

Esa Nelimarkka, Harri Jäppinen,  
and Aarno Lehtola,  
Helsinki University of Technology  
Espoo, Finland

## A COMPUTATIONAL MODEL OF FINNISH SENTENCE STRUCTURE<sup>1</sup>

### Introduction

The present paper propounds an outline of a computational model of Finnish sentence structures. Although we focus on Finnish we feel that the ideas behind the model might be applicable to other languages as well, in particular to other inflectional free word order languages.

A parser based on this model is being implemented as a component of a larger system, namely a natural language data base interface. There it will follow a component of morphological analysis (see Jäppinen et al [8]); hence, throughout the present paper it is assumed that all relevant morphological and lexical information is computationally available for all words in a sentence. Even though we have a data base application in mind, sentence analysis will be based on general linguistic knowledge. All application dependent inferences are left to subsequent modules which are not discussed here.

### The linguistic foundations

We shall freely borrow ideas of Anderson [1], [2], Tarvainen [11] and Pajunen [9], and Fillmore [3], [4] and Siro [10] concerning dependency and case grammars. We shall use the latter grammar to introduce semantics into syntax and the former to give a basis for a functional syntax where the subordinate dependency relations are specified with formal binary relations. The structure which these grammars will impose on sentences is a dependency-constituency hybrid structure with labelled dependants, similar to the sister-dependency structure of Hudson [7]. We shall briefly explicate and reason our choices.

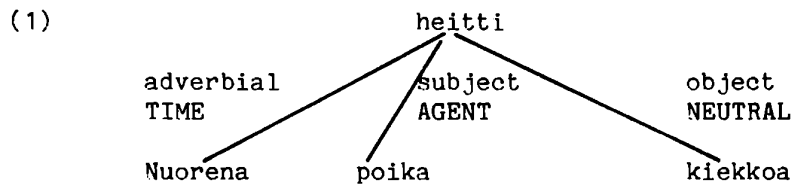
-----  
1 This research is supported by SITRA (Finnish National Fund for Research and Development), PL 329, 00120 Helsinki 12, Finland

Firstly, Finnish is a "free word order" language in the sense that the order of the main constituents of a sentence is relatively free. Variations of word order configurations convey thematical and discursional information. Hence, we must be ready to meet sentences with seemingly odd word orders which in a given context, however, are quite natural.

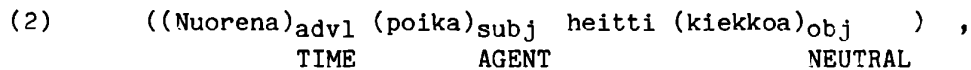
The computational model should acknowledge this state of affairs and be adjusted to cope efficiently with it. This demands a structure within which word order variations can be conveniently described. An important case in point is to avoid discontinuities in the structure caused by transformations.

We argue that a dependency-constituency hybrid structure induced by a dependency grammar meets the requirements. This structure consists of part-of-whole relations of constituents and labelled binary dependency relations within a constituent.

The sentence "Nuorena poika heitti kiekkoa" ("As young, the boy (used to) throw the discus"), for example, will be given the structure



or, a linearized equivalent with labelled parentheses,



where parentheses indicate constituent boundaries and, within each constituent, the word without parentheses is the head. (To be more precise, an inflected word appears as a complex of all its syntactic, morphological and semantic properties. Hence, our structure representation is a tree with nodes which are labelled with complex expressions.)

The advantage of such structures lies in the fact that many word order varying transformations can now be localized to a permutation of the head and its dependants in a constituent. As an example we have the permutations

- (2') ((Poika)<sub>subj</sub> heitti (nuorena)<sub>advl</sub> (kiekkoa)<sub>obj</sub>)  
 (2'') (Heittikö (poika)<sub>subj</sub> (nuorena)<sub>advl</sub> (kiekkoa)<sub>obj</sub>)  
 (2''') ((Kiekkoako)<sub>obj</sub> (poika)<sub>subj</sub> heitti (nuorena)<sub>advl</sub>)

Having reduced the depth of structures (by having a verb and its subject, object, adverbials etc. on the same level) we bypass many discontinuities that would have appeared in a deeper structure as a result of rising transformations.

The second argument for our choices is the well acknowledged prominent role of a finite verb in regard to the form and meaning of a sentence. The meaning of a verb (or a word of other categories as well) includes knowledge of its deep cases, and the choice of a particular verb to express this meaning determines to a great extent what deep cases are present on the surface level and in what functions.

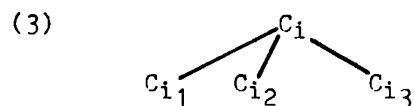
Moreover, due to the free word order of Finnish, the main means to indicate the function of a word in a sentence is the use of surface case suffixes, and very often the actual case depends not only on the intended function or role but on the verb as well.

Both of these arguments speak well for a combination of dependency and case grammars. We claim that such a combination can be put into a computational form, and that the resulting model is one which efficiently takes advantage of the central role of the constituent head in the actual parsing process. We shall outline below how this can be done with finite 2-way tree automata.

#### Specification of dependency structures

In a dependency approach the description of a constituent associates the head with a specification of its dependants and their order.

Hays [5] used the notation

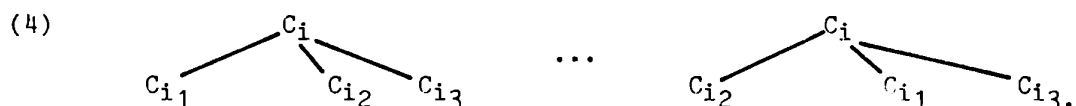


or its linear equivalent,

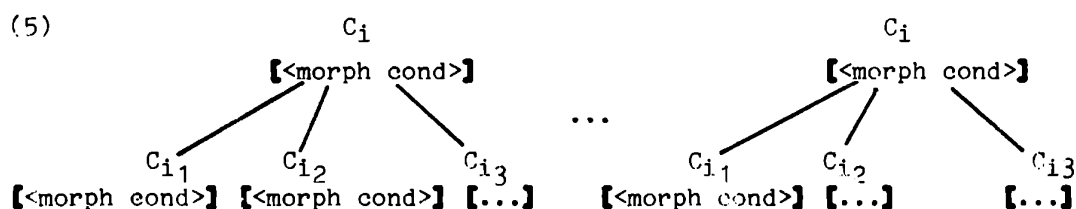
(3')  $C_i(C_{i_1} * C_{i_2} C_{i_3}),$

to describe the dependants of a word in a category  $C_i$ .

Such a formalism is not suitable for our purposes. Firstly, due to free word order, one would either have to postulate permutation transformations or directly proliferate the "frame" (3) to the different word order configurations, to



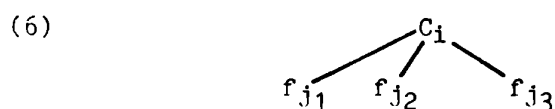
for example. Secondly, in case of an inflectional language, the definition of dependants with restrictions on mere categories would not suffice. Morphological conditions on the head and its dependants are needed, too. This would lead to another kind of frame multiplication:



where the different frames describe mutually exclusive morphological situations. Thirdly, we wish to know the syntactic function of a dependant, not only its place, category, number, case etc. We shall now show how to overcome these difficulties.

#### A 2-way finite tree automaton -model

We begin with a unification of the frames (5) by abstracting the laws that govern the restrictions on the head and its dependants in such a sequence. These abstractions, which we call functions, are defined as formal binary relations on the set of inflected words (which are considered as complexes of all their relevant properties). It is no surprise that these abstract functions will correspond to — and are named after — the traditional parsing categories like subject, object, adverbial, genitive attribute etc. After the adoption of these abstract functions the sequence (5) reduces to a single "relation frame"



or to a linear expression

$$(6') \quad C_1(f_{j_1} * f_{j_2} f_{j_3}) ,$$

where the  $f_j$ 's stand for functions.

We are still left with the permutational variants

$$(7) \quad \begin{array}{c} C_1 \\ / \quad \backslash \\ f_{j_1} \quad f_{j_2} \quad f_{j_3} \end{array} \quad \dots \quad \begin{array}{c} C_1 \\ / \quad \backslash \\ f_{j_2} \quad f_{j_1} \quad f_{j_3} \end{array} .$$

Our final step is to combine these relation frames with a structure building 2-way finite tree automaton. At this point we also give a computational form to the idea of the dominance of the head of the constituent with respect to its form and meaning.

Recall that a standard 2-way finite automaton consists of a set of states, one of which is a starting state and some of which are final states, and of a set of transition arcs between the states. Each arc recognizes a word, changes the state of the automaton and moves the reading head either to the left or right. (Cf. Hopcroft-Ullman [6].)

We modify this standard notion to recognize left and right side dependants of a word - obligatory, facultative valencial dependants and free ones - starting from the most immediate neighbour.

Instead of recognizing words (or a word categories) we make these automata recognize functions, i.e. occurrences of abstract relations between the postulated head and its left or right neighbour. Secondly, in addition to recognition we make the "arcs" build the structure determined by the observed function, e.g. attach the neighbour as a dependant, label it in agreement with the function and its interpretation etc.

#### An illustration

We approximate the syntactic function "object" of a finite verb with the two productions

(8)

```
RELATION:Object
  (RecognObj -> (D:=Object) (C:-D) (DELETE D))

RELATION:RecognObj
  ((R=+transitive -nominal) (D=+nominal -sentence)
   -> ((D=Partitive) -> (D=Plural);
       ((D=Singular) -> (D=-countable);
          (D=+countable) (R=<Negative
            +pobj>)))));

  ((D=Accusative) (R=Positive));
  ((D=Nominative) (R=Positive)
   -> (D=Plural);
       ((D=Singular) (R=Active
         <Indicative
          Conditional
          Potential
          Imperative 3P>)))));

  ((D=Genitive Singular) (R=Positive
    <Passive
     Active Imperative <1P 2P>)))));

$
```

The relation "RecognObj" in the above production form is a kind of boolean expression of the morphological restrictions on a finite verb and its nominal object. The relation "Object", on the other hand, after a successful match of these conditions, labels the postulated dependant (D) as "object" and attaches it to the postulated regent (R).

The fragment

(9)

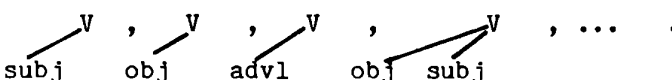
```
AUTOMATON:Verb

STATE:?V
  ((D=+phrase) -> (Subject -> (C:=?VS LEFT));
                  (Object -> (C:=?VO LEFT));
                  (Adverbial -> (C:=?V LEFT));
                  (GenSubj -> (C:=VS? RIGHT));
                  (SentAdv1 -> (C:=?V RIGHT));
                  (T => (C:=V? RIGHT)));
  ((D=-phrase) -> (C:=V? RIGHT))

STATE:?VS
  ((D=+phrase) -> (Object -> (C:=?VSD LEFT));
                  (Adverbial -> (C:=?VS LEFT));
                  (SentAdv1 -> (C:=VS? RIGHT));
                  (T -> (C:=VS? RIGHT)));
  ((D=-phrase) -> (C:=VS? RIGHT) (BuildPhraseOn RIGHT));

  .
  .
  .
```

of a verb automaton recognizes and builds, for example, partial structures like

(10) 

### Parsing with a sequence of 2-way automata

So far we have shown how to associate a 2-way automaton to a word via its syntactic category. This gives a local description of the language. We argue that with a few simple control instructions we can make these local automata activate each other and actually parse an input sentence.

An unfinished parse consists of a sequence of constituents, which may be complete or incomplete. Further, each such constituent is associated with an automaton in some state and reading position. Now, the question is how to activate these automata. At any time, exactly one of the automata is to be active, i.e. trying to find dependants to the constituent head in the immediate neighbourhood.

Only a completed constituent (one featured as +phrase) is to be matched as a dependant. To start the completion of an uncomplete constituent the control has to be moved to this constituent. This is done with a push-like control operation "BuildPhraseOnRight" which deactivates the current automaton and activates the neighbour next to the right (see the illustration above).

On the other hand, a tree in a final state will be labelled as a +phrase (along with other relevant labels such as †sentence, †nominal, †main etc.). Pop-like control operations "FindRegOnLeft" and "FindRegOnRight" deactivate the current constituent (i.e. the corresponding automaton) and activate the leftmost or rightmost constituent, respectively.

We claim that such simple "local control" yields a strongly data driven bottom-up and left-to-right strategy which has also top-down features in the form of expectations on lacking dependants.

We use heuristic rules to reduce the number of alternative postulates in the course of parsing. For example, we might include the production

$$\begin{aligned} ((R1 = ',')(R2 = 'että')(C = +trans +cogn) \rightarrow (C := SV?Sent0 \text{ RIGHT}) \\ \text{(BuildPhraseOnRIGHT)}) \end{aligned}$$

in the state VS? of the verb automaton to recognize an evident forthcoming sentence object of a cognitive verb and to set the verb to the state SV?Sent0 to wait for this sentence.

### Comparison

Our model, as we have shown, consists of a collection of finite transition networks which activate each other with pop- and push-like control operations. How does our approach then differ from, say, Woods' ATN-formalism, which seems to have similar characteristics?

One difference is the use of 2-way automata instead of 1-way automata. There are also other major differences. ATN-parsers seem to use pure constituent structures containing non-terminal auxiliary categories (VP, NP, AP...) without explicit use of dependency relations within a constituent. In our dependency oriented model non-terminal categories are not needed, and a constituent is not postulated until its head is found. In fact, each word collects actively its dependants to make up a constituent where it is the head.

A further characteristic of our model is the late postulation of a function or a semantic role. Trees are built blindly without any predecided purpose. The function or semantic role of a constituent is not postulated until some earlier or forthcoming neighbour is activated to recognize dependants of its own. Thus, a constituent just waits to be chosen into some function.

The above feature explains why no registers are needed in our approach.

### Conclusions

We have outlined a model of Finnish which is based on 2-way structure building transition networks. We have, as the above illustration exhibits, specified our model with a kind of production-rule formalism.

A compiler which compiles such descriptions into LISP is under construction. This LISP-code is further compiled into a directly executable code so that no interpretation of the productions or production packets of the grammar is necessary. That is, most of the linguistic knowledge is put into active form. We hope to get implementational results in early spring 1984.



## References

- 【1】 Anderson, J.: The Grammar of Case: Towards a Localistic Theory. Cambridge University Press, London & New York, 1971.
- 【2】 Anderson, J.: On Case Grammar. Croom Helm, London 1977.
- 【3】 Fillmore, C.: The case for a case. In Universals in Linguistic Theory (eds. Bach, E., and Harms, T.), Holt, Rinehart & Winston, New York 1968, 1-88.
- 【4】 Fillmore, C.: Types of lexical information. In Semantics: An Interdisciplinary Reader (eds. Steinberg, D., and Jacobovitz, L.), Cambridge University Press, 1971, 109-137.
- 【5】 Hays, D.: Dependency theory: A formalism and some observations. Language 40, 1964, 511-525.
- 【6】 Hopcroft, J., and Ullman, J.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading, Mass., 1979.
- 【7】 Hudson, R.: Arguments for a Non-transformational Grammar. The University of Chicago Press, 1976.
- 【8】 Jäppinen, H., et al.: Morphological Analysis of Finnish: A Heuristic Approach. Helsinki Univ. of Tech., Digital Systems Laboratory, Report B26, 1983.
- 【9】 Pajunen, A.: Suomen kielen verbien leksikaalinen kuvaus. Lisensiaattityö. Turun yliopiston suomalaisen ja yleisen kielitieteen laitos, 1982.
- 【10】 Siro, P.: Sijakielioppi (2., korjattu painos). Oy Gaudeamus Ab, Helsinki, 1977.
- 【11】 Tarvainen, K.: Dependenssikielioppi. Oy Gaudeamus Ab, Helsinki, 1977.
- 【12】 Woods, W.: Transition network grammar for natural language analysis. Communications of the ACM 13, 1970, 591-606.