AN EFFICIENT CONNECTIONIST CONTEXT-FREE PARSER

Klaas Sikkel and Anton Nijholt
Department of Computer Science, University of Twente,
PO box 217, 7500 AE Enschede, The Netherlands
sikkel@cs.utwente.nl
anijholt@cs.utwente.nl

ABSTRACT

A connectionist network is defined that parses a grammar in Chomsky Normal Form in logarithmic time, based on a modification of Rytter's recognition algorithm. A similar parsing network can be defined for an arbitrary context-free grammar. Such networks can be integrated into a connectionist parsing environment for interactive distributed processing of syntactic, semantic and pragmatic information.

INTRODUCTION

Connectionist networks are strongly interconnected groups of very simple processing units. Such networks are studied in natural language processing since their inherent parallelism and distributed decision making allows an integration of syntactic, semantic and pragmatic processing for language analysis. See, e.g., (Waltz and Pollack, 1988), (Cotrell and Small, 1989). By isolating the syntactic component — without abandoning the connectionist paradigm — it becomes possible to study context-free parsing in environments where we can make different assumptions about types of networks, learning rules and representations of concepts. Examples of this type of research can be found in (Fanty, 1985), a simple connectionist implementation of the CYK method; (Selman and Hirst, 1987), Boltzmann machine parsing; (Howells, 1988), a relaxation algorithm that utilizes decay over time; (Nakagawa and Mori, 1988), a parallel left-corner parser incorporated in a learning network; (Nijholt, 1990), a Fanty-like connectionist Earley parser.

In this paper we push the speed of the parsing network to its limits, so as to investigate how much parallelism is possible in principle. We define a parsing network that constructs a shared forest of parse trees in $O(\log n)$ time for an input string of length n, using $O(n^6)$ units. Our network is based upon Fanty's "dynamic programming" approach and a type of algorithm first introduced by Rytter (1985). The network is rather large, but not too large: no logarithmic-time parsing algorithm for arbitrary contrext-free languages is known that uses less than $O(n^6)$ processors. Furthermore, the number of units can drastically be reduced (albeit within the same complexity bounds) by a meta-parsing algorithm that constructs a minimal network custom-tailored for a specific grammar.

After some preliminary definitions, we construct a network for a grammar in Chomsky Normal Form. At the end of this paper we argue that a similar network can be built for an arbitrary CFG; space limitations do not allow a detailed presentation.

PRELIMINARIES AND DEFINITIONS

Let $G = (N, \Sigma, P, S)$ be a grammar in Chomsky Normal Form (CNF), i.e., production rules have the form $A \rightarrow BC$ or $A \rightarrow a$. We consider input strings $a_1 \cdots a_m$ with $m < n$, where n is an implementation-dependent constant.

A central role in the parsing algorithm is played by items of the form $\langle A,i,j \rangle$, which are called *triangles.* A triangle $\langle A,i,j \rangle$ is called *recognizable* if $A \Rightarrow^+ a_{i+1} \cdots a_j$. The set of triangles Ξ is defined by

$$\Xi \overset{\text{def}}{=} \{\langle A,i,j \rangle \mid A \in N,\ 0 \leq i < j \leq n\}.$$

A triangle $\langle A,i,j \rangle$ is called *parsable* if it is recognizable and $S \Rightarrow^+ a_1 \cdots a_i A a_{j+1} \cdots a_m$. The collection of all parsable triangles is called the *shared forest* of an input sentence; "forest" because it comprises all different parse trees for that sentence, "shared" as common sub-trees of different parse trees are represented only once. The algorithm and network in section 3 compute the shared forest of a sentence.

We shall also need items of a different kind, called *triangles with a gap*, denoted $\langle\langle A,i,j \rangle, \langle B,k,l \rangle\rangle$. A triangle with a gap $\langle\langle A,i,j \rangle, \langle B,k,l \rangle\rangle$ is called *proposable* if $A \Rightarrow^+ a_{i+1} \cdots a_k B a_{l+1} \cdots a_j$. During the application of the algorithm, we will propose triangles with a gap that need further investigation. If $\langle\langle A,i,j \rangle, \langle B,k,l \rangle\rangle$ has been proposed and $\langle B,k,l \rangle$ can be recognized, we can fill up the gap and recognize $\langle A,i,j \rangle$. The set of all triangles with a gap is denoted by

$$\Gamma \overset{\text{def}}{=} \{\langle\langle A,i,j \rangle, \langle B,k,l \rangle\rangle \mid \langle A,i,j \rangle \in \Xi,\ \langle B,k,l \rangle \in \Xi,\ i \leq k < l \leq j, i \neq k \text{ or } l \neq n\}.$$

The gap can be at the inside or at the outside of a triangle, as shown in Figure 1.

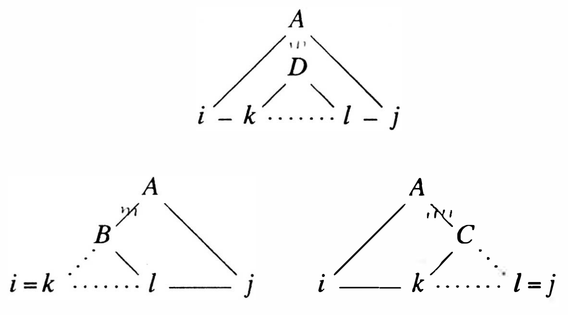

Figure 1. Triangles with a gap

The *size* of a triangle is defined as the length of the substring $a_{i+1} \cdots a_j$: $size(\langle A,i,j\rangle) = j - i$. The size of a triangle with a gap is defined as the size of the triangle minus the size of the gap: $size(\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle) = size(\langle A,i,j\rangle) - size(\langle B,k,l\rangle) = j - i - l + k$.

A FAST CONNECTIONIST PARSING NETWORK FOR CNF GRAMMARS

A variant of Rytter's recognition algorithm

The algorithm presented here is a (for our purpose) improved version of Rytter's recognition algorithm (Gibbons and Rytter, 1988). It can be trivially extended into a parsing algorithm and has a simpler correctness proof. Remarks about the differences with the original algorithm are deferred to the end of this section, so as to keep the exposé as clear as possible. We will describe first what is to be computed by the algorithm, and elaborate on how to compute it afterwards. The recognition algorithm uses two tables of boolean values:

- $recognized(\langle A,i,j\rangle)$ for $\langle A,i,j\rangle \in \Xi$, which is *true* once we have established that $\langle A,i,j\rangle$ is indeed recognizable, and *false* otherwise;
- $proposed(\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle)$ for $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle \in \Gamma$, which is *true* once we have established that $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$ is indeed proposable, and *false* otherwise.

The algorithm will satisfy the following loop-invariant properties:

(I) if $size(\langle A,i,j\rangle) \le 2^k$ and $\langle A,i,j\rangle$ is recognizable then $recognized(\langle A,i,j\rangle) = true$ after k steps,

(II) if $size(\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle) \le 2^k$ and $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$ is proposable then $proposed(\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle) = true$ after k steps.

Acceptance or rejection of the input string depends on the recognizability of $\langle S,0,m\rangle$, hence the number of steps that need to be performed is $\lceil {}^2\log m \rceil$ (the smallest integer $\ge {}^2\log m$).

The well-known Cocke-Younger-Kasami algorithm uses an upper triangular recognition table T_{CYK}: The nonterminal A is added to table entry $t_{i,j}$ if $\langle A,i,j\rangle$ is recognized. The statements $A \in t_{i,j}$ and $recognized(\langle A,i,j\rangle) = true$ are equivalent. We can illustrate the results of our algorithm (though the operations are different) with an extension of the T_{CYK} recognition table. In this case, the recognition table is a three-dimensional structure,

$$T_R = \{t_{i,j,k} \mid 0 \le i < j \le n,\ 0 \le k \le j - i\}.$$

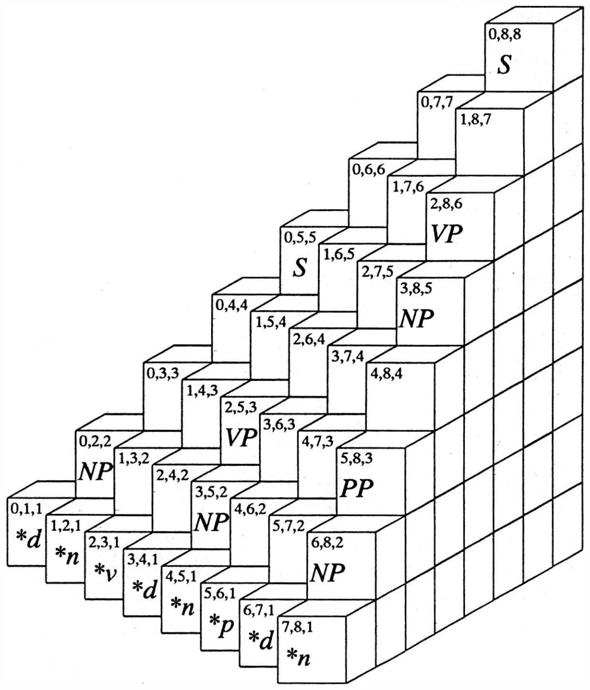

Figure 2. The surface of T_R as it should be computed by the algorithm

The third index k denotes the size of an item. When a triangle $\langle A,i,j\rangle$ is recognized, the nonterminal A is added to $t_{i,j,j-i}$. When a triangle with a gap $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$ is proposed, an object $\langle A,B,k,l\rangle$ is added to $t_{i,j,h}$ with $h = j - i - l + k = size(\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle)$. Hence the surface of T_R is equal to T_{CYK}, representations of triangles with a gap are contained in entries inside the table. Invariants (I) and (II) guarantee that a table entry with height k will be completed within $\lceil {}^2\log k \rceil$ steps. As a simple example, consider the grammar

$$S \rightarrow NP\ VP \mid S\ PP$$
$$NP \rightarrow *det\ *noun \mid NP\ PP$$
$$VP \rightarrow *verb\ NP$$
$$PP \rightarrow *prep\ NP$$

and the input sentence *the boy saw a man with a telescope*. Figure 2 shows the surface of T_R after application of the algorithm.

Having illustrated the purpose of the recognition algorithm, we can now explain how it works. We define the following operations on Ξ, Γ and the tables *recognized* and *proposed*;

```
INITIALIZE :
    for all $\langle A,i,j\rangle \in \Xi$
    do recognized($\langle A,i,j\rangle$) := false od ;
    for all $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle \in \Gamma$
    do proposed($\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$) := false od ;
    for all $\langle A,i-1,i\rangle \in \Xi$
    do  if $A \rightarrow a_i \in P$
          then recognized($\langle A,i-1,i\rangle$) := true fi
    od
```

```
PROPOSE :
    for all $\langle A,i,j\rangle, \langle B,i,k\rangle, \langle C,k,j\rangle \in \Xi$
          such that $A \rightarrow BC \in P$
    do  if recognized($\langle B,i,k\rangle$)
          then proposed($\langle\langle A,i,j\rangle, \langle C,k,j\rangle\rangle$) := true fi ;
          if recognized($\langle C,k,j\rangle$)
          then proposed($\langle\langle A,i,j\rangle, \langle B,i,k\rangle\rangle$) := true fi
    od
```

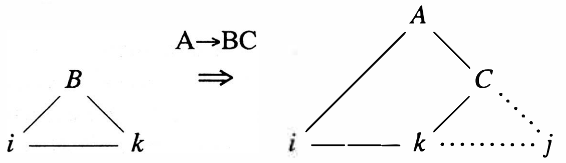

Figure 3. *PROPOSE*

```
RECOGNIZE :
    for all $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle \in \Gamma$ , $\langle B,k,l\rangle \in \Xi$
    do  if proposed($\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$)
             and recognized($\langle B,k,l\rangle$)
          then  recognized($\langle A,i,j\rangle$) := true fi
    od
```

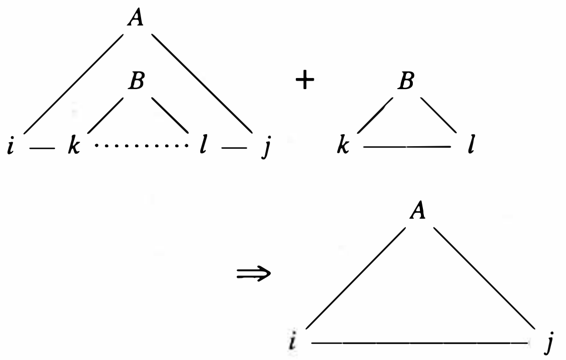

Figure 4. *RECOGNIZE*

```
COMBINE :
    for all $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$ , $\langle\langle B,k,l\rangle,\langle C,m,n\rangle\rangle \in \Gamma$
    do  if proposed($\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$)
             and proposed($\langle\langle B,k,l\rangle,\langle C,m,n\rangle\rangle$)
          then proposed($\langle\langle A,i,j\rangle,\langle C,m,n\rangle\rangle$) := true fi
    od
```

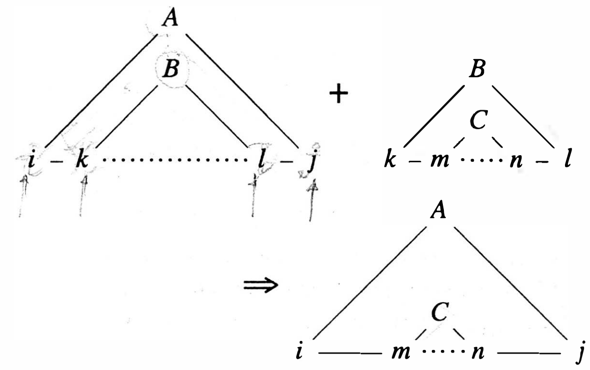

Figure 5. *COMBINE*

The functioning of the operators *PROPOSE*, *RECOGNIZE* and *COMBINE* is illustrated in Figures 3–5. Everything in a **for all** statement can be computed in parallel. The recognition algorithm, using these operators, can be given as:

```
begin
    INITIALIZE ;
    PROPOSE ;
    repeat $\lceil {}^2\log m \rceil$ times
    begin
        RECOGNIZE ;
        PROPOSE ;
        COMBINE ;
        COMBINE
    end ;
    if recognized($\langle S,0,m\rangle$)
    then accept
    else reject
    fi
end
```

In the sequel, we will give a proof of the correctness of the modified Rytter algorithm. But let's first look at an example.

In Figure 6 one parse tree of the input sentence is shown. The algorithm obviously recognizes much more than a single parse tree, but it is sufficient to show that all items in one parse tree are recognized in order to make clear that the top item is recognized. $\langle S,0,8\rangle$ can be recognized in a number of different ways, but that would only clutter up the example. The nodes in the parse tree have been numbered, so

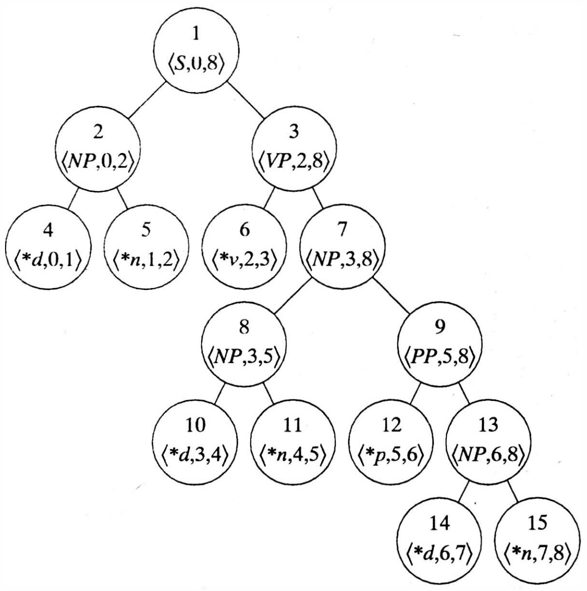

Figure 6. A parse tree

We can identify the triangles by their number rather than by the more cumbersome $\langle A,i,j\rangle$ notation. We will apply the algorithm step-by-step on the items in this tree. Step 0 is shown in Figures 7–8, step 1 in Figures 9–12 and (the first half of) step 2 in Figures 13–14. Circles correspond to recognized triangles, lines correspond to proposed triangles with gaps. The example shows that the algorithm may need less than $\lceil {}^2\log k \rceil$ steps in some cases; we need only 2 steps although 3 are allowed.

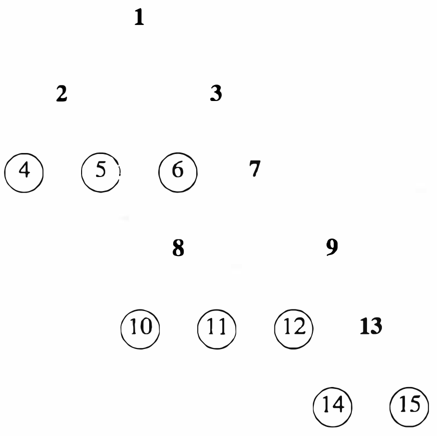

Figure 7. After step 0(a): *INITIALIZE*

I K J

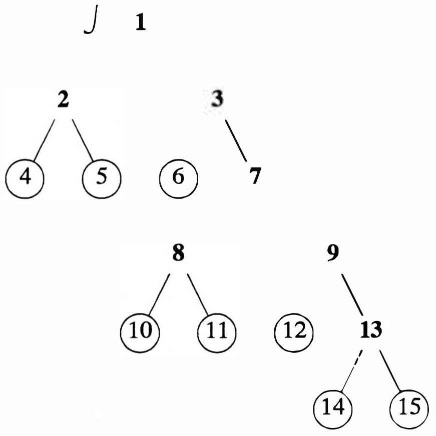

Figure 8. After step 0(b): *PROPOSE*

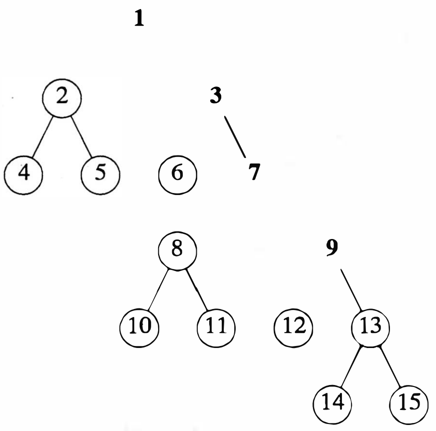

Figure 9. After step 1(a): *RECOGNIZE*

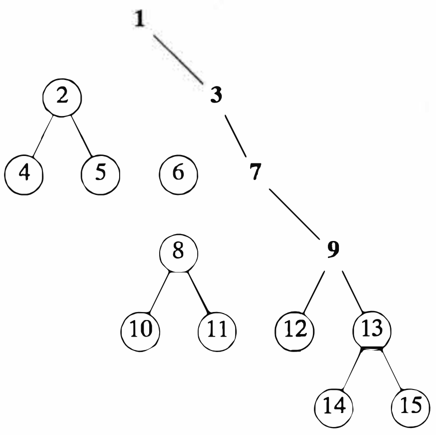

Figure 10. After step 1(b): *PROPOSE*

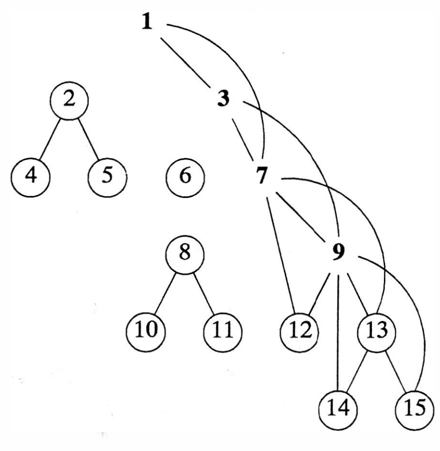

Figure 11. After step 1(c): *COMBINE*

sto calcolando la chiusura di proposed

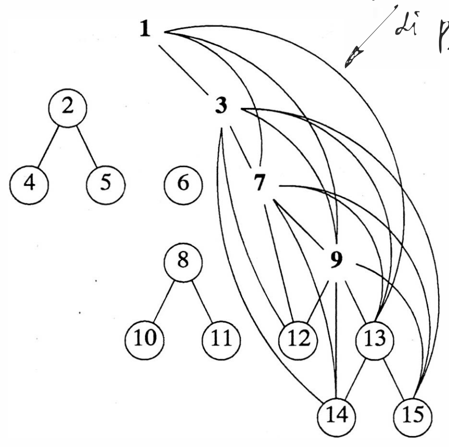

Figure 12. After step 1(d): *COMBINE*

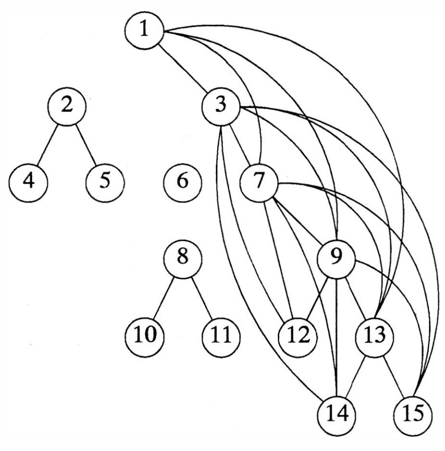

Figure 13. After step 2(a): *RECOGNIZE*

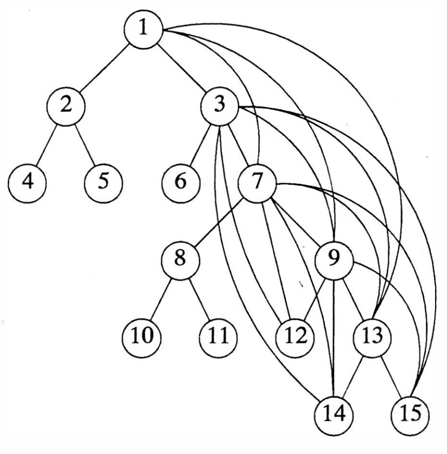

Figure 14. After step 2(b): *PROPOSE*

Correctness of the algorithm

Theorem. *After application of the above algorithm, a triangle will have been recognized if and only if it is recognizable; a triangle with a gap will have been proposed if and only if it is proposable.*

We will give a proof that is a great deal simpler than the proof of the original algorithm (Gibbons and Rytter, 1988). For more details see (Sikkel and Nijholt, 1990).

Terminology. We denote triangles with greek letters ξ, η, ζ etc. The triangles η, ζ are called *a pair of sons of* ξ if $\xi = \langle A,i,j \rangle$, $\eta = \langle B,i,k \rangle$, $\zeta = \langle C,k,j \rangle$ for some $A, B, C \in N$ with $A \rightarrow BC \in G$ and $0 \leq i < k < j \leq n$. For technical reasons we allow empty triangles with a gap $\langle \xi, \xi \rangle$. For such an empty triangle, *proposed* $(\langle \xi, \xi \rangle)$ = *true* by definition.

Basis. It is easy to verify that proposable triangles with a gap of size 1 have been proposed after the initialization step and recognizable triangles of size ≤ 2 have been recognized after step 1.

Induction hypothesis. We write $(\mathrm{I})_k$ for the claim that (I) holds for ξ with $size(\xi) \leq 2^k$ and $(\mathrm{II})_k$ for the claim that (II) holds for $\langle \xi, \eta \rangle$ with $size(\langle \xi, \eta \rangle) \leq 2^k$. Hence $(\mathrm{II})_0$ and $(\mathrm{I})_1$ have been established above. From the induction hypothesis $(\mathrm{II})_{k-1}$, $(\mathrm{I})_k$ we will derive $(\mathrm{II})_k$, $(\mathrm{I})_{k+1}$.

$(\mathrm{II})_k$. Given $(\mathrm{II})_{k-1}$, $(\mathrm{I})_k$, we prove $(\mathrm{II})_k$. Let $\langle \xi, \eta \rangle$ be proposable, $2^{k-1} < size(\langle \xi, \eta \rangle) \leq 2^k$.

- **Claim A**. There is a ϕ with sons ψ, ψ', such that ϕ, ψ, ψ' are recognizable, $\langle \xi, \phi \rangle$, $\langle \psi, \eta \rangle$ are proposable, $size(\langle \xi, \phi \rangle) \leq 2^{k-1}$, $size(\langle \psi, \eta \rangle) \leq 2^{k-1}$. See Figure 15.
  Proof. If $\langle \xi, \eta \rangle$ is proposable, there is a sequence $\zeta_0, \cdots, \zeta_p$ with $\zeta_0 = \xi$, $\zeta_p = \eta$ such that each

$\langle \zeta_i, \zeta_{i+1} \rangle$ is proposed by a *PROPOSE* operation; these "atomic" triangles with a gap are subsequently *COMBINE*d into $\langle \xi, \eta \rangle$. Choose $\langle \phi, \psi \rangle = \langle \zeta_i, \zeta_{i+1} \rangle$ with the largest i such that $size(\langle \xi, \zeta_i \rangle) \leq 2^{k-1}$. From $size(\langle \zeta_{i+1}, \eta \rangle) > 2^{k-1}$ it follows that $size(\langle \xi, \zeta_{i+1} \rangle) \leq 2^{k-1}$, hence a larger i could have been chosen.

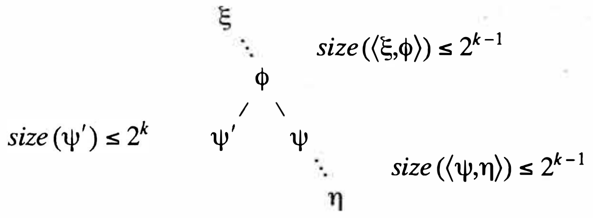

Figure 15. Claim A

From the induction hypothesis we find that $\langle \xi, \phi \rangle$, $\langle \psi, \eta \rangle$ have been proposed after step $k-1$; ψ' has been recognized after the *RECOGNIZE* in step k. Then $\langle \phi, \psi \rangle$ is *PROPOSE*d in step k and two *COMBINE* operations yield *proposed*$(\langle \xi, \eta \rangle)$.

$(\mathrm{I})_{k+1}$. Given $(\mathrm{II})_{k-1}$, $(\mathrm{I})_k$, we prove $(\mathrm{I})_{k+1}$. Let ξ be recognizable, $2^k < size(\xi) \leq 2^{k+1}$.

- **Claim B**. There is an η with a pair of sons θ, ζ such that $size(\langle \xi, \eta \rangle) \leq 2^k$, $size(\theta) \leq 2^k$, $size(\zeta) \leq 2^k$ and η, θ, ζ are recognizable.
  Proof. let ϕ_1 be the largest son of ξ, ϕ_2 the largest son of ϕ_1, etc. Let ϕ_j be the first one with size $\leq 2^k$. Then $\eta = \phi_{j-1}$.

If $\eta = \xi$, $(\mathrm{I})_{k+1}$ follows trivially. Otherwise, Claim A holds and we find a situation as shown in Figure 16.

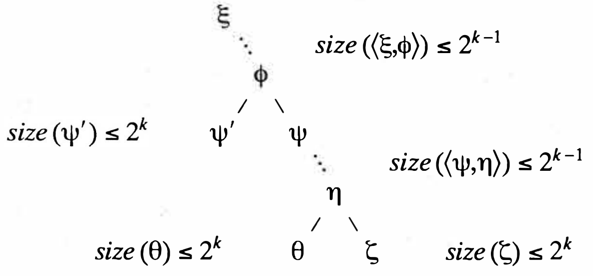

Figure 16. Claims B and A

From the induction hypothesis we find that $\langle \xi, \phi \rangle$, $\langle \psi, \eta \rangle$ have been proposed after step $k-1$; ψ', θ, ζ have been recognized after the *RECOGNIZE* in step k. Then $\langle \phi, \psi \rangle$, $\langle \eta, \zeta \rangle$ are *PROPOSE*d in step k and $\langle \xi, \psi \rangle$, $\langle \psi, \zeta \rangle$ are *COMBINE*d. The second *COMBINE* in step k proposed $\langle \xi, \zeta \rangle$. Hence ξ will be recognized in step $k+1$.

Conclusion. Thus we have established invariants (I) and (II). The "if" parts of the Theorem follow from (I), (II); the "only if" parts from the soundness of each of the operators *INITIALIZE*, *RECOGNIZE*, *PROPOSE* and *COMBINE*. □

A recognizing network

We define a connectionist network in a way that resembles the parsing network of Fanty (1985). The network consists of simple units computing AND and OR functions. The output of every unit is either 1 or 0. An AND unit is activated — i.e. its outputs have value 1 — iff all its inputs have value one; an OR unit is activated iff at least one of its inputs has value 1. In neural networks terminology: an AND unit with k inputs has a threshold value $k - 0.5$, an OR unit has a threshold value 0.5, irrespective of its number of inputs. In order to make a distinction between the two types of units we will write OR units between parentheses "( )" and AND units between brackets "[ ]".

For each triangle $\langle A,i,j \rangle \in \Xi$, the network contains a unit $(R\,\langle A,i,j \rangle)$ with an activation level corresponding to the value of *recognized*$(\langle A,i,j \rangle)$. Likewise, *proposed*$(\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle)$ is represented by a unit $(\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle)$. Furthermore, we need an output unit (*accept*) that is activated only if the sentence is accepted and input units $(\langle a, i \rangle)$ for $a \in \Sigma \cup \{\$\}$, $1 \leq i \leq n+1$. It is assumed that the input units are activated externally and that their activation level remains fixed. If a sentence has m words, then unit $(\langle m+1, \$ \rangle)$ should be activated to mark the end of the sentence.

- *INITIALIZE* is implemented by linking the units $(\langle a, i \rangle)$ to $(\langle A, i-1, i \rangle)$ for $A \rightarrow a \in P$.
- For the *PROPOSE* operation, for all $A \rightarrow BC \in P$ and $0 \leq i < k < j \leq n$, a link from $(R\,\langle B,i,k \rangle)$ to $(\langle \langle A,i,j \rangle, \langle C,k,j \rangle \rangle)$ and a link from $(R\,\langle C,k,j \rangle)$ to $(\langle \langle A,i,j \rangle, \langle B,i,k \rangle \rangle)$ are added to the network.
- For an implementation of *RECOGNIZE*, we need additional match units $[\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle]$ for each $\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle \in \Gamma$. This is because a unit $(R\,\langle A,i,j \rangle)$ can be recognized in more than one way. It should be recognized if both $(\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle)$ and $(R\,\langle B,k,l \rangle)$ are active for some $\langle B,k,l \rangle \in \Xi$. To this end, $(R\,\langle B,k,l \rangle)$ and $(\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle)$ are linked to $[\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle]$ that ANDs their values; $[\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle]$ is linked to $(R\,\langle A,i,j \rangle)$.
- For the *COMBINE* operation, we also need additional match units. For each $\langle \langle A,i,j \rangle, \langle B,k,l \rangle \rangle$ and $\langle \langle B,k,l \rangle, \langle C,m,n \rangle \rangle \in \Gamma$, an AND unit $[\langle \langle A,i,j \rangle, \langle B,k,l \rangle, \langle C,m,n \rangle \rangle]$ is added. It receives

input from $((\langle A,i,j\rangle,\langle B,k,l\rangle))$ and $((\langle B,k,l\rangle,\langle C,m,n\rangle))$ and sends output to $((\langle A,i,j\rangle,\langle C,m,n\rangle))$.

- The $(accept)$ unit receives input from match units $[accept, i]$ that will be activated if a sentence of length i could be recognized. This is accomplished by linking $((\$, i+1))$ and $(R\langle S,0,i\rangle)$ to $[accept, i]$.

An example of a small fraction of the network is given in Figure 17. It represents the units that are used for the recognition of the propositional phrase $\langle PP,5,8\rangle$.

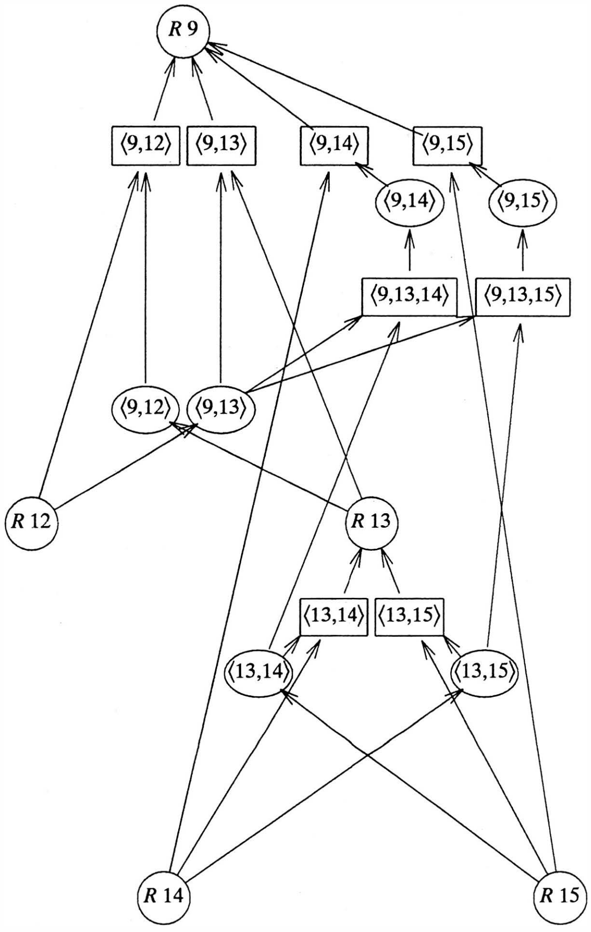

Figure 17. A fraction of the recognizing network

Construction of the shared forest

The main purpose of modifying Rytter's algorithm is the introduction of invariant (II). It will become clear now why we need it. Tacitly we have done all the necessary preparations for the extension to a parsing algorithm, all that is left is to reap the results.

Let $\langle A,i,j\rangle$ be parsable for a particular input string $a_1 \cdots a_m$ $(m \le n)$, and assume $\langle A,i,j\rangle \ne \langle S,0,m\rangle$. Then the following two conditions hold:

(i) $A \Rightarrow^{+} a_{i+1} \cdots a_j$,

(ii) $S \Rightarrow^{+} a_1 \cdots a_i A a_{j+1} \cdots a_m$.

In other words, $\langle A,i,j\rangle$ is parsable if $\langle A,i,j\rangle$ is recognizable and $\langle\langle S,0,m\rangle,\langle A,i,j\rangle\rangle$ is proposable. Consequently, *parsed*$(\langle A,i,j\rangle)$ must be made *true* if both *proposed*$(\langle\langle S,0,m\rangle,\langle A,i,j\rangle\rangle)$ and *recognized*$(\langle A,i,j\rangle)$ are *true*. This can be done in parallel in one step! We define an additional boolean table *parsed*$(\langle A,i,j\rangle)$ for $\langle A,i,j\rangle \in \Xi$ and an operation on Ξ, Γ and the table *parsed*, as follows:

```
PARSE :
    for all $\langle A,i,j\rangle \in \Xi$
    do parsed($\langle A,i,j\rangle$) := false od ;
    if recognized($\langle S,0,m\rangle$)
    then parsed($\langle S,0,m\rangle$) := true ;
         for all $\langle A,i,j\rangle \in \Xi$
         do   if proposed($\langle\langle S,0,m\rangle,\langle A,i,j\rangle\rangle$)
                    and recognized($\langle A,i,j\rangle$)
              then parsed($\langle A,i,j\rangle$) := true fi
         od
    fi
```

The recognition algorithm is extended to a full-fledged parsing algorithm by inserting one *PARSE* operation after the **repeat** loop.

We can extend the network accordingly. The table *parsed* will be represented by a collection of AND units $[P\langle A,i,j\rangle]$. Additionally, we need a collection of match units $[Q_m\langle A,i,j\rangle]$ for $1 \le m \le n$ and $\langle A,i,j\rangle \in \Xi$ and a collection of OR match units $(Q\langle A,i,j\rangle)$ for $\langle A,i,j\rangle \in \Xi$.

- $[Q_m\langle A,i,j\rangle]$ will be activated if *parsed*$(\langle S,0,m\rangle)$ = *true* and *proposed*$(\langle\langle S,0,m\rangle,\langle A,i,j\rangle\rangle)$ = *true*. That is, for all possible values of m, $[P\langle S,0,m\rangle]$ and $((\langle S,0,m\rangle,\langle A,i,j\rangle))$ are linked to $[Q_m\langle A,i,j\rangle]$.
- $(Q\langle A,i,j\rangle)$ will be activated if the above holds for *some* m. To this end, each $[Q_m\langle A,i,j\rangle]$ is linked to $(Q\langle A,i,j\rangle)$.
- $[P\langle A,i,j\rangle]$, obviously, receives input from $(R\langle A,i,j\rangle)$ and $(Q\langle A,i,j\rangle)$.
- In order to start the parsing phase, all $[accept, m]$ units are linked to $(Q\langle S,0,m\rangle)$. If a string of length m is accepted, then $(R\langle S,0,m\rangle)$ will be active, hence $[P\langle S,0,m\rangle]$ will be activated.

If $(Q\langle A,i,j\rangle)$ is activated (via $[Q_l\langle A,i,j\rangle]$) by any $[P\langle S,0,l\rangle]$ with $1 \le l \le m$, it is also activated by $[P\langle S,0,m\rangle]$, because $\langle\langle S,0,m\rangle,\langle S,0,l\rangle\rangle$ and $\langle\langle S,0,l\rangle,\langle A,i,j\rangle\rangle$ can be *COMBINE*d. Thus $[P\langle A,i,j\rangle]$ will be activated if and only if $\langle A,i,j\rangle$ is parsable.

Complexity of the network

The number of input units ($\langle a, i\rangle$) is $(|\Sigma|+1)\cdot(n+1) = O(|\Sigma|\cdot n)$. The *COMBINE* match units $[\langle\langle A,i,j\rangle,\langle B,k,l\rangle,\langle C,m,n\rangle\rangle]$ account for the highest order of all other types of units, $O(|N|^3\cdot n^6)$, yielding a total of $O(|\Sigma|\cdot n+|N|^3\cdot n^6)$ units. It is easy to verify that the number of connections is also $O(|\Sigma|\cdot n+|N|^3\cdot n^6)$.

These numbers conform to the best known complexity measures for logarithmic parsing algorithms: $O(\log n)$ time on a CRCW PRAM and $O(\log^2 n)$ time on a CREW PRAM. PRAM models use $O(n^6)$ processors. It is not obvious that an equivalent network exists with the same order of complexity. A general method to construct a network composed of AND and OR units for an arbitrary PRAM is given by Stockmeyer and Vishkin (1984). Applying this general method, however, would yield $O(n^{13})$ units, rather than our custom-tailored network of $O(n^6)$ units.

Meta-parsing

We defined units for *all* $\langle A,i,j\rangle\in\Xi$ and $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle\in\Gamma$. A large fraction of these units will never be needed. For any particular grammar we can establish a much smaller network, by an algorithm that closely resembles the parsing algorithm. Such analysis has been called *meta-parsing* (Nijholt, 1990).

A triangle $\langle A,i,j\rangle$ is called *meta-recognizable* if $\langle A,i,j\rangle$ is recognizable for some input string $a_1\cdots a_m\in\Sigma^m$, $(m\leq n)$. Similarly, $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle$ is called *meta-proposable* if there is an input string such that $\langle A,i,j\rangle$ is proposable; $\langle A,i,j\rangle$ is called *meta-parsable* if there is an input string such that $\langle A,i,j\rangle$ is parsable. These meta-properties can be computed in advance, and incorporated in the structure of the network. The meta-recognizable and meta-parsable items for our example grammar and $n=8$ are shown in tabular form in Figures 18 and 19. The meta-parsing algorithm is identical to the parsing algorithm but for two small differences:

- *meta-recognized*($\langle A,i-1,i\rangle$) is made *true* if $A\rightarrow a\in P$ *for any* $a\in\Sigma$,
- *meta-parsed*($\langle S,0,i\rangle$) is made *true for every* $\langle S,0,i\rangle$ *that has been meta-recognized.*

It is easy to verify that after application of the meta-algorithm, $\langle A,i,j\rangle$ has been recognized if and only if $\langle A,i,j\rangle$ is meta-recognizable; similarly for meta-proposable and meta-parsable items.

For the construction of the shared forest, we only have to consider triangles that are meta-parsable. All triangles that are not meta-parsable can be discarded:

*d *n *v *p	NP	VP PP		S NP	VP PP		S NP
	*d *n *v *p	NP	VP PP		S NP	VP PP	
		*d *n *v *p	NP	VP PP		S NP	VP PP
			*d *n *v *p	NP	VP PP		S NP
				*d *n *v *p	NP	VP PP	
					*d *n *v *p	NP	VP PP
						*d *n *v *p	NP
							*d *n *v *p

Figure 18. $A\in t_{i,j}$ if $\langle A,i,j\rangle$ is meta-recognizable

*d	NP			S NP			S
	*n						
		*v *p		VP PP			VP
			*d	NP			NP
				*n			
					*v *p		VP PP
						*d	NP
							*n

Figure 19. $A\in t_{i,j}$ if $\langle A,i,j\rangle$ is meta-parsable

even if such a triangle is recognized, it can never contribute to a parse tree for any input. Thus we define the *minimal* set of triangles and triangles with gaps, as follows:

$$\Xi_{\min} = \{\langle A,i,j\rangle\in\Xi \mid \langle A,i,j\rangle \text{ is meta-parsable}\}\,,$$
$$\Gamma_{\min} = \{\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle\in\Gamma \mid \langle A,i,j\rangle\in\Xi_{\min},\ \langle B,k,l\rangle\in\Xi_{\min},\ \langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle \text{ is meta-proposable}\}\,.$$

While constructing the network, we only have to introduce units for $\langle A,i,j\rangle\in\Xi_{\min}$, $\langle\langle A,i,j\rangle,\langle B,k,l\rangle\rangle\in\Gamma_{\min}$ and appropriate match units. The reduced network still yields the shared forest.

Robustness of the network

In contrast with Fanty's network, even the minimal network is rather robust. When a few units do not function, it is most likely that the proper input strings will be accepted. There is a multitude of different ways in which a triangle can be recognized; if the most direct path is broken, chances are that the triangle is recognized by an alternative path, using slightly more time. That is, unless one of the relatively few vital units breaks down, the recognition network shows graceful degradation. The parsing part of the network has no redundancy, however. If any unit fails, a triangle in the shared forest may be lost. But this is less dramatic than failure to recognize a valid sentence.

It is possible to supplement the recognition network with a robust parsing network if a top-down structure is used that is equivalent to the bottom-up structure, as in Fanty's network. Such a top-down network would yield a parse forest in logarithmic, rather than constant time. But that does not really matter as time complexity of the network is logarithmic anyway.

Bibliographic notes

The CYK algorithm can be found in any textbook on formal languages, e.g. (Harrison, 1978). A connectionist network for the CYK algorithm has been defined by Fanty (1985) and circulated on a wider scale in (Fanty, 1986).

Rytter's recognition algorithm is presented in (Rytter, 1985) and (Gibbons and Rytter, 1988). A similar algorithm is independently described by Brent and Goldschlager (1984). The operators *PROPOSE*, *COMBINE* and *RECOGNIZE* were called *ACTIVATE*, *SQUARE* and *PEBBLE* in the original algorithm. The word "activate" had to be changed so as to avoid confusion with activation of a unit. The new identifiers are chosen because we operate in a parsing context ("recognize") rather then a combinatorial context ("pebble"). Rytter's algorithm performs the following steps:

- step 0: *INITIALIZE*
- step k ($k > 0$): *ACTIVATE*; *SQUARE*; *SQUARE*; *PEBBLE*

which do *not* satisfy invariant (II)! Hence the algorithm does not allow a similar trivial extension for the computation of a shared forest. In (Gibbons and Rytter, 1988), the correctness of the Rytter's algorithm is derived from the correctness of a "pebble game" on binary trees, which has a rather complicated proof. The proof of the modified algorithm as presented above is a lot simpler, mainly due to the introduction of invariant (II).

EXTENSION TO ARBITRARY CONTEXT-FREE GRAMMARS

It is possible to define a similar parsing network for an arbitrary context-free grammar. Rytter's algorithm can be regarded as a speed-up of the CYK algorithm, using more resources. In the same way, bottom-up versions of Earley's algorithm (Graham, Harrison and Ruzzo, 1980), (Chiang and Fu, 1984) can be speeded up in a similar way. Triangles have the form $\langle A \rightarrow \alpha \bullet \beta, i, j \rangle$ for $A \rightarrow \alpha\beta \in P$ and $0 \leq i \leq j \leq n$.

$A \rightarrow \alpha \bullet \beta$ is recognizable iff $\alpha \Rightarrow^* a_{i+1} \cdots a_j$. Proposability can be defined accordingly. A triangle $\langle A \rightarrow \alpha \bullet \beta, i, j \rangle$ is parsable if there is a $\gamma \in V^*$ such that $S \Rightarrow^* a_1 \cdots a_i A \gamma$, $\alpha \Rightarrow^* a_{i+1} \cdots a_j$ and $\beta\gamma \Rightarrow^* a_{j+1} \cdots a_m$.

The network for arbitrary CFGs has $O(g^3 \cdot |P|^3 \cdot n^6)$ units and $O(g^3 \cdot |P|^3 \cdot n^6)$ connections, in which g is the average number of symbols in the right-hand side of a grammar rule. For a full treatment we refer to (Sikkel and Nijholt, 90).

A similar parsing algorithm for arbitrary CFGs on PRAM models is discussed in (de Vreught and Honig, 1991).

CONCLUSIONS

A modification of Rytter's logarithmic time recognition algorithm for CNF grammars has been introduced. This algorithm is conceptually easier than the original, and the correctness proof is a great deal simpler. Furthermore, the construction of a shared parse forest represented by a set of triangles can be added in constant time.

We have defined a connectionist network that parses a CNF grammar with the above algorithm in $O(\log n)$ time using $O(|\Sigma| \cdot n + |N|^3 \cdot n^6)$ units. This conforms to the best known complexity bounds on a CRCW PRAM, and is a factor $\log n$ faster than the best algorithm on a CREW PRAM known to date. A Similar network can be constructed for an arbitrary context-free grammar.

A network of minimum size for a particular grammar can be custom-tailored. The meta-parsing algorithm that establishes the configuration of a network for the specific grammar is almost identical to the parsing algorithm that is implemented by the network.

The network is robust in the sense that a few broken down units will most likely cause some degradation in performance but still all valid sentences will be recognized.

A network structure with $O(n^6)$ units is too large for any serious practical implementation in natural language processing. The purpose of our investigations, however, has been to push the time complexity to its very limits to see how much parallelism is possible *in principle*. These results confirm that connectionist networks can be used as a suitable abstract machine model for parallel algorithms. It is also confirmed that traditional parsing algorithms for general context-free languages can be given connectionist implementations, allowing integration into more comprehensive connectionist networks for natural language analysis.

REFERENCES

Brent, Richard P. and Goldschlager, Leslie M. (1984). "A Parallel Algorithm for Context-Free Parsing," *Australian Computer Science Communications* **6**, 7, 7.1–7.10.

Chiang, Y.T. and Fu, K.S. (1984). "Parallel Parsing Algorithms and VLSI implementations for Syntactic Pattern Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**, 3, 302–314.

Cottrell, Garrison W. (1989). *"A Connectionist Approach to Word Sense Disambiguation"*, Research Notes in Artificial Intelligence, Morgan Kaufmann Publishers.

Fanty, Mark A. (1985). *"Context-free Parsing in Connectionist Networks"*, report TR 174, Computer Science Dept., University of Rochester, Rochester, NY.

Fanty, Mark A. (1986). M.A. Fanty: "Context-free Parsing in Connectionist networks," in: J.S. Denker (Ed.), *"Neural Networks for Computing"*, Snowbird, UT, API conference proceedings 151, American Institute of Physics, 140–145.

Gibbons, Alan and Rytter, Wojciech (1988). *"Efficient Parallel Algorithms"*, Cambridge University Press.

Graham, Susan L, Harrison, Michael A. and Ruzzo, Walter L. (1980). "An Improved Context-Free Recognizer," *ACM Transactions on Programming Languages and Systems* **2** 415–462.

Harrison, Michael (1978). *"Introduction to Formal Language Theory"*, Addison-Wesley, Reading, Mass.

Howells, Tim (1988). "VITAL: a Connectionist Parser," *Proc. 10th Conf. of the Cognitive Science Society* 18–25.

Nakagawa, Hiroshi and Mori, Tatsunori (1988). "A parser based on a connectionist model," *Proc. 12th Int. Conf. on Computational Linguistics* (COLING'88), Budapest 454–458.

Nijholt, Anton (1990). "Meta-Parsing in Neural Networks," *Proc. 10th European Meeting on Cybernetics and Systems Research*, R. Trappl (Ed.), Vienna 969–976.

Rytter, Wojciech (1985). "On the recognition of context free languages," in: *"Proc. 5th Symp. on Fundamentals of Computation Theory"*, Lecture Notes in Computer Science 208, Springer-Verlag 315–22.

Selman, Bart and Hirst, Graeme (1987). "Parsing as an energy minimization problem," in: *Genetic algorithms and Simulated Annealing*, Research Notes in AI, Morgan Kaufmann Publishers, Los Altos 141–154.

Sikkel, Klaas (1990). *"Connectionist Parsing of Context-Free Languages"*, Memoranda Informatica 90-30, Computer Science Department, University of Twente, Enschede, The Netherlands.

Stockmeyer, Larry and Vishkin, Uzi (1984). Simulation of Parallel Random Access Machines by Circuits, *SIAM J. Comput.*, **13**, 2, 409–422.

de Vreught, Hans and Honig, Job (1991) "Slow and Fast Parallel Recognition," *Proc. 2nd Int. Workshop on Parsing Technologies*, (IWPT'91), Cancun, Mexico.

Waltz, David L. and Pollack, Jordan B. (1988). "Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation," in: D.L. Waltz, J.A. Feldman (Eds.), *Connectionist models and their implications*, Readings from Cognitive Science, Ablex Publishing Co., Norwood, NJ, 181–204.