioms; these idioms are described in the lexicon bound to the particular word representing the "thread" of that idiom; in WEDNESDAY 2 terms, this is the word that bears the Main of the immediate constituent including the idiom. Thus if we have an idiom like "to build castles in the air", it will be described along with the verb "to build".

After the normal word specifications, the word may include a list of idiomatic entries. Figure 5 shows a BNF specification of idioms in the lexicon. The symbol + stands for "at least one occurrence of what precedes". Each idiom is described in two sections: the first one describes the elements that characterize that idiom, expressed coherently with the normal characterization of the word; the second one describes the interpretation, i.e., which substitutions should be performed when the idiom is recognized.

Let us briefly describe Figure 5. The lexical form indicates whether passivization (which, in our theory, as in LFG, is treated in the lexicon) is admitted in the idiomatic reading. The idiom-stats, describing configurations of the components of an idiom, are based on the basic impulses included in the word. In other words constituents of an idiom are described as particular fillers of linguistic functions or particular modifiers. For example "build castles in the air", when "build" is in an active form, has "castles" as a further description of the filler of the OBJ function and the string "in the air" as a further specification of a particular modifier that may be attached to the Main node. MORESPECIFIC, the further specification of an impulse to set a filler for a function includes: a reference to one of the possible alternative types of fillers specified in the normal impulse, a specification that describes the fragment that is to play this particular role in the idiom, and

```
<idioms>::=(IDIOMS<idiomentry>+)
<idiomentry>::=(<lexicalform><idiom-stat>+SUBSTITUTIONS<idiomsubst>+)
<lexicalform>::=T/(NOT-PASSIVE)
<idiom-stat>::=(MORESPECIFIC<lingfunc><alternnum><fragmentspec><weight>)/
                (CHANGEIMPULSE<lingfunc><alternative>+<fragmentspec><weight>)/
                (IDMODIFIER <fragmentspec><weight>)/
                (REMOVEIMPULSE<lingfunc>)
<alternative>::=(<test><fillertype><beforelh><features><mark><sideffect><fragmentspec>)
<fragmentspec>::=(WORD<word>)/(FIXWORDS<wordseq>)/(FIRSTWORDS<wordseq>)/
                  (MORPHWORD<wordroot>)/(SEM (<concept>+)<prep>)/(EQSUBJ)
<idiomsubst>::=(SEM-UNITS<sem-unit>+)/(MAIN<node>)/
                (BINDINGS(<lingfunc><node>)+)/
                (NEWBINDINGS(<node><lingfunc path>)+)
```

Figure 5.

the weight that this component has in the overall recognition of the idiom. IDMODIFIER is a specification of a modifier, including the description of the fragment and the weight of this component. CHANGEIMPULSE and REMOVEIMPULSE allow normal syntactic behavior to be modified. The former specifies a new alternative for a filler for an existing function, including the description of the component and its weight (for instance, the new alternative may be a partial NP instead of a complete NP, as in "take care"), or a NP marked differently from usual). The latter specifies that a certain impulse, specified for the word, is to be considered to have been removed for this idiom description.

There are a number of possible fragment specifications, including string patterns, semantic patterns, morphological variations, coreferences, etc.

Substitutions include the semantics of the idiom, which are supposed to take the place of the literal semantics, plus the specification of the new Main and of the bindings for the functions. New bindings may be included to specify new semantic linkings not present in the literal meaning (e.g., "take care of <someone>", if the meaning is "to attend to <someone>", then "<someone>" must become an argument of "attend").

6.2 IDIOM PROCESSING

Idiom processing works in WEDNESDAY 2 by being integrated in the nondeterministic, multiprocessing-based behavior of the parser. As the normal (literal) analysis proceeds and partial representations are built, impulses are monitored in the background, and a check made for possible idiomatic fragments. Monitoring is carried out only for fragments of idioms not in contrast with the present configuration. A dynamic activation table is introduced with the occurrence of a word that has some associated idiom specification. The occurrence of an expected fragment of an idiom in the table raises the level of activation of that idiom, in proportion to the relative weight of the fragment. If the configuration of the sentence contrasts with one fragment, then the relative idiom is removed from the table. So all the normal processing continues, including the possible nondeterministic choices, the establishing of new processes, etc. The activation tables are included in the edges of the chart.

When the activation level of a particular idiom crosses a fixed threshold, a new process is introduced, dedicated to that particular idiom. In that process only that particular idiomatic, interpretation is considered. So, in the first place, an edge is introduced in which substitutions are carried out; the process will proceed with the idiomatic representation. Note that the process begins at that precise point, with all the previous literal analysis acquired by the idiomatic analysis. The original process goes on as well (unless the fragment that caused the new process is nonsyntactic and peculiar to that idiom alone); the idiom is merely removed from the active idiom table. At this point there are two working processes and it is up to the (external) scheduling function to decide priorities. Relevant points are: a. the idiomatic process may still result in failure: further analysis may not confirm what has been hypothesized as an idiom; and b. a different idiomatic process may start from the literal process at a later stage, when its own activation level crosses the threshold. Overall, this yields all the analyses, literal and idiomatic, with likelihoods for the different interpretations. But it also provides a reasonable model of how humans process idioms. Some psycholinguistic experiments have given support to this view, which seems also compatible with the model presented by Swinney and Cutler (1978).

Here we have disregarded the situation in which a possible idiomatic form occurs and its role in disambiguating. The whole parsing mechanism in WEDNESDAY 2 is based on dynamic unification, i.e., at every step in the parsing process a partial interpretation is provided; dynamic choices are performed by scheduling the agenda on the basis of the relation between partial interpretation and the context.

```
(sem-units(n1(p-take n2 n3)))
(likeliradix 0.8)
(main n1)
(lingfunctions (subj n2)(obj n3))
(cat v)
(uni(subj)
    (must 0.7)
    ((t np 0.9 nil nom)))
(uni (obj)
    (must)
    ((t np 0.3 nil acc)))
(idioms((t
        (morespecific (obj) 1 (fixwords il toro) 8)
        (idmodifier (fixwords per le corna) 10)
        substitutions
        (sem-units (m1(p-confront m2 m3))
                   (m4 (p-situation m3))
                   (m5 (p-difficult m3)))
        (main m1)(bindings (subj m2))]
```

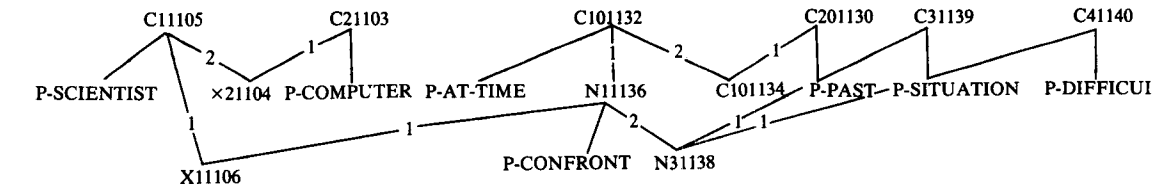
Figure 6.

6.3 AN EXAMPLE

As an example, let us consider the Italian idiom *prendere il toro per le corna* (literally: "to take the bull by the horns"; idiomatically: "to confront a difficult situation"). The verb *prendere* ("to take") in the lexicon includes some descriptions of idioms. Figure 6 shows the representation of *prendere* in the lexicon, but includes only information relevant to our example. The stem representation will be unified with other information and constraints coming from the affixes involved in a particular form of the verb. The first portion of the representation is devoted to the literal interpretation of the word and includes the semantic representation, the likelihood of that reading, and functional information, including the specification of impulses for unification. The numbers are likelihoods of the presence of an argument or of a relative position of an argument. The second portion, after idioms, includes the idioms involving *prendere*. In Figure 6 only one such idiom is specified. It is indicated that the idiom can also occur in a passive form (the first *t* specification) and the specification of the expected fragments is given. The numbers here are the weights of the fragments (the threshold is fixed at 10). The substitutions include the new semantic representation, with the specification of the main node and of the binding of the subject. Note that the surface functional representation will not be destroyed after the substitutions, only the semantic (logical) representation will be recomputed, imposing its own bindings.

As mentioned, Italian allows great flexibility. Let the input sentence be *L'informatico prese per le corna la capra* (literally: "The computer scientist took by the horns the goat"). When *prese* ("took") is analyzed, its idiom activation table is inserted. When the modifier *per le corna* ("by the horns") shows up, the activation of the idiom referred to above crosses the threshold (the sum of the two weights goes up to 12). A new process starts at this point, with the new interpretation unified with the previous interpretation of the subject. Also, semantic specifications coming from the suffixes are reused in the new partial interpretation. The process merely departs from the literal process—no backtracking is performed. At this point we have two processes going on: an idiomatic process, where the interpretation is already "The computer scientist is facing a difficult situation" and a literal process, where, in the background, yet other active idioms monitor the events. In Figure 7 the two semantic representations are shown in the form of semantic networks (they correspond to two sets of propositions).

When the last NP, *la capra* ("the goat"), is recognized, the idiomatic process fails (it needed "the bull" as object). The literal process yields its analysis, but



(a)

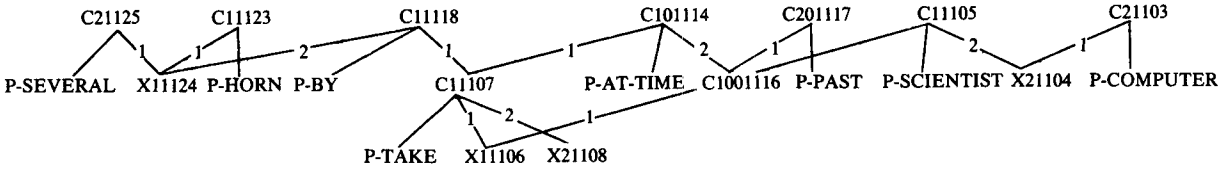


Figure 7.

another idiom crosses the threshold, starts its process with the substitutions, and immediately concludes positively. This latter, unlikely, idiomatic interpretation is that the computer scientist confused the goat and the horns.

7 The Explorative Environment

Computer-based environments for the linguist are conceived as sophisticated workbenches, built on AI workstations around a specific parser, where the linguist can try out his/her ideas about a grammar for a certain natural language. In doing so, he/she can take advantage of rich and easy-to-use graphic interfaces that "know" about linguistics. Of course, behind all this lies the idea that cooperation with linguists will provide better results in NLP. To substantiate this assumption it may be recalled that some of the most interesting recent ideas on syntax have been developed by means of joint contributions from linguists and computational linguists.

Instances of the tools introduced above are the LFG environment, which was probably the first of its kind, an environment built by Ron Kaplan for lexical-functional grammars, D-PATR, built by Lauri Karttunen (Karttunen 1986) and conceived as an environment that would suit linguists of a number of different schools all committed to a view of parsing as a process that makes use of a unification algorithm.

The environment we have developed has a somewhat different purpose. Besides a number of tools for entering data in graphic mode and inspecting resulting structures, it provides a means for experimenting with strategies in the course of the parsing process. We think that this can be a valuable tool for gaining insight into the cognitive aspects of language processing as well as for tailoring the behaviour of the processor when used with a particular sublanguage.

In this way an attempt can be made to answer basic questions in the course of a nondeterministic approach: what heuristics to apply when facing a certain choice point, what to do when facing a failure point, i.e., which of the pending processes to activate, taking account of information resulting from the failure? Of course this kind of environment makes sense only because the parser it works on has some characteristics that make it a psychologically interesting realization.

Psychologically motivated parsers may be put in three main categories. First, those that embody a strong claim on the specification of the general control structure of the human parsing mechanism. The authors usually consider the level of basic control of the system as the level they are simulating and are not concerned with more particular heuristics. An instance of this class of parsers in Marcus's parser (Marcus 1979), based on the claim that, basically, parsing is a deterministic process: only sentences that we perceive as surprising (the so-called "garden paths") actually imply backtracking. Connectionist parsers are also instances of this category. The second category refers to general linguistic performance notions such as the **Lexical Preference Principle** and the **Final Argument Principle** (Fodor, Bresnan, and Kaplan 1982). It includes theories of processing like the one expressed by Wanner and Maratsos for ATNs in the mid-'70s. In this category the arguments are at the level of general structural preference analysis. A third category tends, at every stage of the parsing process, to consider the full complexity of the data and the hypothetical partial internal representation of the sentence, including, at least in principle, interaction with knowledge of the world, aspects of memory, and particular task-oriented behavior. Worth mentioning here are Church and Patil (1982) and Barton et al. (1987), who attempt to put some order in the chaos of complexity and computational load.

Our parser lies between the second and the third of the above categories. The parser is seen as a nondeterministic apparatus that disambiguates and gives a "shallow" interpretation and an incremental functional representation of each processed fragment of the sentence, and allows choices to be made among alternatives in a cognitively meaningful way. We shall briefly describe the environment and, by way of example, illustrate its behavior by analyzing oscillating sentences, i.e., sentences in which one first perceives a fragment in one way, then changes one's mind and takes it in a different way, then, as further input comes in, goes back to the previous pattern (and possibly continuing like this till the end of the sentence).

7.1 Overview of the Environment

WEDNESDAY 2 and its environment are implemented on a Xerox Lisp Machine. The coverage of WEDNESDAY 2 with the present Italian data includes subordinates, complex relative clauses, interrogatives, and so on, all this possibly mixed with idioms. The current lexicon is about 1,500 stems, resulting in about 30,000 forms, but it is very easy to extend it, with the tools built around the parser. An effort is currently being made to revisit linguistic specifications in the lexicon. Moreover, we have been reversing the process for the purpose of generation (Caraceni and Stock 1987). We think that, beside the interest of generation per se and the usefulness of having a generator based on the same principles and data of the parser, this is, paradoxically, the real test for our parser. The results, so far, have been very encouraging, including the fact that it was not hard to make good the deficiencies we detected in our parser when dealing with problems the other way round.

The environment is composed of a series of specialized tools, each one based on one or more windows (Figure 8).

Using a mouse the user selects a desired behavior from menus attached to the windows. We have the following windows:

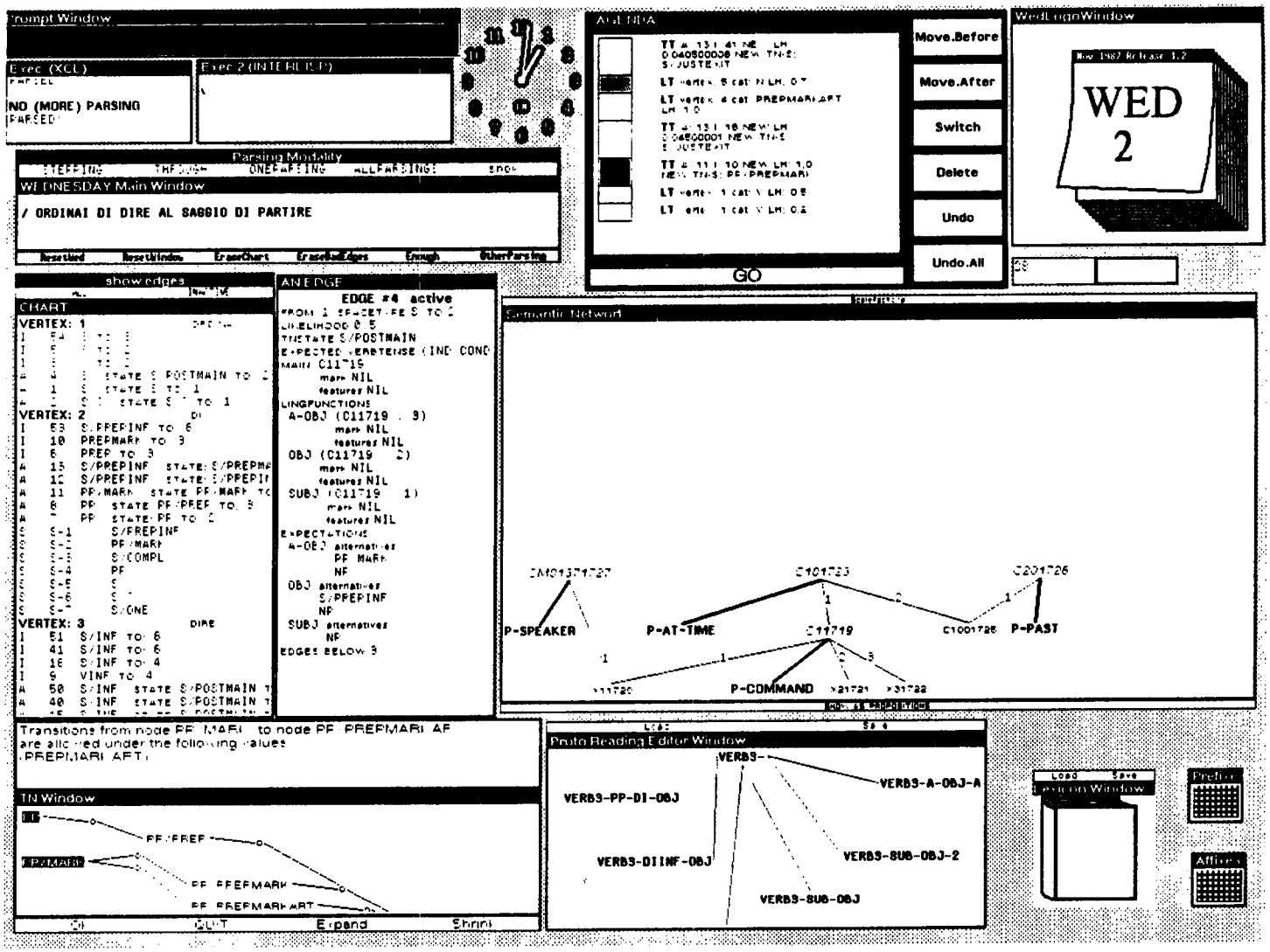


Figure 8.

- the main WEDNESDAY 2 window, in which the sentence is entered. Menus attached to this window specify different modalities (including "through" and "stepping", "all parsings" or "one parsing") and a number of facilities;
- a window where one can view, enter, and modify transition networks graphically (Figure 9).
- a window where one can view, enter and modify the lexicon. As a word reading is a complex object for WEDNESDAY 2, entering a new word can be greatly facilitated by a set of subwindows, each specialized in one aspect of the word, "knowing" what it may be like and facilitating editing.

The lexicon is a lexicon of stems: a morphological analyzer and a lexicon manager are integrated in the system. Let us briefly describe this point. A lexicalist theory such as ours requires that a large quantity of information be included in the lexicon. This information has different origins: some comes from the stem and some from the affixes. All the information must be put into a coherent data structure, through a particularly constrained unification-based process. Furthermore, we must emphasize the fact that, just as in LFG, phenomena such as passivization are treated in the lexicon (the Subject and Object functions and the related impulses attached to the active form are rearranged). This is something that the morphological analyzer must deal with. The internal behavior of the morphological analyzer is beyond the scope of the present paper. We shall instead briefly discuss the lexicon manager, the role of which will be emphasized here.

The lexicon manager deals with the complex process of entering data, maintaining, and preprocessing the lexicon. One notable aspect is that we have arranged the lexicon on a hierachical basis according to inheritance, so that properties of a particular word can be inherited from a word class and a word class can inherit aspects from another class. One consequence of this is that we

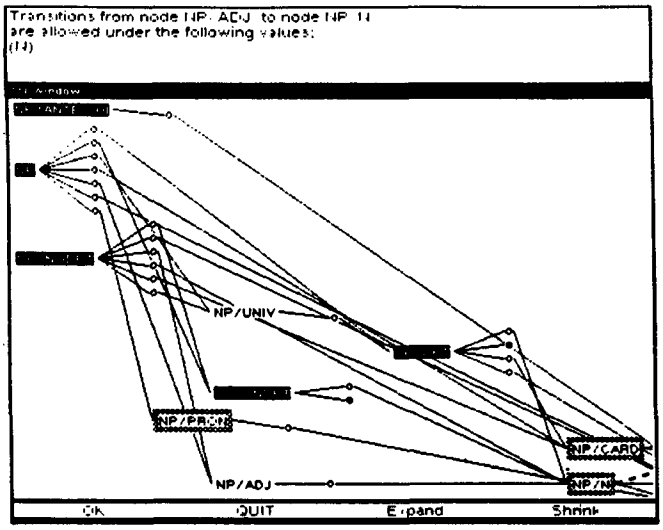


Figure 9.

can introduce a graphic aspect (Figure 10) and the user can browse through the lattice (the lexicon appears as a tree of classes where one has specialized editors at each level). What is even more relevant is the fact that one can factorize knowledge that is in the lexicon, so that if one particular phenomenon needs to be treated differently, the change of information is immediate for the words concerned. Of course, this also means that there is a space gain: the same information does not need to be duplicated—complete word data are reconstructed when required.

There is also a modality by which one can enter the syntactic aspects of a word through examples, à la TEAM (Grosz 1984).

- a window showing the present configuration of the chart;
- a window that allows zooming into one edge, showing several aspects of the edge, including its structural aspects, its likelihood, the functional aspect, the specification of unrealized impulses, etc.
- a window graphically displaying the semantic interpretation of an edge as a semantic net or, if one prefers (this is usually the case when the net is too complex), in logical format;
- a window where one can manipulate the agenda (Figure 11).

Attached to this window we have a menu including a set of functions that the tasks included in the agenda to be manipulated: "move before", "move after", "delete", "switch", "undo", etc. One just points to the two particular tasks one wishes to operate on with the mouse and then to the menu entry, thus obtaining the desired effect. The same effect could be obtained by applying a different scheduling function: the tasks will be picked up in the order here prescribed by manual scheduling. This tool, when the parser is in the stepping modality, provides a very easy way of altering the default behavior of the system and of trying out new strategies. The manual scheduling mode is supplemented by a series of counters that provide control over the penetrance of these strategies. (The penetrance of a nondeterministic algorithm is the ratio between the

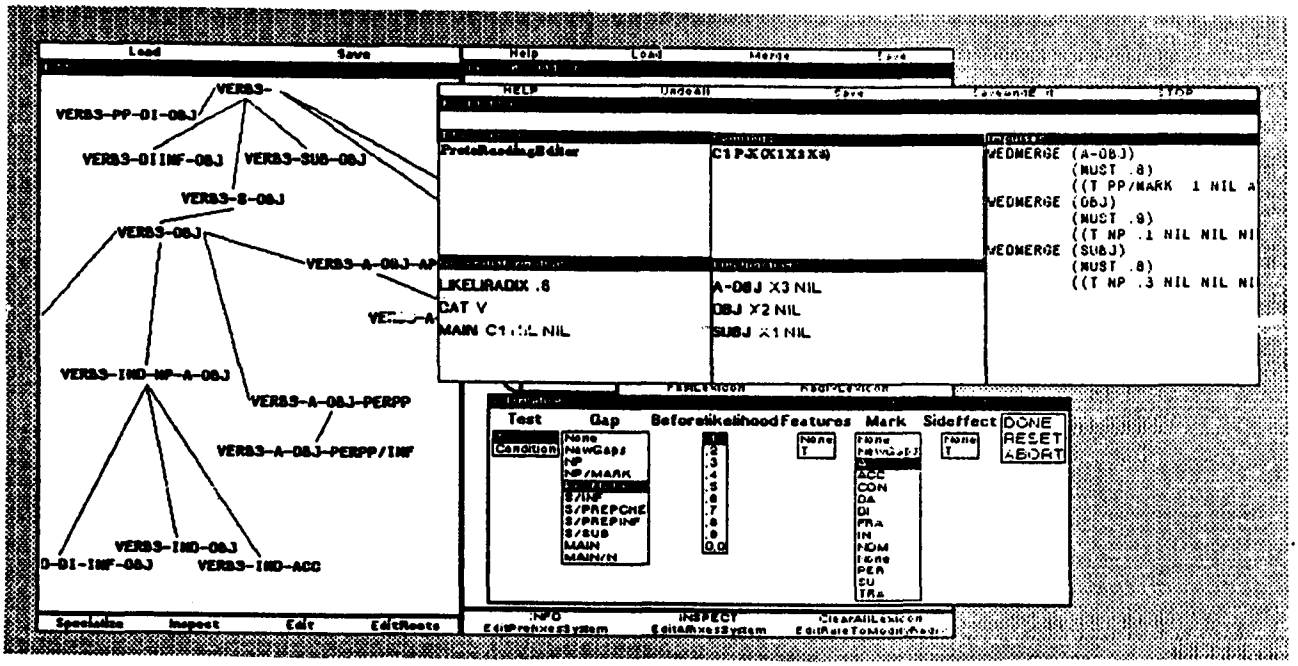


Figure 10.

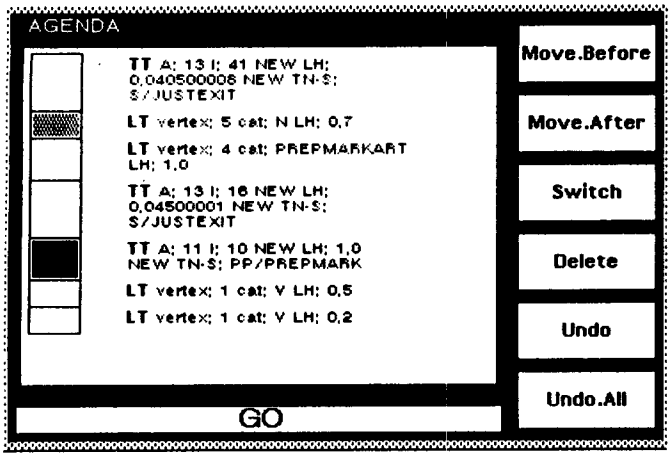


Figure 11.

steps that lead to the solution and the overall steps carried out in trying to obtain the solution. Of course this measure lies between 0 and 1).

Dynamically, it is attempted to find sensible strategies, by interacting with the agenda. When, after formalizable heuristics have been tried out, they can be introduced permanently into the system through a given specialized function. This is the only place where some knowledge of Lisp and of the internal structure of WEDNESDAY 2 is required.

7.2 AN EXAMPLE OF EXPLORATION: OSCILLATING SENTENCES

We shall now briefly discuss a processing example that we have been able to understand using the environment described above. The following example is a good instance of flexibility and parsing problems present in Italian:

A Napoli preferisco Roma a Milano.

The complete sentence reads "while in Naples I prefer Rome to Milan". The problem arises during the parsing process with the fact that the "to" argument of "prefer" in Italian may occur before the verb, and the locative preposition "in" is *a,* the same word as the marking preposition corresponding to "to".

The reader/hearer first takes *a Napoli* as an adverbial location, then, as the verb *preferisco* is perceived, *a Napoli* is clearly reinterpreted as an argument of the verb (in the sense of surprise). As the sentence proceeds after the object *Roma,* the new word *a* causes things to change again and we go back with a feeling of surprise to the first hypothesis. When this second reconsideration takes place, we feel the surprise, but this does not cause us to reconsider the sentence, we only go back to adding further to an hypothesis we were already working at. It should also be noted that the surprise seems to be caused not by a heavy computational load, but by a sudden readjustment of the weights of the hypotheses. In a sense it is a matter of memory, rather than computation.

We have succeeded in getting WEDNESDAY 2 to perform naturally in such situations, taking advantage of the environment. The following simple heuristics were found: a. try solutions that satisfy the impulses (if there are alternatives consider likelihoods); b. maintain viscosity (prefer the path you are already following); and c. follow the alternative that yields the edge with the greatest likelihood chosen among edges of comparable length.

The likelihood of an edge depends on: 1. the likelihood of the included edges; 2. the level of obligatoriness of the filled impulses; 3. the likelihood of a particular relative position of an argument in the string; 4. the likelihood of that transition in the network, given the previous transition.

The critical points in the sentence are the following (note that we distinguish between a PP and a "marked NP" possible argument of a verb, where the preposition has no associated semantics):

i. At the beginning: only the PP edge is expanded, (not the one including a marked NP) because of static preference for the former expressed in the lexicon and in the transition network.

ii. After the verb is detected: on the one hand there is an edge that, if extended, would not satisfy an obligatory impulse, on the other hand, one that possibly would. The marked NP alternative is chosen because of (a) of the above heuristics.

iii. After the object *Roma:* when the preposition *a* comes in, the edge that can extend the sentence with a PP on the one hand, and on the other hand a cycling active edge that is a promising satisfaction for an impulse are compared. Since this relative position of the argument is so favorable for the particular verb *preferisco* (.9 to .1 for this position compared to the preceding one), the parser proceeds with the alternative view, taking *a Napoli,* as a modifier, and so on, after reentering the working hypothesis. The object is already there, analyzed for the other reading and does not need to be reanalyzed. So *a Milano* is taken as the filler for the impulse and the analysis is concluded properly.

CONCLUSIONS

We have focused on some important features of a parser, e.g., able to perform unification while working on structures, able to analyze idiomatic forms, providing a good testbed for ideas about dynamic strategies. The parser we have introduced is based on linguistic knowledge distributed fundamentally through the lexicon and uses a chart with unification activity developed when new edges are established. Especially if word order is flexible, it is important to avoid redundancy and detect useless hypotheses at an early stage. Here, chart

parsing is based on a bottom up with left context confirmation strategy and unification is based on the Principle of the Good Clerk. Also, we have proposed a view of the idiom recognition process that is completely integrated with the parser, so that, in particular, idiom recognition can take advantage of the flexibility of the approach without redundancy. We think that this is important, because idioms are on a continuum with literal language, and, share the "normal" characteristics, including the flexibility, of the particular language involved. Moreover, given that the algorithm is of a multiprocessing type, one can consider what strategies can be introduced in order to select dynamically the tasks that have best chances of leading to the (preferable) final analysis. These heuristics can also take advantage of measures of likelihood associated with each partial analysis. An integrated interactive environment has been built to experiment with these ideas, which can also be used to define sublanguages and select strategies for particular applications.

Acknowledgments

The author wishes to thank the following people who advised him, probably without much hope, at different stages of this work: C. Cacciari, C. Castelfranchi, R. Kaplan, M. Kay. Special thanks to F. Cecconi for his contribution to the implementation of the environment. Particular thanks go to the anonymous referees that have helped (hopefully) in making a readable paper out of an unreadable one.

Most of the work was carried out while the author was working for the Italian National Research Council, at the Istituto di Psicologia in Rome, and reaped particular benefit from the resources of the Artificial Intelligence Strategic Project, which, in spite of its name, was a purely civilian, basic research initiative.

References

Aho, A.V. and Ullman, J.D. 1972 *The Theory of Parsing, Translation, and Compiling*. Vol. I: Parsing. Prentice-Hall, Englewood Cliffs, NJ.

Barton, E.; Berwick, R.; and Ristad, E. 1987 *Computational Complexity and Natural Language*. MIT Press, Cambridge, MA.

Bresnan, J. (ed.) 1982 *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Caraceni, R. and Stock, O. 1987 Reversing a Lexically Based Parser for Generation. Applied Artificial Intelligence. An International Journal 1:149–174.

Church, K. and Patil, R. 1982 Coping with syntactic ambiguity or how to put the block in the box on the table. *American Journal of Computational Linguistics* 8: 139–149.

Dyer, M. and Zernik, U. 1986 Encoding and Acquiring Meaning for Figurative Phrases. In *Proceedings of the 24th Meeting of the Association for Computational Linguistics*, New York, NY.

Earley, J. 1970 An Efficient Context-free Parsing Algorithm. *Communications of the Association for Computing Machinery* 13(2): 94–102.

Ferrari, G. and Stock, O. 1980 Strategy Selection for an ATN Syntactic Parser. In *Proceedings of the 18th Meeting of the Association for Computational Linguistics*, Philadelphia, PA.

Fillmore, C. 1979 Innocence: a Second Idealization for Linguistics. In Proceedings of the Fifth Annual Meeting of the Berkeley Linguistics Society, University of California at Berkeley: 63–76.

Ford, M.; Bresnan, J.; and Kaplan, R. 1982 A Competence-Based Theory of Syntactic Closure. In Bresnan, J. (ed.) The Mental Representation of Grammatical Relations. MIT Press, Cambridge, MA.

Gazdar, G. and Pullum, G.K. 1982 *Generalized Phrase Structure Grammar: A Theoretical Synopsis*. Indiana University Linguistics Club, Bloomington, IN.

Grosz, B. 1983 TEAM, a Transportable Natural Language Interface System. In *Proceedings of the Conference on Applied Natural Language Processing*, Santa Monica, CA.

Hayes, P.J. and Carbonell, J.G. 1981 Multi-Strategy Parsing and its Role in Robust Man-Machine Communication. Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

Hendrix, G.G. 1977 LIFER: a Natural Language Interface Facility. *SIGART Newsletter* 61.

Jacobs, P. 1985 PHRED: A Generator for Natural Language Interfaces. *Computational Linguistics* 11(4): 219–242.

Joshi, A., and Levy, L. 1982 Phrase Structure Trees Bear More Fruits Than You Would Have Thought. *American Journal of Computational Linguistics* 8: 1–11.

Kaplan, R. 1973 Augmented Transition Networks as Psychological Models of Sentence Comprehension. *Artificial Intelligence* 3: 77–100.

Kaplan, R. 1973 A General Syntactic Processor. In Rustin, R. (ed.) *Natural Language Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Kaplan, R. and Bresnan, J. 1982 Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Bresnan, J. (ed.) *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA: 173–281.

Kaplan, R., Maxwell, J., and Zaenen, A. 1987 Functional Uncertainty. *CSLI Monthly* 2(4): 1–6.

Karttunen, L. 1986 D-PATR: A Development Environment for Unification-Based Grammars, Report No. CSLI-86-61. Center for the Study of Language and Information. Palo Alto, CA.

Kay, M. 1979 Functional Grammar. In *Proceedings of the Fifth Meeting of the Berkeley Linguistic Society*, Berkeley, CA: 142–158.

Kay, M. 1980 *Algorithm Schemata and Data Structures in Syntactic Processing*. Xerox, Palo Alto Research Center, Palo Alto, CA.

Kay, M. 1985 Parsing in Functional Unification Grammar. In Dowty, D.; Karttunen, L.; and Zwicky, A. *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*. Cambridge University Press, Cambridge, England.

Marcus, M. 1979 *An Overview of a Theory of Syntactic Recognition for Natural Language* (AI memo 531). MIT Artificial Intelligence Laboratory, Cambridge, MA.

Pereira, F. and Warren, D. 1980 Definite Clause Grammars for Language Analysis. A Survey of the Formalism and a Comparison with Augmented Transition Networks. *Artificial Intelligence* 13: 231–278.

Ritchie, G. and Thompson, H. 1984 Implementing Natural Language Parsers. In O'Shea, T. and Eisenstadt, M. (eds.) *Artificial Intelligence: Tools, Techniques, and Applications*. Harper and Row. New York, NY.

Shieber, S.M. 1986 An Introduction to Unification-Based Approaches to Grammar. CSLI Lecture Notes Series No. 4, University of Chicago Press, Chicago, IL.

Shieber, S.M.; Uzkoreit, H.; Pereira, F.; Robinson, J.; and Tyson, M. 1983 The Formalism and Implementation of PATR II. In Grosz B. and Stickel M. (eds.) *Research on Interactive Acquisition and Use of Knowledge*, SRI Report 1894, SRI International, Menlo Park, CA.

Small, S. 1980 *Word Expert Parsing: a Theory of Distributed Word-Based Natural Language Understanding*. Technical Report TR-954 NSG-7253. University of Maryland, Baltimore, MD.

Stock, O. 1986 Dynamic Unification in Lexically Based Parsing. In *Proceedings of the Seventh European Conference on Artificial Intelligence;* Brighton, England: 212–221.

Stock, O.; Cecconi, F., and Castelfranchi, C. 1986 Analisi Morfologica Integrata in un Parser a Conoscenze Linguistiche Distribuite, In *Atti del Convegno AICA-86*, Palermo, Italy.

Swinney, D.A., and Cutler, A. 1978 The Access and Processing of Idiomatic Expressions. *Journal of Verbal Learning and Verbal Behaviour* 18, 523–534.

Thompson, H.S. 1981 Chart Parsing and Rule Schemata in GPSG. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, Alexandria, VA.

Tomita, M. 1985 An Efficient Context-Free Parsing Algorithm for Natural Languages. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

Waltz, D. 1978 An English Language Question Answering System for a Large Relational Database. *Communications of the Association for Computing Machinery* 21(7).

Wanner, E. and Maratsos, M. 1978 An ATN Approach to Comprehension. In Halle, M.; Bresnan, J.; and Miller, G.A. (eds.): *Linguistic Theory and Psychological Reality*. MIT Press, Cambridge, MA.

Wasow, T.; Sag, I.; and Nunberg, G. 1982 Idioms: an Interim Report. *Preprints of the International Congress of Linguistics*, Tokyo, Japan: 87–96.

Wilensky, R. and Arens, Y. 1980 *PHRAN—A Knowledge-Based Approach to Natural Language Analysis*. ERL Memorandum No. UCB/ERL M80/34. University of California at Berkeley, CA.

Wiren, M. 1987 A Comparison of Rule Invocation Strategies in Context-free Chart Parsing. In *Proceedings of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, Denmark.

Woods, W. 1970 Transition Network Grammars for Natural Language Analysis. *Communications of the Association for Computing Machinery* 13.

Note

1. A recent alternative treatment of long-distance dependencies within the LFG approach, which gives away with structure-based operations, is Kaplan et al. (1987).