The translation of constitutent structure grammars into connectionist networks

Helmut SCHNELLE and Rolf WILKENS
Sprachwissenschaftliches Institut
Ruhr Universität
D-4630 Bochum 1
Germany

Abstract: Description of a connectionist implementation of an Earley parser.

1. Introduction

We are going to describe a connectivity structure which is a quasi neuronal representation of the computational knowledge usually presented in terms of rules and algorithms. Our system provides the proof that connectionist networks can represent cognitive knowledge of high complexity a fact that has recently been questioned by critics of connectionism (cp. FODOR and PYLYSHYN 1988). However, our system is of the variety of an implementational connectionism implementing systems, whose computational knowledge is already defined by rules - it is not a system which generates new knowledge structures through learning.

More specifically, our system implements parsers for constituent structure grammars according to Earley's rules in terms of networks of Boolean operators. The implementation is automatic, i.e. executed by a compi-ler which automatically translates a grammar into a set of Boolean equations. Our connectionist net is thus like a special purpose parser network defined by the Boolean equations in the same way as any costums specific circuit definition. We shall now explain the essential ideas which characterize the parallel (connectionist) networks compiled from constituent structure rule systems through parallelizing Earley's algorithm. The formal definitions of the compilation algorithm and of the definition of the resulting connectionist network can be found in SCHNELLE and DOUST (1989).

In the following paragraphs our essential ideas will be presented by means of a simple example, the system of constituent structure rules S --> aA, S --> Ab, A --> aa, A --> a to be applied in a parsing process on the string aab.

2. Earley's Representation

Let us first summarize the essentials of Earley's algo-rithm. It operates in two stages: In the first stage, a parse list is computed and in the second stage the correct parse is filtered out from the parse list. For the string aab the information contained in the parse list can be represented as in figure 1 by a superposition of possible sub-trees found applicable in going through the string from left to right. The correct parse "filtered out" is represented in figure 2.

Earley uses another way of representing parse lists and correct parses. He represents them by means of dotted rule symbols and dominance scope numbers entered in ists, one for each input interval. The parse list containing the same information given in the superposition of the trees is as in figure 3. The meaning of such a list should be clear: The

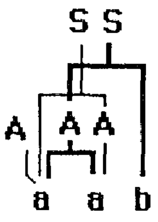

Figure 1:
Parse list tree

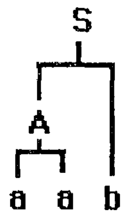

Figure 2:
Correct parse tree

symbols of the input string represented at the bottom exist in the intervals <0,1>, <1,2>,<2,3>. At each completed interval, the rules which have found application so far are entered in the corresponding lists, together with a number indicating the number of intervals dominated by the head symbol of the rule.

List 0	List 1	List 2	List 3
			<S->aA., 2>
			<A->aa. , 2>
	<A->a. , 1>	<A->a. ,1>	<S->Ab. , 3>
	a	a	b

Figure 3. Parse list information with completed dotted rule symbols according to Earley

Let us indicate a feature which is essential in view of our connectionist implementation: Each piece of information in Earley's system is in fact a triple

< list number, dotted symbol, length of dominance >.

The representation in figure 3 is, however, not yet complete as a representation of the parse list. In fact, the parsing process as defined by Earley makes use of further dotted symbols derived from the rules of the underlying constituent struture, namely all dotted rule symbols which can be obtained by placing exactly one dot between symbols to the right of the arrow. The system of dotted rule symbols for our grammar is presented in figure 4. All dotted rule symbols are needed for controlling the parse process.

S -> aA. , S -> a.A , S -> .aA ,
S -> Ab., S -> A.b , S -> .Ab
A -> aa. , A -> a.a , A -> .aa ,
A -> .a , A -> .a
.a. , .b. , .S.

Figure 4 The set of dotted rule symbols derived from the example grammar

The complete parse is computed list by list from left to right as the input string is read in. In principle many dotted rule symbols in the list could be placed simulta-nously but only in a parallel system like the one we shall present, not in Earley's completely sequential implementation on a von Neumann machine.

3.Our representation

How are we going to implement Earley's algorithm in a connectionist net? We follow the localist principle of connectionist implementation: One concept - one unit, but we apply it to the triples in Earley's represen-tation: One triple - one unit. This principle applied to our example of three intervals and, correspondingly, to 3 as the longest possible dominance and to 14 dotted rules (as enumerated in figure 4) yields 3*14*3 = 126 units. In general, a system with n dotted rules and length of input string l would have $n*l^2$ units. The connectivities between the units must be defined in such a way that they generate activity patterns over the three-dimensional system of units (each member of a triple indi- cating a dimension), such that a unit becomes active (1) exactly when the corresponding triple is specified in the Earley algorithm. All other units not specified in the algorithm must remain inactive (0). The parse list given in figure 3 would be represented by the activity pattern over the units in a three dimensional space indicated in figure 5.

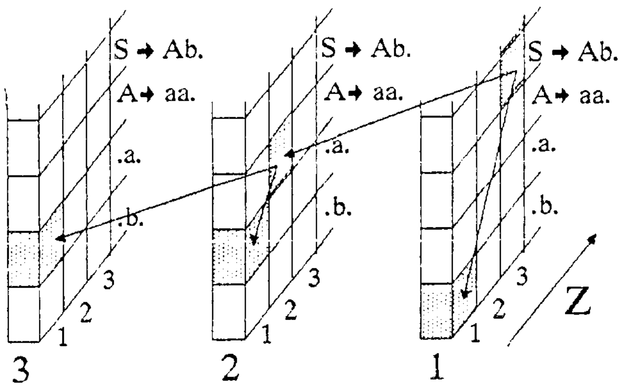

Figure 5. Terminal stage of parse list generation (for terminal dotted rules only). The arrows show how a pattern of activity in this system can be used to represent the correct parse tree given in figure 2.

The representation outlined so far seems to have an essential disadvantage: The space built by the units which represent the parse list structures seems to be unlimited, since it depends on the length of the input string. This is indeed the case. However, the structurally essential feature is not the space used for representing the complete parse list structure but only the space in which the process of generating the parse list structure is executed. Our system can indeed be subdivided archi-tectonically into the representation spaces - one for the parse list, one for the correct parse, and a limited space containing the units which generate the representations. It is only this latter space - comprising grammar units (0,Y,0),(-1,Y,0) and control units (0,Y,-1).(-1,Y,-1) for all dotted rules Y - which has an inhomogenous connectivity structure whose specificity is determined by the constituent structure rule system from which it is compiled. Obvviously, this space of inhomogenous connectivity is limited in our implementation and is 2*2*n (where n is the number of dotted rules).

In this space 2*n units are control bit units whereas 2*n units correspond directly to dotted rule symbols of the original grammar such that their connectivities represent the logical and procedural interdependencies between these symbols in Earley's algorithm. The extension of this space is thus independent of the length of the input string to be parsed.

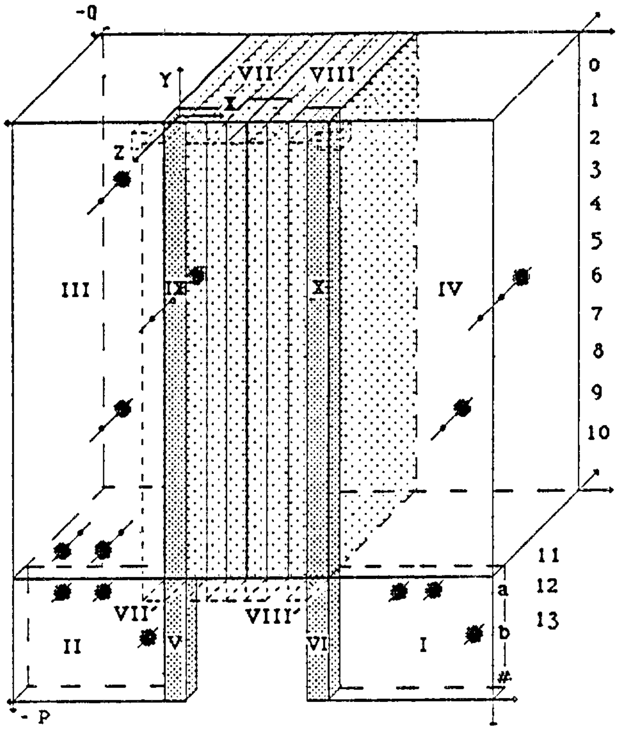

Figure 6. The architecture of the connectionist parser system. (Parse list representation corresponding to figures 1 in space III and correct parse representation corresponding to figure 2 in space 4. Input representations in spaces I and II)

In contrast to this, the units in the representation space have a homogenous connectivity among them, which is completely independent of the grammar implemented. Instead, this connectivity corresponds to the circuit connectivity of a shift register implemented as an integrated circuit.

The overall architecture which derives from our automatic compilation process applied to a given constituent structure is now given as in figure 6. Space I and II contain the representations of the input string, the units in space III represent the parse list under construction and after completion,

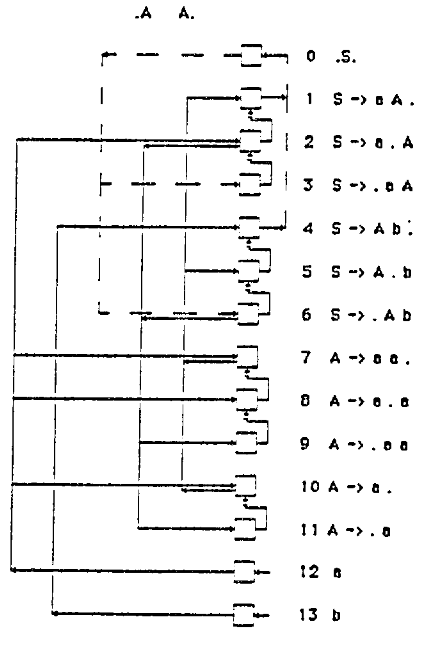

Figure 7 The internal connectivity of the units in the processing space derived from our simple grammar

space IV represents the same for the correct parse. Space IX (resp. X) is the inhomogenous processing space whose connectivity corresponds strictly to the structure of the grammar from which it is compiled.

The inhomogenous internal connectivity within space IX is represented in figure 7. The units represented are also connected to the neighbouring units in the representation space III and to control bits which determine the shifting processes in the representation space.

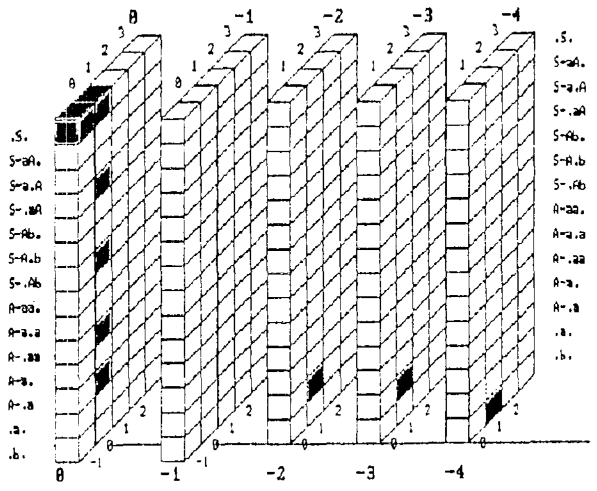

Figure 8 The initial stage of processing. The activity of a control bit unit (0, .S. ,-1) forces the parser to shift the input string in the next step

4. An outline of the connectionist parsing process

The computational process is as follows: Initially the input string is in space I (or is transferred to this space from a word recognizer array analysing acoustic or graphic input). The first input symbol is read into the processing space - more correctly into a connected buffer place of space VII, i.e. the unit (-2, .a., 1) is activated and simultanously the unit (0, .S., 0) - i. e. the initializer unit. (Cp. figure 8)

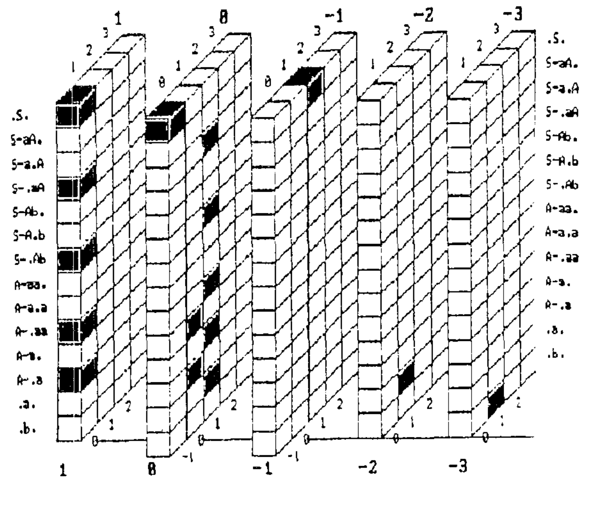

Figure 9 An intermediate stage occurring after reading in the first symbol

Due to the connectivities in position 0 (i.e. in space IX) the units (0, S -> .aA , 0) and (0, S -> .Ab, 0) become simultanously active, and then, depending on them, simultanously the units (0, A-> .aa, 0) and (0, A-> .a, 0). To scan-in the the first terminal the complete pattern of activity has to be shifted one step to the left with the exception of the activation of unit (-2. .t. ,1). The activity of this unit ist transferred to the unit (0, .t. , 1). (This is done because the units located at X=-1 are used as a temporary buffer by the parser.) Figure 9 shows the state after this shifting process has been carried out. But simultanously the parser has to perform the computation of the parse list for the terminal just read. Since the units (O, A ->.a,0), (0,A ->.aa,0) and (0,S->.aA,0) were active while the terminal 'a' was read, the parser must activate the units (0,A->a.,1), (0,A->a.a,1) and (0,S->a.A,1). And the activity of the unit (0,A->a.,1) forces the unit (0, S->A.b,1) to become active. These actions take place according to the connectivities in space IX of figure 6 represented in figure 7.

It should be clear by now how, in principle, the parsing process develops over the connectionist space until the final stage represented schematically in figure 5 is reached. It should also be clear, in principle, how the process of generating the complete parse is produced in space IV through the operation of the units in space X. They determine the "filtering out" of certain unconfirmed parse tree information in the parse list in a process of stepwise information shift from III to IV. We shall not discuss this process here.

5. Perspectives for further research

From a linguistic point of view, it is important to be able to generate connectionist networks for more complicated grammars, in particular for unification based grammars and for principles and parameters based approaches such as those recently developed by Chomsky. So far we have been able to define the appropriate representation space - i.e. the extension of our spaces III and IV - and to develop first ideas about the connectivities derived from symbolic definitions of grammatical properties, i.e. the structures in our spaces IX and X. We are optimistic about the possibilities of translating any unification based formalism working with feature structures into a corresponding connectionist network.

References

Feldman, J.A.(1988) Structured neural networks in nature and in computer science. In: Eckmiller, R. v.d. Malsburg, Chr. Neural Computers, Berlin etc.: Springer

Fodor, J.A., Pylyshyn, Z.W. (1988) Connectionism and cognitive architecture, A critical analysis, Cognition 28: 3 - 71

Schnelle, H., Doust, R. (1990)) A net-linguistic chart parser, In: Reilly, N., Sharkey, N.E. Connectionist Approaches to Languages, Vol.I , Amsterdam: North-Holland