DECLARATIVE MODEL FOR DEPENDENCY PARSING - A VIEW INTO BLACKBOARD METHODOLOGY

Valkonen, K., Jäppinen, H., Lehtola, A. and Ylilammi, M.
KIELIKONE-project, SITRA Foundation
P.O.Box 329, SF-00121 Helsinki
Finland
tel. intl + 358 0 641 877

ABSTRACT

This paper presents a declarative, dependency constraint model for parsing an inflectional free word order language, like Finnish. The structure of Finnish sentences is described as partial dependency trees of depth one. Parsing becomes a nondeterministic search problem in the forest of partial parse trees. The search process is able to solve also ambiguities and long-distance dependencies. Parsing is controlled by a blackboard system. A working parser for Finnish has been implemented based on the model.

1 INTRODUCTION

The development of our computational model for dependency parsing has progressed in three parallel and interrelated phases:

(1) The development of a perspicuous high level grammar specification language which grasps well regularities and idiosyncracies of inflectional free word order languages.
(2) The acquisition of a grammar using that language as the description media.
(3) The development of a parsing strategy and a compiler for the specification language.

In our first approach the parsing process is described as a sequence of local decisions (Nelimarkka et al. 1984). A pair of adjacent structures of an input sentence is connected if there exists a valid binary dependency relation between them. Binary relations are boolean expressions of the morphological and syntactic restrictions on argument structures. In that first version dependency sructures were modelled procedurally with finite two-way automata (Lehtola et al. 1985). Grammar descriptions turned out to be complicated to handle, and due to purely local decisions some global phenomena, such as long-distance dependencies, were not analyzed.

A new grammar description formalism and computational method was developed: a declarative high level language FUNDPL (Jäppinen et al. 1986) for a grammar, and an underlying blackboard-based parsing method (Valkonen and Lehtola, 1986). Recently, we have augmented the dependency parsing model to cover also long-distance dependencies. According to the augmented model we have implemented a blackboard-based dependency parser ADP (Augmented Dependency Parser). In this paper we shortly describe our model and focus on the parsing strategy. For the grammar development environment and the compilation of the high level description language, see Lehtola et al. (1985, 1986).

Our parsing method belongs to the class of constraint systems: a user specifies the constraints holding in the problem domain, and a goal for the computation. The interpreter must search for the goal. The result follows indirectly from the search process. In our model binary relations specify constraints on argument structures. The goal is to find a matching local environment description for each word of an input sentence. As a side effect of the recognition corresponding partial dependency trees are built. The partial dependency trees are linked into a parse tree covering the whole sentence (Figure 1).

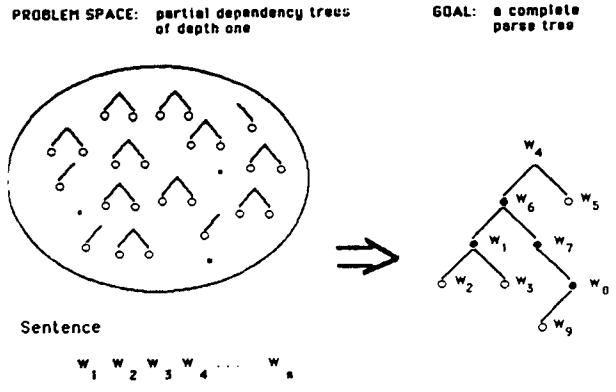

Figure 1. Parsing as a search process in a forest of partial dependency trees.

2 GRAMMAR DESCRIPTION

For the development of a grammar notation idiosyncracies of the object language had to be observed. Finnish is a relatively free word order language. The syntactic-semantic knowledge is often expressed in the inflections of the words. Furthermore, the parser was needed to work as a practical tool for real production applications, so the process of parsing was taken as a starting point instead of sentence generation.

A grammar description consists of four parts:
(1) Type definitions: linguistic properties, features and categories.
(2) A lexicon for associating features with words.
(3) Binary dependency relations that may hold between regents and their dependents.
(4) Functional schemata for defining the local environments of regents.

2.1 Type definitions

In the type definition part a grammar writer defines the types and their values used in a grammar description. This corresponds to the classification of linguistic properties. There are three kinds of types: CATEGORIES, FEATURES and PROPERTIES. In addition to this the structure of the lexical entries is described in this part.

CATEGORY statement assigns names in hierarchies. For example, a category SyntCat for word classes could be defined as

```
(CATEGORY: SyntCat
        < (Word)
             (Noun ! Word)
               (Proper ! Noun)
               (Common ! Noun)
             (Pronoun ! Word)
               (PersPron ! Pron)
               (DemPron  ! Pron)
               (IntPron  ! Pron)
               ...
```

In a FEATURE statement a feature name and its values are defined. Values can be mutually exclusive: adding of the complement value automatically destroys the old value.

```
(FEATURE: SyntFeat
        < (Locative) ;a name of a place
          (InfAttr)  ;a noun, that may have an
                       infinitivial attribute
          (CountMeasure) ;a countable measure noun
          ...
```

PROPERTY values are like FEATURES except that they may have default values. For example:

```
(PROPERTY: Polar < ( Pos )  Neg >)
```

In this type definition polarity is positive by default.

2.2 Lexicon

The parser is preceded by a morphological analyzer (Jäppinen and Ylilammi 1986). The morphological analyzer produces for each word its morphological interpretation including lexical information. The parser associates default features for words. Those words which have idiosyncratic features, as all verbs do, are in the parser's lexicon. Some example entries of the parser's lexicon:

```
METRI     (Common (SubstMeasure))
HELSINKI (Proper (Locative))
AJATELLA (TrProcV (InfObj PartisObj))
```

"Metri" (meter) is a measure unit for common nouns. "Helsinki" is a proper noun and a name of a place. "Ajatella" (to think) is a transitive verb that may have infinitivial or participle objects.

2.3 Binary dependency relations

The dependency parsing model aims at providing analyzed sentences with their dependency trees. According to this approach two elements of a sentence are directly related in a dependency relation if one depends on another. The two elements are called the regent R (or head or governer) and the dependent D (or modifier). Binary relations define all permitted dependency relations that may exist between two words in Finnish sentences. For example, the binary relation Subject is the following boolean expression of the morphological and syntactic features of a finite verb and its nominal subject:

```
(RELATION: Subject  (D := Subject)
  ((R = Verb Act
        < (< Ind Cond Imper Pot IIpartis > (PersonP D)(PersonN D)
           - Negative - Auxiliary)
          (Auxiliary IIpartis Nom - Negative)
            (Negative < (Imper Pr < (S 2P) Neg >)
                        (Cond Pr S 3P) (Pot Pr Neg)
                       (IIpartis Nom)> - Auxiliary)>)
    (D = PersPron Nom))...
```

R must be an active verb. Further restrictions for it

appear within angle brackets that indicates a disjunction. Negation is expressed by "-". (PersonP D) (PersonN D) indicates an agreement test. D must be a personal pronoun in nominative case in this fragment.

In our computational model words of an input sentence appear as complexes of their morphological, syntactical and semantic properties. We call this complex a constituent. If a binary relation holds between R and D, they are adjoined into a single constituent. This is what we mean by a functional description. It can be stated formally as mapping

f(R,D) -> R'

where R' stands for the regent R after that it has bound D. Function f is defined by the corresponding binary relation. This function abstraction should be distinguished from grammatical functions, even though in our grammar specification dependency relations also estimate grammatical functions.

2.4 Functional schemata

In functional schemata the local environment of a regent is described by dependency functions. Functional schemata can be seen as partial dependency tree descriptions. A simplified schema for verb phrases, when a regent is a transitive verb and it is preceeded by a negative auxiliary verb, could be defined as

```
(SCHEMA:    NegTransVerb
 WHEN       (AND (R = ProcVerb Act Transitive)
                 (LEFT = Auxiliary Negative))
 FUNCTIONS  (MULTIPLE Adverbial)
            (OBLIGATORY Negation Subject Object)
            (LEFT Negation Subject Object Adverbial)
            (RIGHT Object Subject Adverbial)
 MARK       (R := VerbP))
```

This schema is able to recognize and build, for instance, partial dependency trees shown in Figure 2.

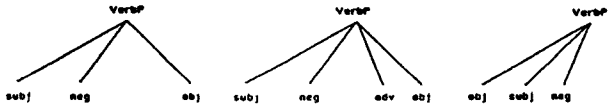

Figure 2. Example trees built by a schema NegTransVerb.

There are three parts in the simplified schema NegTransVerb: **WHEN**, **FUNCTIONS** and **MARK**. WHEN part describes features for the regent and its context. FUNCTIONS part describes the dependents for the regent. **MULTIPLE** clause indicates which dependents may exist multiple times. **OBLIGATORY** names obligatory dependents. **LEFT** and **RIGHT** give the structure of the left and right context of the regent.

The free word order is allowed by default because of the particular interpretation of the clauses LEFT and RIGHT. The definition only indicates which dependents exist in the named context, not their mutual order. All the permutations are allowed. There is also means of fixing word ordering. **ORDER** clause indicates mutual ordering of dependents. For example, a grammar writer may define for the simple NP's

(ORDER AdjAttr GenAttr R RelAttr)

For this particular regent the most immediate left neighbour must be a genetive attribute. The next to that is an adjective attribute. The right neighbour is a relative clause.

For **long-distance dependencies** the local decision strategy must be augmented. The binding of long-distance dependents has two phases: the recognition and the actual binding.

In transformational grammar, long-distance dependencies are dealt with by assuming that in the deep structure the missing word is in the place it would be in the corresponding simple sentence. It is then moved or deleted by a transformation. The essential point is that long-distance dependency is caused by an element which has moved from the local environment of a regent to the local environment of another regent. Hence a moved element must be recognized by the functional schema associated with that latter regent. The binding, then, is done later on by the schema of the former regent.

In the recognition phase the long-distance dependents are recognized and bound "away" (captured), so that the current regent can govern its environment. After this capture the possible long-distance dependent remains waiting for binding by another schema.

Capturing dependency functions are marked in the CAPTURE clause:

(CAPTURE DistantMember)

The dependency function DistantMember is general enough to capture all possible long-distant dependents. For the actual binding of long-distance dependents, one must mark in the clause **DISTANT** the dependents which may be distant:

(DISTANT Object)

3 BLACKBOARD-BASED CONTROL FOR DEPENDENCY PARSING

Blackboard is a problem-solving model for expert systems (Hayes-Roth et al. 1983, Nii 1986). We have adopted that concept and utilized it for parsing purposes. Our blackboard model application is rather simple (Figure 3).

There are three main components: a blackboard, a control part and knowledge sources. The blackboard contains the active environment description for a regent. According to the structural knowledge in that environment description corresponding partial parse tree is built in the blackboard. Also all other changes in the state of computation are marked in the blackboard.

Functional schemata and binary dependency relations are independent and separate knowledge sources; no communication happens between them. All data flow takes place through the blackboard. Which module of knowledge to apply is determined dynamically, one step at a time, resulting in the incremental generation of partial solutions.

In functional schemata a grammar writer has described local environments for regents by dependency functions. The schemata are compiled into an internal LISP-form. At a time, only one of the schemata is chosen as an active environment description for the current regent. The activated schema is matched with the environment of the regent by binary relation tests. The binary relations respond to the changes in the blackboard according to the structural description in the active schema and the properties of the regent and dependent candidates. At the same the partial dependency tree is built by corresponding dependency function applications. When a schema has been fully matched and the active regent bound to its dependents through function links, the local partial dependency parse is complete.

A scheduler for knowledge sources controls the whole system. It monitors the changes on the blackboard and decides what actions to take next. The scheduler employs a finite two-way automaton for recognition of the dependents.

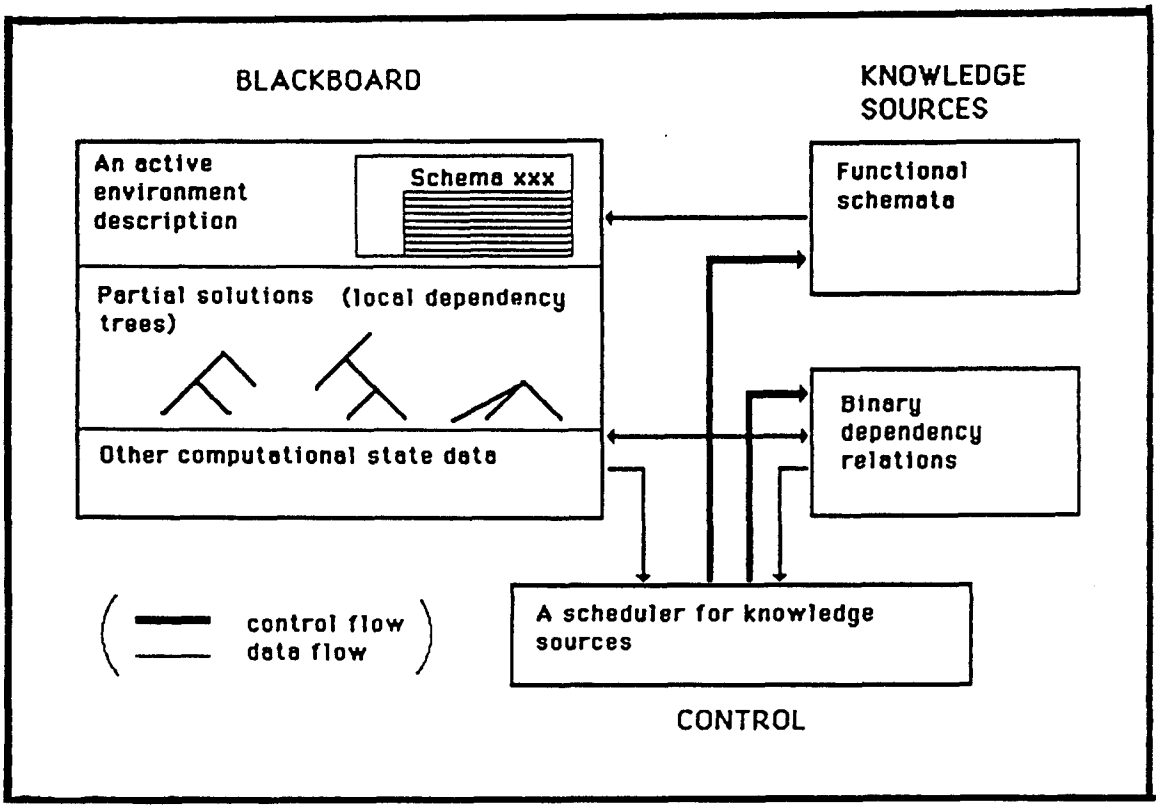

Figure 3. A blackboard model for dependency parsing.

3.1 The blackboard-based control strategy for dependency parsing

For the formal definition of the parsing process we describe the input sentence as a sequence (c(1),c(2),...,c(i-1), c(i), c(i+1),...,c(n)) of word constituents. With each constituent c(i) there is associated a set (s(i,1),...,s(i,m)) of functional schemata. The general parsing strategy for each word constituent c(i) can be modelled using a transition network. During parsing there are five possible computational states for each constituent c(i):

S1 The initial state. One of the schemata associated with c(i) is activated.

S2 Left dependents are searched for c(i).

S3 c(i) is waiting for the building of the right context.

S4 Right dependents are searched for c(i).

S5 The final state. The schema associated with c(i) has been fully matched and becomes inactive. c(i) is the head of the completed (partial) dependency tree.

At any time, only one schema is active, i.e. only one constituent c(i) may be in the state S2 or S4. Only a completed constituent (one in the state S5) is allowed to be bound as a dependent for a regent. There may be a number of constituents simultaneously in the state S3. We call these pending constituents (implemented as a stack PENDING).

The parsing process starts with c(1) and proceeds to the right. Initially all constituents c(1),..,c(n) are in the state S1. A sentence is well formed if in the end of the parsing process the result is a single constituent that has reached the state S5 and contains all other constituents bound in its dependency tree. For each constituent c(i) the parsing process can be described by the following five steps. Parsing begins from the step 1 with i,k = 1.

1) A schema candidate s(i,k) associated with c(i) is activated, i.e. the constituent c(i) takes the role of a regent. Following the environment description in s(i,k), dependents for c(i) are searched from its immediate neighbourhood. Go to the step 2 with j = i-1.

2) The search of left dependents. There are two subcases:

2a) There are no left neighbours (j = 0), none is expected for c(i), or c(j) (j < i) exists and is in the state S3.
Go to the step 3 with j = j+1.

2b) c(j) (j < i) exists and is in the state S5. Binary relation tests are done. In the case of a succes the mapping f(c(i), c(j)) -> c(i)' takes place. Repeat the step 2 with j = j-1 and c(i) = c(i)'.

3) Building the right context of the regent. There are two subcases:

3a) There are no right neighbours (j > n) or none is expected for c(i). Go to the step 5.

3b) c(j) (j > i) exists. Go to the step 1 with c(i) = c(i+1) and PENDING = push (c(i), PENDING).

4) The search of right dependents. Binary relation tests are done. In the case of succes the mapping f(c(i), c(j)) -> c(i)' takes place. Repeat the step 3 with j = j+1 and c(i) = c(i)'.

5) The final state. There are two subcases:

5a) The environment description has been matched. If there remains no unbound c(j)'s (j < i or j > i) the sentence is parsed. If c(i+1) exists go to the step 1 with i = i+1. If c(i+1) doesn't exist or the steps followed previous case returned a failure, go to the step 4 with c(i) = pop (PENDING).

5b) The environment description has not been matched. Return a failure.

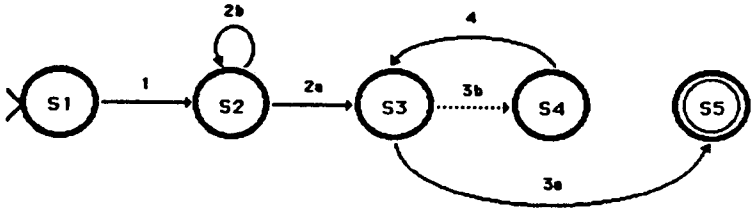

Figure 4. The transition network model of the control strategy.

3.2 The implementation of the control strategy

The control system has two levels: the basic level employs a general two-way automaton and the upper level uses a blackboard system. There is a clear correspondence between the grammar description and the control system: the two-way automaton makes local decisions according to the binary relations. These local decisions are controlled by the blackboard system which utilizes the environment descriptions written in the schemata. This two-level control model has certain advantages. The two-way automaton is computationally efficient in local decisions. On the other hand, the blackboard system is able to utilize global knowledge of the input sentence.

Chronological backtracking

To account for ambiguities there are three kinds of backtracking points in the control system. Backtracking may be done in regard to choice of dependency functions, homographic word forms, or associated schemata. Backtracking is chronological. In our system a constituent c(i) may contain several different morphotactic interpretations of a word form. Function backtracking takes place if there are several possible binary relations between a given constituent pair. The preconditions of the schemata may allow multiple schema candidates for a given constituent. All alternatives are gone through one by one, if necessary, in chronological backtracking. As a result, the system may perform an exhaustive search and produce all possible solutions.

Register for long-distance dependencies

The recognition of possible lond-distant dependencies is done by the capture function. An element is bound as a possible "distant member" in the context where the capture function fires. An element is also moved to the special register for a set of distant elements. The actual binding is done by the distant function from another schema. In chronological backtracking also distant bindings are undone.

The strategy of local decisions controlled by global knowledge of the input sentence yields a strongly data-driven, left-to-right and bottom-up parse whereby partial dependency trees are built proceeding from middle to out.

3.3 EXAMPLES

To visualize our discussion, a functional schema IntrImpNegVP is described in Figure 5. A grammar writer has declared in WHEN-part that R must be a transitive process verb in active tense and imperative mood. In its left context there must be a negative verb in imperative mood and of the lexical form "EI" ("NOT"). There is one obligatory dependency relation NegVerb. Adverbials may exist multiple times. A grammar writer has written in clauses LEFT and RIGHT the left and right context binary relations of the regent. After the schema has fully matched, the regent is marked VerbP and features PersonN and PersonP of the dependent recognized as NegVerb are marked for the regent.

```
(SCHEMA:      IntrImperNegVP
   WHEN       (AND
                 (R = ProcVerb Act Imper (NOT VerbTr))
                 (Left = 'EI Imper))
   FUNCTIONS (OBLIGATORY NegVerb)
              (MULTIPLE Adverbial)
              (LEFT NegVerb Adverbial Connect)
              (RIGHT Adverbial)
   MARK       (R := VerbP (RecNegVerb (PersonP PersonN)))
)
```

Figure 5. A functional schema IntrImperNegVP

A full trace of parsing the sentence "Älä eksy metsässä!" (Don't get lost in a forest) appears in Figure 6. Parsing starts from the left (an arrow). Next line indicates the selected schema and dependents that are tested. The first word "älä" is identified as a negative imperative verb with no dependents (schema DummyVP ok). The imperative verb "eksy" (to get lost) is then tried by the schema IntrImperNegVP. The binary relation NegVerb holds between the two verbs, and the corresponding dependency function adjoins them. The other functions fail. Dependents are searched next from the right context. The control proceeds to the word "metsässä" (forest). For that word no dependents are found and the system returns to the unfinished regent "eksy". The schema IntrImperNegVP has only two relations remaining: Connect and Adverbial. The word "metsässä" is bound as an adverbial. The schema has been fully matched and the input sentence is completely parsed.

```
> älä eksy metsässä!
MORFO:
((((("älä" EI Verb Act Imper Pr S /2P/)))
 ((("eksy" EKSYÄ Verb Act Imper Pr S /2P/)))
 ((("metsässä" METSÄ Noun SG In)))
 ((("!" EXCLAMATION))))

=> (älä) (eksy) (metsässä)
Schema: DummyVP nil
DummyVP ok
(älä) => (eksy) (metsässä)
Schema: IntrImperNegVP (NegVerb Adverbial Connect)
NegVerb ok
Adverbial failed
Connect failed
((älä) eksy) => (metsässä)
Schema: TrivialSP (DefPart R)
DefPart failed
TrivialSP ok
  returning to unfinished constituent...
((älä) eksy) <= (metsässä)
Schema: IntrImperNegVP (Connect Adverbial)
Adverbial ok
IntrImperNegVP ok
=> ((älä) eksy (metsässä)) PARSED

The parse took 0.87 seconds CPU-time on VAX-11/751.
```

Figure 6. An example of parsing.

The second example shows how our parser solves the following sentence (adopted from Karttunen, 1986b) which has a long-distance dependency:

En minä tennistä aio ruveta pelaamaan.
not I tennis intend start play
I do not intend to start to play tennis.

The object of the subordinated infinitivial clause ("tennistä") has been raised in the main clause thus creating a gap. The parse tree of the sentence is in Figure 7.

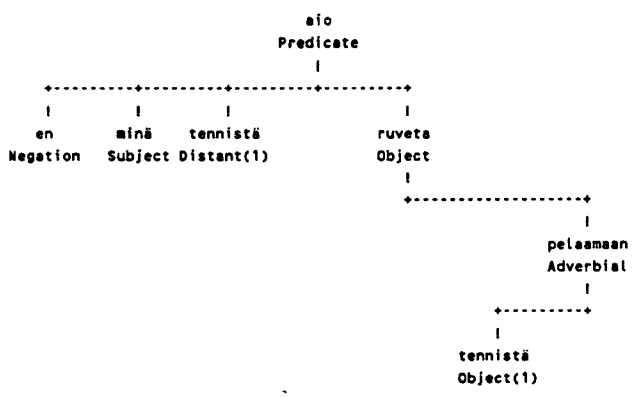

Figure 7. An example of a long-distance dependency.

In the parsing process the schema NO-VP has matched the environment of the verb "aio" (intend) and the schema O-LocativeVP of the verb "pelaamaan" (play).

```
(SCHEMA:    NO-VP
 ASSUME     (R := Negative)
 FUNCTIONS (OBLIGATORY Object Negation)
           (MULTIPLE Adverbial DistantMember)
           (LEFT Auxiliary Negation Object Adverbial Connector)
           (RIGHT Object Adverbial Comma)
           (CAPTURE DistantMember)
 CLAUSE_READY
 CHECK      (VerbObjCongr Negation Object)
 MARK       (R := ProcVP Predicate (Negation (PersonP PersonN)))
)

(SCHEMA:    O-LocativeVP
 FUNCTIONS (OBLIGATORY Object)
           (MULTIPLE Adverbial DistantMember)
           (RIGHT Object Adverbial)
           (LEFT Object Adverbial)
           (CAPTURE DistantMember)
           (DISTANT Object Adverbial)
 MARK       (R := LocativeVP Predicate)
)
```

The schema NO-VP has captured the word "tennistä" as a DistantMember. The schema O-LocativeVP has later on bound it as a removed Object.

4 COMPARISON

The notion of unification has recently emerged as a common descriptive device in many linguistic theories like FUG, PATR-II and HPSG (Shieber 1986). Another popular approach has been to apply attribute grammars originally developed as a theory for formal languages (Knuth 1968). LFG and DCG can be viewed as attribute grammar systems. The trend has been towards strictly declarative descriptions of syntactic structure. Syntactic rules are often expressed in the form of complex feature sets.

Our ADP system also uses features, but differs both from the unification-based approach and attribute grammar approach. The basic difference is, of course, that there is neither unification nor correspondence to attribute grammars in our system. We use a pattern matching via binary relation tests. Through blackboard approach we have gained a flexible control. Blackboard system can conveniently take into account global knowledge of the sentence. In our model dependents become "hidden" from further processing once they have been found. A regent solely represents the constituents hanging below. This makes the parsing process simpler as the number of constituents decreases during parsing. There are, however, some cases where some information must be raised from the dependent to the regent (e.g. from conjuncts to the conjunction), so that the regent could represent the whole constituent.

5 CONCLUSION

In our system linguistic knowledge and processing mechanisms are separated. Structural information of the functional schemata is interpreted by the blackboard scheduler as control knowledge, according to which dependencies are searched. The difference between local and global decisions is clearly separated. Local decisions controlled by global knowledge of the input sentence has made it possible to find solutions for problems that are difficult to solve in traditional parsing systems. ADP finds all solutions for an ambiguous sentence. Augmented search process covers long-distance dependencies as well.

Different criteria have been expressed for grammar formalisms (Winograd 1983, Karttunen 1986a): perspicuity, nondirectionality, correspondence with meanings, multiple dimensions of patterning, order-independency, declarativeness and monotonicity. Our model rates well in most of these criteria. Perspicuity, correspondence with meanings and declarativeness are satisfied in the way the functional schemata describe local environments for regents. The functional description is monotonic and allows multiple dimensions of patterning.

There is a process of parsing as a starting point in the grammar specification, so it lacks nondirectionality. The weakest point is the order-dependent control mechanism, albeit the grammar description is order-independent. Plans for the general, order-independent control strategy have been done.

ADP has been implemented in FranzLisp. Experiments with a non-trivial set of Finnish sentence structures has been performed on VAX 11/751 system. An average time for parsing a six word sentence is between 0.5 and 2.0 seconds for the first parse. At the moment the grammar description contains common sentence structures quite well. There are 66 binary relations, 188 functional schemata and 1800 lexicon entries. The lexicon of the morphological analyzer contains 35 000 words.

ACKNOWLEDGEMENTS

This research has been supported by SITRA Foundation.

REFERENCES

Hayes-Roth, F., Waterman, D. and Lenat, D. 1983 Building Expert Systems. Addison-Wesley Publishing Company, Reading.

Jäppinen, H. and Ylilammi, M. 1986 Associative Model of Morphological Analysis: an Empirical Inquiry. Computational Linguistics, Volume 12, Number 4, October-December 1986, pp. 257-272.

Jäppinen, H., Lehtola, A. and Valkonen, K. 1986 Functional Structures for Parsing Dependency Constraints. Proceedings of COLING86/ACL, Bonn, pp. 461-463.

Karttunen, L. and Kay, M. 1985 Parsing in a free word order language. In Dowty, Karttunen and Zwicky (Eds.), Natural Language Parsing, Cambridge University Press.

Karttunen, L. 1986a The Relevance of Computational Linguistics. A paper presented at the Conference on Finnish Linguistics.

Karttunen, L. 1986b Radical Lexicalism. A paper presented at a Conference on Alternative Conceptions of Phrase Structure, New York.

Knuth, D. 1968 Semantics of Context-Free Languages. Mathematical Systems Theory 2(1968a), pp. 127-145.

Lehtola, A., Jäppinen, H. and Nelimarkka, E. 1985 Language-based Environment for Natural Language Parsing. Proceedings of the 2nd European Conference of ACL, Geneve, pp. 98-106.

Lehtola, A. and Valkonen, K. 1986 Knowledge Representation Formalisms and Metadescriptions for the Interpretation of Finnish. Proceedings of the Third Finnish Symposium on Theoretical Computer Science, pp. 64-87.

Nelimarkka, E., Jäppinen, H., and Lehtola, A. 1984 Parsing an inflectional free word order language with two-way finite automata. Proceedings of the 6th European Conference on Artificial Intelligence, Pisa, pp. 167-176. Also in O'Shea, T. (Ed.), Advances in Artificial Intelligence, North-Holland.

Nii, H. 1986 Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. The AI Magazine, Summer 1986, pp. 38-53, August 1986, pp. 82-106.

Shieber, S. 1986 An Introduction to Unification-Based Approaches to Grammar. CSLI Lecture Notes Series, No. 4.

Valkonen, K. and Lehtola, A. 1986 Blackboard Control for Dependency Parsing. A paper presented in Nordisk seminar om maskinoversattelse, 9.-11.10 1986, University of Copenhagen, 12 p. (in print).

Winograd, T. 1983 Language as a Cognitive Process. Volume I: Syntax. Addison-Wesley.