THE PARALLEL EXPERT PARSER: A MEANING-ORIENTED,

LEXICALLY-GUIDED, PARALLEL-INTERACTIVE

MODEL OF NATURAL LANGUAGE UNDERSTANDING

G. ADRIAENS
Siemens NLP Research
& Katholieke Universiteit Leuven
M. Theresiastraat 21
B-3000 Leuven, Belgium
+32 16 285091
(siegeert@kulcs.uucp or
siegeert@blekul60.bitnet or
siegeert@cs.kuleuven.ac.be)

Abstract

The Parallel Expert Parser (PEP) is a natural language analysis model
belonging to the interactive model paradigm that stresses the parallel
interaction of relatively small distributed knowledge components to arrive
at the meaning of a fragment of text. It borrows the idea of words as
basic dynamic entities triggering a set of interactive processes from the
Word Expert Parser (Small 1980), but tries to improve on the clarity of
interactive processes and on the organization of lexically-distributed
knowledge.   As of now, especially the procedural aspects have received
attention: instead of having wild-running uncontrollable interactions,
PEP restricts the interactions to explicit communications on a structured
blackboard; the communication protocols are a compromise between maximum
parallelism and controllability. At the same time, it is no longer just
words that trigger processes; words create larger units (constituents),
that are in turn interacting entities on a higher level.   Lexical
experts contribute their associated knowledge, create higher-level
experts, and die away.   The linguists define the levels to be considered,
and write expert processes in a language that tries to hide the procedural
aspects of the parallel-interactive model from them.   Problems include
the possibility of deadlock situations when processes wait infinitely for
each other, the way to efficiently pursue different alternatives (as of
now, the system just uses don't-care determinism), and testing whether the
protocols allow linguists to fully express their needs.   PEP has
been implemented in Flat Concurrent Prolog, using the Logix programming
environment.   Current research is oriented more towards the problem of
distributed knowledge representation.   Abstractions and generalizations
across lexical experts could be made using principles from object-oriented
programming (introducing generic, prototypical experts; cp. Hahn 1987).
Thoughts also go in the direction of an integration of the coarse-grained
parallelism with knowledge representation in a fine-grained parallel
(connectionist) way.

1. Introduction

In the course of the last decade, interest in parallel machines and applications has steadily been growing in the different disciplines that deal with natural language understanding (NLU). This of course holds in the first place for researchers in computer science and AI, who have always been interested in computational processes (see e.g. Kowalik 1988). Recent developments in (computational) linguistics and cognitive psycholinguistics show that these NLU-related disciplines have also been moving towards parallel models. A major factor influencing this development is the rapidly growing interest in *interactive* approaches to NLU (see e.g. Briscoe 1987, Altmann 1988, Altmann & Steedman 1988). Briefly, these models move away from the traditional linguistics-inspired views of language understanding as non-interactive, i.e. as a serial application of processing modules whose sole means of communication is a unidirectional input-output channel (cp. Forster 1979). There is a growing belief (based on experience in computational linguistics and on psycholinguistic experimentation) that non-interactive models are incorrect, and interactive ones allowing more flexible communications among components (mainly to deal with ambiguity) prove to be superior. Although matters are more complicated than stated here, it will be clear that interactive models lend themselves easily to parallel architectures (see also Adriaens & Hahn forthcoming). For the presentation of the Parallel Expert Parser, I will only briefly distinguish two kinds of approaches to parallel NLU. On the one hand, there is what can be called *fine-grain parallelism;* on the other hand, there is *coarse-grain parallelism*. With *fine-grain parallel NLU* I refer basically to the connectionist approach and its decendants. Connectionist models feature huge networks of small nodes of information; computation is represented by fluctuations of the activation levels of nodes and by (parallel) transmission of excitation and inhibition along connections. (For connectionism in general, see Feldman & Ballard 1982, VanLehn 1984, Hillis 1986, McClelland & Rumelhart 1986; for connectionist models of NLU, see Cottrell & Small 1983, Cottrell 1985, Pollack & Waltz 1985, McClelland & Rumelhart 1986). With *coarse-grain parallel NLU,* I refer to a more modest kind, in which the smallest item of information is more complex than a node in a connectionist model (it may be a rule, for instance), but in which one attempts to keep the parallel computation involving the items of information more under control than can be done in a connectionist model. (For examples of coarse-grain parallel NLU, see Hirakawa 1983, Matsumoto 1987 or Granger, Eiselt & Holbrook 1986).

The research presented here is of the latter type of parallel NLU. A potentially parallel NLU system (the Word Expert Parser, Small 1980) has been drastically revised so as to allow a truly parallel implementation (viz. in Flat Concurrent Prolog, using the Logix environment (Silverman et al. 1986)); we call the resulting system the Parallel Expert Parser (PEP, Devos 1987).

2. The Word Expert Parser (WEP) revisited

The Word Expert Parser (WEP, Small 1980) is a natural language understanding program in the AI tradition of semantic parsing (see also Hirst 1983, Hahn 1986/1987, Cottrell 1985, Adriaens 1986a/b for WEP-inspired or -related work). The organization of the model differs strongly from that of a "classical" NLU system. Rather than having a number of components of rules that are applied (serially) to linguistic

input by a general process, WEP considers the words themselves as active agents (word experts) that interact with each other and with other knowledge sources in order to find the meaning of a fragment of text. Words are implemented as coroutines, i.e. processes that run for a while (broadcasting information or performing side-effect operations to refine the representation of the meaning of a text fragment), and suspend when they have to wait for information from other experts. The information they send or wait for are either signals relating to the status of the parsing process (broadcast on a dedicated signal channel) or concepts that represent the meaning of parts of the linguistic input (broadcast on a dedicated concept channel). The experts coordinate the understanding process in turn, eventually converging towards a conceptual structure that represents the meaning of a text fragment.

Although the model inspired several researchers, it has received little attention in the linguistic community (but see Berwick 1983) and has been considered as "an interesting rarity". Moreover, the original researchers have abandoned the model in favor of the connectionist approach mentioned above. Yet, the original model still has some interesting features that are worthy of further consideration; on the other hand, both for linguistic and for computational reasons, some drastic revisions are needed.

3. From WEP to PEP

In general, the idea of parallel interacting processes is a very attractive one if one wants a flexible parser capable of using any type of information at any moment it needs it. This basic principle of WEP has been retained for PEP. Yet, although the design of the system seemed to lend itself easily to a parallel implementation, linguistic and computational flaws in the model have made drastic revisions necessary before this could actually be done.

3.1 True parallelism

Although WEP claimed to be "potentially parallel", it heavily (and implicitly) relied on sequentiality to make its principles work. Especially for the restarting of suspended experts, a last-in first-out regime (stack) took care of contention for messages: the expert that placed an expectation for a message last, mostly got it first. Also, to avoid complications in expert communication, no new experts were initialized before the queue of ready-to-run experts was empty. The adherence to this sequentialization, not to mention the side-effects involved, obviously made WEP's claim of being "potentially parallel" invalid.

In a truly parallel environment, sequentiality can no longer be relied on. PEP uses parallelism whenever possible: for the execution of expert code AND for initializing new experts (initializing all of them as soon as they are read and morphologically analyzed). In order to realize this, the most important departure from the original model is that experts are no longer only associated with words (the only linguistic entities acknowledged by WEP). We will now discuss what experts are associated with, and how the new view of experts leads to clearer and more explicit concepts of waiting and communicating in a parallel environment.

3.2 Word-Experts versus Concept-Experts at different levels

A major item of criticism uttered against WEP has been that it considers the word as the only entity to be turned into an expert process. Linguistically speaking, the existence of larger constituents is undeniable and must be taken into account, whatever model one advocates. From the computational viewpoint, squeezing all interactions into words makes it almost impossible to figure out what is going on in the overall parsing process (This non-transparency is one of the reasons why fully-integrated interactive models are not so popular). Words have to decide on everything, from morphological issues to pragmatic issues, with jammed communication channels as a result.

In PEP, experts are associated with *concepts* rather than with words. It is very natural to do so: words are only used to evoke the concepts that constitute the meaning of a fragment of text. Still, concepts have a concrete link to words and can be regarded as being associated with the group of words that evokes them. E.g. in "the young girl" three concepts can be discovered, associated with the basic word-groups "the", "young" and "girl". At a higher level a compound concept constituting the meaning of the entire construct "the young girl" is invoked.

Concretely, in PEP a specific data structure (the *expert frame*) is associated with every expert. The hierarchy that originates from the concepts is reflected by the interconnection of the expert frames. These are vertically related by level interdependencies, and horizontally by the relative role the concepts of the frames play in the frame that is being built out of them one level higher. Besides its level, an expert frame has three attribute slots: a *function* attribute (stating what the role is the expert concept plays at a specific level), a *concept* attribute (representing the contents of the expert) and a *lexical* attribute (simply corresponding to the group of words associated with the concept). Below, we will see that this definition of an expert frame is crucial for the restricted communication protocol among experts.

The "analysis process" consists of the collection of currently active experts that try to establish new concepts. If a new concept can successfully be formed, the corresponding expert is added to the analysis process, while the combined concept's experts may die. They pass their expert frames, and so the contained information, to the new expert, which will usually incorporate them in its own expert frame. Notice that this view has interesting software engineering aspects not present in WEP: by having a leveled approach expert code becomes more local, modular and adaptable. The dynamic process hierarchy enables the linguist/expert writer to write generic or prototypical experts that can be parameterized with the value of the concept they represent (cp. object-oriented programming).

A final note about the levels. Each level is intended to deal with a more or less independent part in the derivation and composition of meaning. However, we leave it up to the linguists writing the expert processes to declare (1) what levels they want to consider and (2) what the appropriate functions are that they want to use at the respective levels. By combining this flexible filling in of a rigorously defined model, we force the linguist-user to clearly specify the experts and help him to keep the experts relatively small (hence, more readable) and to figure out more easily where things could go wrong in the parsing process. A possible hierarchy of levels might be: morpheme, word, constituent, clause, sentence (each level having its own function attributes). In the somewhat oversimplified example below, three levels will be used (between brackets: the respective function attributes), viz. word_level [article;adjective;substantive], constituent_level [action;agent;object], and sentence_level.

3.3 Broadcasting versus Explicit Communication

Experts are the active components of the analysis system. New concepts come into existence only through their interaction. Since parallelism was a major goal of the PEP approach, the communication protocols have been based on explicit identification of the expert frames involved in some interaction, which allows one to keep communication under control. Two kinds of communication take place:

(1) attribute-refining:

Experts are allowed to refine the attributes of expert frames. The attributes are considered to be information that is accessible by all experts.

(2) attribute-probing:

Basing themselves on the attributes of the probed expert frames, experts decide which way to go in the analysis process. All attribute probing is in the choose_alt predicate, that is described in the next subsection.

3.4 Suspending/Resuming:
Explicit Machinery versus Declarative Reading

In the course of the analysis process, experts walk through a discrimination network, gradually refining and constructing the meaning of a text fragment. The predicate that allows experts to decide which way to go in this process on the basis of information they expect to get from other experts is the *choose_alt* predicate:

```
choose_alt([
        alt(frame(frame-specification,
                attribute_condition),
                    invoke(expert)),
        alt(frame(frame-specification,
                attribute_condition),
            invoke(expert)),
        ...
    else(invoke(expert))
    ]).
```

It consists of a number of alternatives and an optional elsative. The alternatives contain a test, which may fail, suspend or succeed. In the last case the corresponding expert may be invoked. If tests from several alternatives succeed, an arbitrary corresponding expert is invoked, whereas the others are not further considered (don't-care committed choice; see also below and Devos 1987, however, for a suggestion of how to realize non-determinism in view of ambiguity). Only after failure of all tests is the elsative-expert executed.
Tests consist of a frame-specification and an attribute-condition. The latter constitutes the actual test on the attribute of the frame selected by "frame-specification". This frame can be referred to with

testframe in the corresponding invoked expert. One will already have noticed that the choose_alt predicate does not contain any explicit scheduling commands. Indeed, the intention is to entirely mask the program flow by a declarative reading. However, flow control remains necessary and it is realized by suspending an expert routine (or a branch in the choose_alt predicate, since the alternatives in the choose_alt may be executed in parallel), if it requires information that is not yet available. Only after this required information is filled in, does the expert-routine resume. This can cheaply be implemented using read-only unification (Shapiro 1986). Intuitively, predicates that probe for information suspend, if the variable that supplies this information is not yet instantiated. This suspension takes place during unification of the Flat Concurrent Prolog (FCP) predicate (see below), into which expert routines are compiled. Resumption occurs whenever the required variable gets instantiated. Suspension of a choose_alt branch may take place in the following cases:

(1) If the search for the testframe requires information that is not yet available, it simply suspends. As a result the frame-specification always leads to the selection of a frame in a deterministic way. Hence, explicit communication becomes possible.

(2) The attribute-test suspends until the information to be tested is available.

There is one other predicate or command that may cause suspension of an expert, viz. begin_level (a_level). The execution of an expert that specifies begin_level(a_level), is only resumed after all attributes of incorporated expert frames are specified. This filling in of attributes takes place between different expert frames on the same level (intra-level communication). With rigid rules as to which expert fills in which frame, it is possible to prove that the expert code is deadlock free. These rules will further be referred to as the deadlock avoidance rules. It suffices e.g. to prove that every frame that is at the lowest level that still contains unfilled frames, will eventually be filled in. It must then not be difficult to construct a deadlock analyser, that checks whether the deadlock avoidance rules are violated. This has not yet been further elaborated.
However, to ensure flexibility (especially from linguistic considerations) we are forced to allow inter-level communication. e.g. in sentences as "the little girl loved her toy", where "her" is level equivalent to "little", but anaphorically refers to "the little girl", which will probably be at a higher (hence, different) level than "her". In this case deadlock free code is not easy to guarantee, because of the possibility of circular waiting of experts for one another. It is our hope that we can also incorporate restricted and well-specified use of this inter-level communication in the deadlock avoidancy rules.
The system as yet designed, implements a don't-care committed-choice between the alternatives of a choose_alt predicate. This means that an arbitrary alternative that succeeds, will be chosen to determine the expert's behaviour. We are well aware of the fact that don't-care committed-choice is not always what one wants in AI applications. We merely chose this (easy) option here in order not to burden the design and implementation with one more serious problem. Two alternatives to be explored in the future are the following.
The first is intermediate between don't-care committed-choice and full non-determinism. To each alternative in the choose_alt command a priority is assigned. The alternatives are then tried out by descending priority, allowing the more likely ones to succeed first. (These priorities will often reflect frequency of occurrence of specific

linguistic structures.)   A prioritizing approach like this one will
however require more synchronisation among the alternatives of the
choose_alt to ensure a unique semantics of the command.

The second is full non-determinism.   No priorities are assigned to
alternatives, and the system is capable of undoing a wrong choice during
the analysis process.   It can go back to a choice point and try out
another alternative whose test succeeds.   A (costly) implementation
of this strategy should be based on Concurrent Prolog code (Shapiro 1986)
that contains a copy of the global environment for each alternative in the
choose_alt command.   This Concurrent Prolog code would then have to be
flattened to FCP (Codish & Shapiro 1985).

3.5 An Example Analysis

Below we present the code of some sample experts that allow the
analysis of the sentence "the little girl eats the apple".   The example is
simplified, but illustrates well the crucial elements of PEP.   First the
appropriate levels and functions are declared.   Then follows the code of
the actual experts.   Remember that *expframe* refers to the frame that is
associated with the expert and *testframe* refers to the frame that was
referred to in the alternative of the preceding choose_alt command.
"begin_frame" sets the appropriate level and "refine_function"   and
"refine_concept" do the filling in of the attributes of the specified
frame.   The lexical attribute is automatically filled in when beginning
the frame.   The example restricts itself to choose_alt commands that
only require intra-level commur._.:ation. When the sentence is read, the
corresponding experts are initialized and start to run in parallel.   The
rest of the code is self-explanatory.


```
declare(level[
    word_level
    (function[article,adjective,substantive]),
    constituent_level
    (function[action,agent,object]),
    sentence_level
    (function[])
              ]).

the :-
begin_frame(word_level),
refine_function (expframe, 'article'),
refine_concept (expframe, kind("defining")),
refine_concept (expframe, value("defined")).

little :-
begin_frame(word_level),
refine_function(expframe, 'adjective'),
refine_concept(expframe, kind("adjectival")),
refine_concept(expframe, value("young, small")).

girl :-
begin_frame(word_level),
refine_function(expframe, 'sustantive'),
refine_concept(expframe, kind("person")),
refine_concept(expframe, value("female, child_or_maiden")),
choose_alt
  ([alt(frame(minus(1),function(equal('article'))),
```

```
                    invoke(article_incorporation)),
            alt(frame(minus(1),function(equal('adjective'))),
                    invoke(adjective_incorporation)),
            else(invoke(no_incorporation))]).


        apple :- analogous to the code for girl.


        adjective_incorporation :-
        incorporate(testframe),
        choose_alt
          ([alt(frame(minus(1),function(equal('article'))),
                    invoke(article_incorporation)),
            else(invoke(no_incorporation)) ]).


        article_incorporation :-
        incorporate(testframe),
        begin_frame(constituent_level),
        refine_concept(expframe, kind("unused")),
        refine_concept(expframe, value("unused")).


        no_incorporation :-
        begin_frame(constituent_level),
        refine_concept(expframe, kind("unused")),
        refine_concept(expframe, value("unused")).


        eats :-      begin_frame(constituent_level),
        refine_function(expframe, 'action'),
        refine_concept(expframe, kind("ingest")),
        refine_concept(expframe, value("ingest_food")),
        choose_alt
          ([alt(frame(plus(3),concept(view('eatable'))),
                    invoke(eat_something)),
            else(.....................)) ]).


        eat_something :-
        refine_function(testframe, 'object'),
        incorporate(testframe),
        choose_alt
            ([alt(frame(minus(1),concept(view('person'))),
                    invoke(someone_eats_something)),
              else(....................)) ]).


        someone_eats_something :-
        refine_function(testframe, 'agent'),
        incorporate(testframe),
        begin_frame(sentence_level),
        show_solution.
```

4. A Parallel Implementation

In the last section of this paper the implementation (in a logic programming language) of all aspects of PEP discussed so far will receive a closer look. For this implementation Logix has been used, a Flat Concurrent Prolog environment (Silverman et al. 1986).

4.1 General Model Organization

The prototype realization of the PEP model allowing for correct analysis of very simple sentences (such as "The man eats", "A man eats", "Man eats") consists of an expert language (EL) to be used by the linguist when writing his experts, a precompiler that transforms the experts to FCP code and the Logix FCP compiler/emulator, the programming.environment. Linguists are offered the EL, which only contains predicates at a high level of abstraction. They may further tune the expert levels discussed earlier and the function attributes they will be using at each level to their own needs. They are only allowed to use the EL predicates according to their own specification of levels and function attributes. The EL is then precompiled to FCP. The main reason for the approach of precompiling is that flattening techniques have to be used on the predicates. These techniques are the domain of computer scientists and the linguist should not be bothered with them. (Precompiling also offers important additional advantages such as syntax checking, checking of potential deadlock, etc.; these features are still under development).

4.2 Data-Structures:
Frame Interconnection and Blackboard Information

The lexical-morphological analyzer schedules and invokes the experts corresponding to the elementary lexical units and outputs a blackboard, i.e. a matrix with slots whose columns correspond to those units and whose rows correspond to a level. Each expert has one expert frame associated with it; this expert frame fills one slot of the blackboard. In the beginning of the analysis process all frames and the blackboard contain uninstantiated slots. Experts gradually instantiate the slots. Referring to another expert's expert frame requires walking to it over the blackboard. The walk is defined in a unique way. All slots on the path should be instantiated, otherwise the walk suspends and waits for the instantiation. This is elegantly implemented using the read-only unification of the parallel Prolog versions. Slots that will never be of any use any more, are instantiated to dummy constants in order not to indefinitely block suspended walks.

5. Conclusions and further research

In this paper a further development of the procedural view of natural language analysis (NLU) as proposed by Small's Word Expert Parser has been presented. The Parallel Expert Parser tries to present a truly distributed and parallel interactive model of NLU with clearly defined experts on different levels, hierarchically conceived expert frames and rigidly restricted communication protocols.

Besides further work on the implementation and writing/testing more complex experts, the necessary model of knowledge (concept) representation that has to complete the framework is a major issue for further research. As mentioned above, Hahn (1987) has already worked on this matter, introducing generic, prototypical experts. This is not just a knowledge representation matter, but also one of integrating parallel Prolog with an object-oriented framework (Bourgois, forthcoming).

REFERENCES

ADRIAENS, G. (1986a) - Word Expert Parsing Revised and Applied to Dutch. In Proceedings of the 7th ECAI (Brighton, UK), Volume I, 222-235.

ADRIAENS, G. (1986b) - Process Linguistics: The Theory and Practice of a Cognitive-Scientific Approach to Natural Language Understanding. Phd. thesis, Depts of Linguistics and Computer Science, University of.Leuven, Belgium.

ADRIAENS, G. (1987) - A Critical Description of the Coroutine Regime Used in the Word Expert Parser. Computer Science Thesis, Dept of Computer Science, University of Leuven, Belgium.

ADRIAENS, G. & U. HAHN (eds, forthcoming) - Parallel Models of Natural Language Computation. Ablex, New Jersey.

ALTMANN, G. (1988) - Ambiguitiy, Parsing Strategies, and Computational Models. In *Language and Cognitive Processes 3 (2)*, 73-97.

ALTMANN,· G. & M. STEEDMAN (1988) - Interaction with context during human sentence processing. In *Cognition* 30, 191-238.

BERWICK. R.C. (1983) - Transformational Grammar and Artificial Intelligence: a Contemporary View. In *Cognition and Brain Theory* 6(4), 383-416.

BOURGOIS, M. (forthcoming) - The Parallel Expert Parser comes of age. Explorations in the integration of parallel Prolog and Object-oriented Knowledge Representation for Natural Language Understanding. AI Thesis, Department of Computer Science, University of Leuven, Belgium.

BRISCOE, E.J. (1987) - *Modelling Human Speech Comprehension: A Computational Approach*. Ellis Horwood, Chichester UK.

CODISH, M. & SHAPIRO, E. (1986) - Compiling OR-parallelism into AND-parallelism. Technical Report CS85-18, Department of Applied Mathematics. The Weizmann Institute of Science, Israel.

COTTRELL, G.W. (1985) - A Connectionist Approach to Word Sense Disambiguation. University of Rochester Computer Science Phd (TR-154). Rochester, New York.

COTTRELL, G.W. & S.L. SMALL (1983) - A Connectionist Scheme for Modelling Word Sense Disambiguation. In *Cognition and Brain Theory* 6 (1), 89-120.

DEVOS, M. (1987) - The Parallel Expert Parser. Realization of a Parallel and Distributed System for Natural Language Analysis In Logic Programming Languages. Engineer's Thesis, Department of Computer Science, University of Leuven, Belgium (in Dutch).

FELDMAN, J.A. & D.H. BALLARD (1982) - Connectionist Models and Their Properties. In *Cognitive Science* 6, 205-254.

GRANGER, R.H., K.P. EISELT & J.K. HOLBROOK (1986) - Parsing with Parallelism: A Spreading-Activation Model of Inference Processing During Text Understanding.

HAHN, U. (1986) - A Generalized Word Expert Model of Lexically Distributed Text Parsing. In Proceedings of the 7th ECAI (Brighton, UK), Volume I, 203-211.

HAHN, U. (1987) - Lexikalisch verteiltes Text-Parsing. Eine objekt-orientierte Spezifikation eines Wortexpertensystems auf der Grundlage des Aktorenmodells. University of Konstanz Department of Information Science PhD Thesis.

HILLIS, D. (1986) - The Connection Machine. MIT Press, Cambridge Mass.

HIRAKAWA, H. (1983) - Chart Parsing in Concurrent Prolog. Technical Report of the ICOT Research Center (TR-008). Institute for New Generation Computer Technology, Tokyo.

HIRST, G. (1983) - A Foundation for Semantic Interpretation. In Proceedings of the 21st ACL (Cambridge, Mass), 64-73.

KOWALIK, J.S. (ed) (1988) - Parallel Computation and Computers for Artificial Intelligence. Kluwer, Dordrecht The Netherlands.

MATSOMOTO, Y. (1987) - A Parallel Parsing System for Natural Language Analysis. In *New Generation Computing* 5 (1987), 63-78.

McCLELLAND, J. & RUMELHART D.E. (1986) - Parallel Distributed Processing. MIT Press, Cambridge, Mass.

POLLACK, J. & D. WALTZ (1985) - Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation. In *Cognitive Science* 9, 51-74.

SILVERMAN, W. et al. (1986) - The Logix System User Manual - Version 1.21. Technical Report CS-21 , Department of Computer Science. The Weizmann Institute of Science, Rehovot 76100, Israel.

SHAPIRO, E. (1986) - Concurrent Prolog: A Progress Report. Fundamentals of Artificial Intelligence, W. Bibel & Ph. Jorrand. Lecture Notes in Computer Science, Springer-Verlag, Berlin.

SMALL, S.L. (1980) - Word Expert Parsing: a Theory of Distributed Word-Based Natural Language Understanding. Computer Science Technical Report Series. University of Maryland Phd.

VANLEHN, K. (1984) - A Critique of the Connectionist Hypothesis that Recognition Uses Templates, and not Rules. In Proceedings of the 6th Annual Conference of the Cognitive Science Society (Boulder, Colorado), 74-80.