NEW FRONTIERS BEYOND CONTEXT-FREEMESS:
DI-GRAMMARS AND DI-AUTOMATA.

Peter Staudacher
Institut für Allgemeine und Indogermanische Sprachwissenschaft
Universität Regensburg
Postfach 397
8400 Regensburg 1
Germany

Abstract
A new class of formal languages will be defined - the Distributed Index Languages (DI-languages). The grammar-formalism generating the new class - the DI-grammars - cover unbound dependencies in a rather natural way. The place of DI-languages in the Chomsky-hierarchy will be determined: Like Aho's indexed Languages, DI-languages represent a proper subclass of Type 1 (context-sensitive languages) and properly include Type 2 (context-free languages), but the DI-class is neither a subclass nor a superclass of Aho's indexed class. It will be shown that, apart from DI-grammars, DI-languages can equivalently be characterized by a special type of automata - DI-automata. Finally, the time complexity of the recognition-problem for an interesting subclass of DI-Grammars will approximately be determined.

1 Introduction
It is common practice to parse nested Wh-dependencies, like the classical example of Rizzi (1982) in (1),

(1) Tuo fratello, [a cui]1 mi domando [che storie]2 abbiano raccontato t2 t1, era molto preoccupato
(Your Brother, [to whom]1 I wonder [which stories]2 they told t2 t1 was very troubled)

using a stack mechanism. Under the binary branching hypothesis the relevant structure of (1) augmented by wh-stacks is as follows:

\[
\begin{array}{c}
\text{[a cui]}_1 \text{ mi domando} \\
\text{[che storie]}_2 \text{abbiano} V^2(t_2,t_1) \\
/ \\
V^1(t_2) \text{ PP} t_1 \\
/ \\
\text{NP} t_2 \text{ pop} \\
/ \\
\text{raccontato} t_2 \text{ t1} \\
\end{array}
\]

Up to now it is unclear, how far beyond context-freeness the generative power of a Type 2 grammar formalism is being extended if such a stack mechanism is grafted on it (assuming, of course, that an upper bound for the size of the stack can not be motivated).

Fernando Pereira's concept of Extraposition Grammar (XG), introduced in his influential paper (Pereira, 1981; 1983; cf. Stabler, 1987) in order to delimit the new territory, can be shown to be inadequate for this purpose, since it is provable that the class of languages generable by XGs coincides with Type 0 (i.e. XGs have the power of Turing machines), whereas the increase of power by the stack mechanism is not even enough to generate all Type 1 languages (see below).

In (2) an additional point is illustrated: the stack \([t_2,t_1]\) belonging to \(V^2\) has to be divided into the substacks \([t_2]\) and \([t_1]\), which are then inherited by the daughters \(V^1\) and PP. For the PP-index \(t_1\) is not discharged from the top of the \(V^2\)-stack \([t_2,t_1]\). Generalizing to stacks of unlimited size, the partition of a stack among the inheriting subconstituents \(K_1\) and \(K_2\) of a constituent \(K_0\) is as in (3)

\[
\begin{array}{c}
K0[t_1,\ldots,t_j,t_{j+1},\ldots,t_k] \\
/ \\
K1[t_1,\ldots,t_j] \quad K2[t_{j+1},\ldots,t_k] \\
\end{array}
\]
If the generalization in (3) is tenable, the extension of context-free grammars (Vijay-Shanker and Weir, 1991, call the resulting formalism "linear indexed grammar" (LIG)) discussed by Gazdar in (Gazdar, 1988), in which stacks are exclusively passed over to a single daughter (as in (3.1)), is too weak.

(3.1) a) \[ K_0[t_1, \ldots, t_k] \]
\[ \quad \quad \quad K_1[t_1, \ldots, t_k] \]
\[ \quad \quad \quad K_2 \]
\[ \quad \quad \quad K_1 \]
\[ \quad \quad \quad K_2[t_1, \ldots, t_k] \]

Stack-transmission by distribution, however, as in (3) suggests the definition of a new class of grammars properly containing the context-free class.

2 DI-Grammars and DI-languages

A DI-grammar is a 5-tuple \( G = (N,T,F,P,S) \), where \( N,T,S \) are as usual, \( F \) is an alphabet of indices, \( P \) is a set of rules of the following form

1) (a) \( A \rightarrow \alpha \) (b) \( A \rightarrow \alpha B \) (c) \( A \rightarrow \alpha \gamma \)
\( (A,B \in N; \alpha, \gamma \in (N \cup T)^*; \beta \in F) \)

The relation "\( = \Rightarrow \)" or "directly derives" is defined as follows:

2) \( \alpha = \Rightarrow \beta \)
if either i)
\( \alpha = \delta_{\text{Index}} \gamma, \delta, \gamma \in (NF^* \cup T)^*, \text{index} \in F^*, A \in N, \)
\( A \rightarrow B_1B_2\ldots B_n \) is a rule of form 1)(a)
\( \beta = \delta_{B_1\text{Index}}B_2\text{Index}2\ldots B_n\text{Index}_n \gamma \)

or ii)
\( \alpha = \delta_{\text{Index}} \gamma, \delta, \gamma \in (NF^* \cup T)^*, \text{index} \in F^*, A \in N, \)
\( A \rightarrow B_1B_2\ldots B_n \) is a rule of form 1)(b), \( \beta \in F \)
\( \beta = \delta_{B_1\text{Index}}B_2\text{Index}2\ldots B_n\text{Index}_n \gamma \)

or iii)
\( \alpha = \delta_{\text{Index}} \gamma, \delta, \gamma \in (NF^* \cup T)^*, \text{index} \in F^*, A \in N, \)
\( A \rightarrow B_1B_2\ldots B_n \) is a rule of form 1)(c), \( \beta \in F \)
\( \beta = \delta_{B_1\text{Index}}B_2\text{Index}2\ldots B_n\text{Index}_n \gamma \)

(*) and index = index\text{Index}2\ldots index_n, \( 0 \leq \text{index} \leq n \) (i.e. the empty word)

The reflexive and transitive closure \( *=\Rightarrow \) of \( =\Rightarrow \) is defined as usual.

Replacing (*) by index = index\text{Index}2\ldots index_n for \( B_i \in N, \text{index}_i = \varepsilon \) for \( B_i \in T \), changes the above definition into a definition of Aho's well known indexed grammars. How index-percolation differs in indexed and Di-grammars is illustrated in (4).

(4) Index-Percolation

(i) in Aho's Indexed-Grammars

| Index-Multiplication | Index-Distribution |
|----------------------|-------------------|

The region in the Chomsky hierarchy occupied by the class of DI-languages is indicated in (5)

(5)

![Chomsky Hierarchy Diagram]

where

(5.1) \( L_1 = \{a^n b^n c^n; n \geq 1\} \)

(5.2) \( L_2 = \{a^k; k = 2^n, 0 \leq n\} \)

(5.3) \( L_3 = \{w_1 w_2 \ldots w_n z_1 w_1 z_2 \ldots z_{n-1} m(w_n)m(w_{n-1}) \ldots m(w_2)m(w_1); n \geq 1 \}
\quad \& \quad w_i \in \{a,b\}^* \quad (1 \leq i \leq n \}
\quad m(y) \text{ is the mirror image of } y \text{ and } D_1 \text{ is the Dyck language generated by the following CFG } \)
\( G_k (D_1 = L(G_k)) \)
\( G_k = (\{S\}, \{\{\}, \{\}, R_k, S) \)
\( \text{where } R_k = \{S \Rightarrow [S], S \rightarrow SS, S \rightarrow \varepsilon \} \)

(5.4) \( L_4 = \{a^k; k = n^n, n \geq 1\} \)

(5.5) \( L_4 \text{ is not an indexed language, s. Takeshi Hayashi (1973))} \)

By definition (see above), the intersection of the class of indexed languages and the class of DI-languages includes the context-free (cfr) languages. The inclusion is proper, since the (non-cfr) language \( L_1 \) is generated by \( G_1 = (\{S,A,B\}, \{a,b,c\}, \{e\}, R_1, S) \), where \( R_1 = \{S \Rightarrow aAe, A \rightarrow aAge, A \rightarrow B, Bg \rightarrow bB, Be \rightarrow b\} \), and \( G_1 \) obviously is both a Di-grammar and an indexed grammar.

Like cfr. languages and unlike indexed languages, DI-languages have the constant growth property (i.e. for every DI-grammar \( G \) there exists a \( k \in N \), s.th. for every \( w \in L(G) \), s.th. \( |w| > k \), there exists a sequence \( w \) such...
(w_1, w_2, w_3, ..., (w_i \in L(G))) such that |w_n| < |w_{n+1}| < (n+1) \times |w_1| for every member w_i of the sequence. Hence L_2, and a fortiori L_4, is not a DI-language. But L_2 is an indexed language, since it is generated by the indexed grammar G_2 = (\{S,A,D\}, \{a\}, \{f,g\}, R_2, S) where R_2 = \{S \rightarrow Ag, A \rightarrow Af, A \rightarrow D, Df \rightarrow DD, Dg \rightarrow a\}.

L_3 is a DI-language, since it is generated by the DI-grammar G_3 = (\{S,M,Z\}, \{a,b,[,]\}, \{f,g\}, R_3, S) where

\begin{align*}
& R_3 = \{ S \rightarrow aSf, M \rightarrow [M], Zf \rightarrow a, Zg \rightarrow b, \\
& S \rightarrow bSgb, M \rightarrow MM, Zf \rightarrow a, Zg \rightarrow b \}
\end{align*}

E.g., abb[ab]bba (\in L_3) is derived as follows:

\begin{align*}
S & \rightarrow aSf \\
\rightarrow abSFa \\
\rightarrow abbSfa \\
\rightarrow abbSgb \\
\rightarrow abb[ab]bba
\end{align*}

However, the interdependently extendible parts of x_{s_1}, x_{s_2}, ..., x_{s_n}, t_{n-1}, t_n, p_n^1, p_n^2, and s_n_{s_1} can not all be subwords of the central component [x_{m}] of x (or all be subwords of the peripheral components x_{1} or x_{r}).

2.1 DI-Grammars and Indexed Grammars

Considering the well known generative strength of indexed grammars, it is by no means obvious that L_3 is not indexed. In view of the complexity of the proof that L_3 is not indexed, only some important points can be indicated - referring to the 3 main parts of every word x \in L_3 by x_{1}, [x_{m}], x_{r}, as illustrated in the example (6):

\begin{align*}
& \text{ab abb bbbbbb]abbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbbaabbba} \\
& \text{[x_{m}] [x_{r}] [x_{1}] = x}
\end{align*}

Assume that there is a indexed grammar G = (N,T,F,P,S) such that L_3 = L(G):

1. Since G_1 can not be contextfree, it follows from the intercalation (or "pumping") lemma for indexed grammars proved by Takeshi Hayashi in (Hayashi, 1973) that there exists for G_1 an integer k such that for any x \in L_3 such that |x| > k a derivation can be found with the following properties:

S \rightarrow zAf/\mu^* \rightarrow zAf/\mu f \eta r s_1 z' 
= \rightarrow zAf/\mu f \eta r s_1 z \text{ for } \mu \in \{\mu_1; \mu_2; \mu_3; \mu_4; \mu_5; \mu_6; \mu_7; \mu_8; \mu_9\}

\begin{align*}
& (zz', r_1 r_1 \in T^*; s_1 t_1 s_1 \in T^*, f \in F, \mu_1, \eta_1 \in F^*)
\end{align*}

By intercalating subderivations which can effectively be constructed this derivation can be extended as follows:

\begin{align*}
S \rightarrow zAf/\mu f \eta r s_1 z' \\
= \rightarrow zAf/\mu f \eta r s_1 z \text{ for } \mu \in \{\mu_1; \mu_2; \mu_3; \mu_4; \mu_5; \mu_6; \mu_7; \mu_8; \mu_9\}
\end{align*}

The interdependently extendible parts of x_{s_1}, x_{s_2}, ..., x_{s_n}, t_{n-1}, t_n, p_n^1, p_n^2, and s_n_{s_1} can not all be subwords of the central component [x_{m}] of x (or all be subwords of the peripheral components x_{1} or x_{r}).
where \( k = 2^n \), \( w_i \in \{a,b\}^+ \) for \( 1 \leq i \leq 2^n \); \( m(w_i) \) is the mirror image of \( w_i \); i.e., the central part \([x_m]\) of such a word contains \( 2^{n+1} \) pairs of parentheses, as shown in (9) for \( n=3 \):

\[
([w_8][w_7][w_6][w_5][w_4][w_3][w_2][w_1])
\]

According to our assumption, \( G_I \) generates all words having the form (8). Referring to the derivation in (7), consider a path from \( M_\mu \) to any of the parenthesized parts \( w_i \) of \([x_m]\) in (8). (Ignoring for expositional purposes the possibility of "storing" (a constant amount of) parentheses in nonterminal nodes,) because of 2. and 3. an injective mapping can be defined from the set of pairs of parentheses containing at least two other (and because of the structure of (8) disjunct) pairs of parentheses into the set of branching nodes with (at least) two nonterminal daughters. Call a node in the range of the mapping a \( P \)-Node. Assuming without loss of generality that each node has at most two nonterminal daughters, there are \( 2^n+1 \) such \( P \)-nodes in the subtree rooted in \( M_\mu \) and yielding the parenthesized part \([x_m]\) of (8). Furthermore, every path from \( M_\mu \) to the root \( W_i \) of the subtree yielding \([w_i]\) contains exactly \( n \) \( P \)-nodes (where \( 2^n = k \) in (8)).

Call an index-symbol \( f \) inside the index-stack \( \mu \) a \( w_i \)-index if \( f \) is discharged into a terminal constituting a parenthesized \( w_i \) in (8) (equivalently, if \( f \) encodes a symbol of the peripheral \( X_l..X_r \)).

Let \( f_1 \) be the first (or leftmost) \( w_i \)-index from above in the index-stack \( \mu \), and let \( w_1 \) be the subword of \([x_m]\) containing the terminal into which \( f_1 \) is discharged, i.e., all other \( w_i \)-indices in \( \mu \) are only accessible after \( f_1 \) has been consumed. Thus, for \( \mu = \pi_1 f_1 \cdots f_n \) we get from (7)

\[
M_\pi f_1 \cdots f_n \gamma \rightarrow u_1 w_0 v_1 \cdots v_n \gamma
\]

The path \( P_1 \) from \( M_\mu \) to \( w_1 \) contains \( n \) \( B \)-nodes, for \( k=2^n \) in (8). For every \( B \)-node \( B_j \) (\( 0 \leq j \leq n \)) of \( P_1 \), we obtain because of the index-multiplication effect by nonterminal branching:

\[
B_j[x_1 f_1 \cdots f_n \gamma] \rightarrow L_j[x_1 f_1 \cdots f_n \gamma] R_j[x_1 f_1 \cdots f_n \gamma] \\
L_j[x_1 f_1 \cdots f_n \gamma] \rightarrow \ast \rightarrow u_j B_j[1] \cdots [1]+f_1 \gamma \ast+1 \\
(B_j B_j+1, L_j, R_j \in N, \tau_j, \tau_{j+1} \in \alpha, \beta, \{,\})^*)
\]

Every path \( P_j \) branching off from \( P_1 \) at \( B_j[x_1 f_1 \cdots f_n \gamma] \) leads to a word \( w_j \) derived exclusively by discharging \( w_i \)-indices situated in \( \mu \) below (or on the right side of) \( f_1 \) and \( f \)-index symbols situated in \( \mu \) below (or on the right side of) \( f_1 \). Consequently, \( f_1 \) has to be deleted on every such path \( P_j \), before the appropriate indices become accessible, i.e., we get for every \( j \) with \( 0 \leq j < n \):

\[
B_j[x_1 f_1 \cdots f_n \gamma] \rightarrow u_j R_j[x_1 f_1 \cdots f_n \gamma] \ast \rightarrow \ast \rightarrow y_j C_j[x_1 f_1 \cdots f_n \gamma] z_j \\
(B_j, R_j, C_j \in N, \tau_j, \sigma \in \alpha, \beta, f_j \in F)
\]

Thus, for \( n \geq |N| \) in (8) \( |N| \) the cardinality of the nonterminal alphabet \( N \) of \( G_I \), ignoring, as before the constant amount of parenthesis-storing in nonterminals) because of \( |x_1| \) \( P \)-nodes occur twice on different paths branching off from \( P_1 \), i.e., there exist \( p, q (0 \leq p < q < n) \) such that:

\[
M_\pi f_1 \cdots f_n \gamma \rightarrow u_p R_p[x_1 f_1 \cdots f_n \gamma] v_q R_q[x_1 f_1 \cdots f_n \gamma] \text{ and} \\
R_p[x_1 f_1 \cdots f_n \gamma] \rightarrow \ast \rightarrow y_p C_p[x_1 f_1 \cdots f_n \gamma] z_p \rightarrow \ast \rightarrow y_p z_p \\
R_q[x_1 f_1 \cdots f_n \gamma] \rightarrow \ast \rightarrow y_q C_q[x_1 f_1 \cdots f_n \gamma] z_q \rightarrow \ast \rightarrow y_q z_q
\]

\[
(\mu = \pi_1 f_1 \cdots f_n, \sigma, \tau_j, \tau_{j+1} \in \alpha, \beta, C_j \in C \cap N; u_p, v_q \in \gamma; z_p \in z \in \gamma)
\]

I.e., \( G_I \) generates words \( \gamma = x_1 \) \( x_1 \) \( x_1 \) \( x_1 \) \( x_1 \) \( x_1 \), the central part of which contains a duplication (of "z" in \([x_m]\) = \( y_1 z y_2 z y_3 \) without correspondence in \( x_1 \) or \( x_r \), thus contradicting the general form of words of \( L_3 \). Hence \( L_3 \) is not indexed.

2.2 DI-Grammars and Linear Indexed Grammars\(^1\)

As already mentioned above, Gazdar in (Gazdar, 1988) introduced and discussed a grammar formalism, afterwards (e.g., in (Weir and Joshi, 1988)) called linear indexed grammars (LIG’s), using index stacks in which only one nonterminal on the right-hand-side of a rule can inherit the stack from the left-hand-side, i.e., the rules of a LIG \( G=(N,T,F,P,S) \) with \( N,F,T,S \) as above, are of the Form

- \( A[l..f..] \rightarrow A[l..f..] \rightarrow a \)

\[
\text{where } A_1,..,A_n \in N, f, e \in F, \text{ and } a \in T \cup \{e\}. \text{ The "derives"-relation } \rightarrow \text{ is defined as follows}
\]

\[
\gamma \in A[l..f..] \beta \rightarrow \alpha A[l..f..] \beta \rightarrow \alpha A[l..f..] \beta
\]

\[
\beta \text{ if } A[l..f..] \rightarrow A[l..f..] \rightarrow A[l..f..] \rightarrow A[l..f..] \beta
\]

\( \)\(^1\)Thanks to the anonymous referees for suggestions for this section and the next one.

361
\[ \alpha I[f_1 \ldots f_n] \beta \Rightarrow \alpha A[I \ldots A[I[f_1 \ldots f_n] \ldots A_n] \beta \]
\[ \text{if } A[I] \Rightarrow A[I[I[f_1 \ldots f_n] \ldots A_n] \beta \]
\[ \alpha I[f_1 \ldots f_n] \beta \Rightarrow \alpha A[I \ldots A[I[f_1 \ldots f_n] \ldots A_n] \beta \]
\[ \text{if } A[I] \Rightarrow A[I[I[f_1 \ldots f_n] \ldots A_n] \beta \]
\[ \alpha I] \Rightarrow \alpha \alpha \beta \]
\[ \Rightarrow \text{ is the reflexive and transitive closure of } \Rightarrow, \text{ and } \]
\[ L(G) = \{ w ; w \in T^* \land S[I] \Rightarrow w \}. \]

Gazdar has shown that LIGs are a (proper) subclass of indexed grammars. Joshi, Vijay-Shanker, and Weir (Joshi, Vijay-Shanker, and Weir, 1989; Weir and Joshi, 1988) have shown that LIGs, Combinatory Categorial Grammars (CCG), Tree Adjoining Grammars (TAGs), and Head Grammars (HG) are weakly equivalent. Thus, if an inclusion relation can be shown to hold between DIL-languages (DIL) and LILs, it simultaneously holds between the DIL-class and all members of the family.

To simulate the restriction on stack transmission in a LIG GI = (N1, T, F1, P1, S1) the following construction of a DIL-grammar Gd suggests itself:

Let Gd = (N, T, F, P, S) where N = {S}, F = {f; f \in F1} \cup \{\#\}, and P = {S \Rightarrow S' \#}.

\[ \begin{align*}
\cup & \{A' \Rightarrow A_1 # \ldots A_n # \ldots A_n \Rightarrow A[I[I[f_1 \ldots f_n] \ldots A_n] \beta \} \\
\cup & \{A \Rightarrow A[I[I[f_1 \ldots f_n] \ldots A_n] \beta \} \\
\cup & \{A' \Rightarrow A_1 # \ldots A_n # [f[I[I[f_1 \ldots f_n] \ldots A_n] \beta \} \\
\end{align*} \]

It follows by induction on the number of derivation steps that for X e N, X e T*, \mu e F*, \mu e F1*, and w e T*

(10) X' \mu \Rightarrow w if and only if X[\mu] \Rightarrow w

where X' = h(X) and \mu = h(\mu) (h is the homomorphism from (N0, F) into (N1, F) with h(Z) = Z'). For the nontrivial part of the induction, note that A' # \mu' can not be terminated in G.

Together with S \Rightarrow S' \# (10) yields L(Gd) = L(G).

The inclusion of the LIG-class in the DIL-class is proper, since L3 above is not a LIG-language, or to give a more simple example:

Lw = \{a^n b_1 a_1 b_1 a_2 b_2 \ldots b_n a_2 b_2 \ldots b_n a_1 b_1 \mid n = n_1 + n_2 \} is according to (Vijay-Shanker, Weir and Joshi, 1987) not in TAL, hence not in LIL. But (the indexed language) Lw is generated by the DIL-Grammar

\[ Gw = (S, A, B, \{a, a_1 b_1 a_2 b_2, \ldots b_n a_2 b_2 \ldots b_n a_1 b_1 \mid n = n_1 + n_2 \}) \]

The relation of DIL-grammars to Steedman’s Combinatory Categorial Grammars with Generalized Compositions (GC-CCG for short) in the sense of (Weir and Joshi, 1988) is not so easy to determine. If for each n \geq 1 composition rules of the form

(11) (x y) \ldots (y z) \Rightarrow (x y) \ldots (y z) \ldots (x y) \ldots (y z) \ldots (x y) \\

are permitted, the generative power of the resulting grammars is known to be stronger than TAGs (Weir and Joshi, 1988).

Now, the GC-CCG given by

f(a) = \{#\} f(a1) = \{S/X\\#, X/X\\#, #/X#/X#/X#\}

f(a) = \{A, X A\} f(b1) = \{S/Y\\#, Y/Y\\#, #/Y#/Y#/Y#\}

f(b) = \{B, Y B\} f(I) = \{K\} f(I) = \{##/K, #/K\}

generates a language Lc, which when intersected with the regular set

\[ \{a, b\} + \{(a, b) + a, b\} + \]

yields a language Lp which is for similar reasons as L3 not even an indexed language. But Lp does not seem to be a DIL-language either. Hence, since indexed languages and DIL-languages are closed under intersection with regular sets, Lc is neither an indexed nor (so it appears) a DI-language.

The problem of a comparison of DIL-grammars and GC-CCGs is that, in spite of all appearances, the combination of generalized forward and backward composition can not directly simulate nor be simulated by index-distribution, at least so it seems.

3 DI-Automata

An alternative method of characterizing DIL-languages is by means of DI-automata defined below.

**DI-automata (dia)** have a remote resemblance to Aho's nested stack automata (nsa). They can best be viewed as push down automata (pda) with additional power: they can not only read and write on top of their push down store, but also travel down the stack and (recursively) create new embedded substacks (which can be left only after deletion). dias and nsa's differ in the following respects:

1. a dia is only allowed to begin to travel down the stack or enter the stack reading mode, if a tape-symbol A on top of the stack has been deleted and stored in a special stack-reading state qA, and the stack-reading mode has to be terminated as soon as the first index-symbol f from above is being scanned, in which case the index-symbol concerned is deleted and an embedded stack is created, provided the transition-function gives permission. Thus, every occurrence of an index-symbol on the stack can only be "consumed" once, and
only in combination with a "matching" non-index-symbol.

A nsa, on the other hand, embeds new stacks behind tape symbols which are preserved and can, thus, be used for further stack-embeddings. This provides for part of the stack multiplication effect.

2. Moving through the stack in the stack reading mode, a dia is not allowed to pass or skip an index symbol. Moreover, no scanning of the input or change of state is permitted in this mode.

A nsa, however, is allowed both to scan its input and change its state in the stack reading mode, which, together with the license to pass tape symbols repeatedly, provides for another part of the stack multiplication effect.

3. Unlike a nsa, a dia needs two tape alphabets, since only "index symbols" can be replaced by new stacks, moreover it requires two sets of states in order to distinguish the pushdown mode from the stack reading mode.

Formally, a di-automaton is a 10-tuple

\[ D = (q, Q_F; T, \Gamma, I, S, Z_0, \#; \delta; \epsilon; \#) \]

where \( q \) is the control state for the pushdown mode, \( Q_F \) a finite set of stack reading states, \( T \) a finite set of input symbols, \( \Gamma \) a finite set of storage symbols, \( I \) a finite set of index symbols where \( I \cap \Gamma = \emptyset \), \( Z_0 \) is the initial storage symbol, \# is the top-of-stack marker on the storage tape, \( \epsilon \) is the bottom-of-embedded stack marker on the storage tape, and \( \delta \) is a mapping

1) in the push down mode:

- from \( \{ q \} \times T^* \times S \Gamma^* \) into finite subsets of \( \{ q \} \times D \times S \Gamma^* \)

2) in the stack reading mode: for every \( A \in \Gamma \)

- from \( \{ q_A \} \times T^* \times \Gamma^* \) into subsets of \( \{ q_A \} \times \{ 0 \} \times \{ 1 \} \)

- from \( \{ q \} \times T^* \times \{ A \} \) into subsets of \( \{ q \} \times \{ 0 \} \times \{ 1 \} \)

3) in the stack creation mode:

- from \( \{ q \} \times T^* \times I \) into finite subsets of \( \{ q \} \times \{ 0 \} \times \{ A \} \times \{ I \} \times \{ \epsilon \} \times \{ \# \} \)

As in the case of Aho's nested stack automaton a configuration of a DI-automaton \( D \) is a quadruple \( (p, a_1, ..., a_n, \#) \), where

1. \( p \in \{ q \} \cup Q_F \) is the current state of \( D \);
2. \( a_1, ..., a_n \) is the input string; \# the input endmarker;
3. \( i (1 \leq i \leq n + 1) \) the position of the symbol on the input tape currently being scanned by the input head \( (=a_i) \);
4. \( X_1, ..., X_m \) the content of the storage tape where for \( m > 1 \), \( X_1 = S_A, A \epsilon I, X_m = \#, X_2, ..., X_{m-1} \in (I \cup \Gamma \cup \{ Z_0, \# \}) \); \( X_i \) is the stack symbol currently being read by the storage tape head. If \( m = 1 \), then \( X_m = \# \).

As usual, a relation \( \Rightarrow_D \) representing a move by the automaton is defined over the set of configurations:

(i) \( (q, a_1, ..., a_n, \#, i, X_1, ..., X_m) \Rightarrow_D (q, a_1, ..., a_n, \#, i+1, X_1, ..., X_m) \)

(ii) \( (q, a_1, ..., a_n, \#, i, X_1, ..., X_m) \Rightarrow_D (q, a_1, ..., a_n, \#, i+1, X_1, ..., X_m) \)

As in the case of Aho's nested stack automaton a configuration of a DI-automaton \( D \) is a quadruple \( (p, a_1, ..., a_n, \#) \), where

- from \( \{ q \} \times T^* \times \{ \# \} \) into subsets of \( \{ q \} \times \{ 0 \} \)

To illustrate, the DI-automaton \( D_I \) accepting \( L_3 \) by empty stack is specified:

\[ D_I = (q, Q_F; T, \Gamma, I, S, Z_0, \#; \delta; \epsilon; \#) \]

where for every \( x \in T \):

\[ \delta(q, x, S \#) = \{ (q, 0, S a_1), (q, 0, S b_1) \} \]

\[ \delta(q, x, M) = \{ (q, 0, S \#) \} \]

(For the \( G_3 \) rules: \( S \rightarrow a S a, S \rightarrow b S b, S \rightarrow M \))

To illustrate, the DI-automaton \( D_I \) accepting \( L_3 \) by empty stack is specified:

\[ D_I = (q, Q_F; T, \Gamma, I, S, Z_0, \#; \delta; \epsilon; \#) \]

where for every \( x \in T \):

\[ \delta(q, x, S \#) = \{ (q, 0, S a_1), (q, 0, S b_1) \} \]

\[ \delta(q, x, M) = \{ (q, 0, S \#) \} \]
(for: M→[M], M→MM, M→Z)

δ(q,x,5x) = (q,1,§)

(i.e.: if input symbol x = "predicted" terminal symbol x, then shift input-tape one step ("1") and delete successful prediction" (replace 5x by §))

δ(q,x,5Z) contains {(qZ,0,§)},

(i.e.: change into stack reading mode in order to find indices belonging to the nonterminal Z)

δ(qZ,x,5§) = {(qz,0,§)},

(i.e.: seek first index-symbol belonging to Z inside the stack)

δ(qZ,x,5y) = {(qz,1,§)},

(i.e. simulate the index-rules Zf~Za, Zf~a by creation of embedded stacks)

δ(q,x,5¢) = {(q,1,§)},

(i.e. delete empty sub-stack)

δ(q,x,5) = {(q,0,1)},

(i.e. move to top of (sub-)stack).

The following theorem expresses the equivalence of DI-grammars and DI-automata

(11) DI-THEOREM: L is a DI-language (i.e. L is generated by a DI-grammar) if and only if L is accepted by a DI-automaton.

Proof sketch:

I. "only if":(to facilitate a comparison this part follows closely Aho's corresponding proof for indexed grammars and nsa's (theorem 5.1) in (Aho, 1969))

If L is a DI-language, then there exists a DI-grammar G=(N,T,F,P,S) with L(G)=L. For every DI-grammar an equivalent DI-grammar in a normal form can be constructed in which each rule has the form A---~BC, A-~a, A---~Bf or Af---~B, with A•N; B,C•(N-{S}), a•T, f~F; and e • L(G), only if S---~e is in P. (The proof is completely analogous to the corresponding one for indexed grammars in (Aho, 1968) and is therefore omitted).

Thus, we can assume without loss of generality that G is in normal form.

A DI-automaton D such that N(D)=L(G) is constructed as follows:

Let D=(q, QI7T, IS, l,d,,Z~$,¢,#), with T=N~T~{$,¢,#}, QI~{qA;Ae2-~, I=F, Zo=S

where is constructed in the following manner for all a•T:

. {q,1,§)}

(i.e. simulate the index-rules Zf~Za, Zf~a by creation of embedded stacks)

δ(q,x,5¢) = {(q,0,1)},

(i.e. delete empty sub-stack)

δ(q,x,5) = {(q,0,1)},

(i.e. move to top of (sub-)stack).

The following theorem expresses the equivalence of DI-grammars and DI-automata

(11) DI-THEOREM: L is a DI-language (i.e. L is generated by a DI-grammar) if and only if L is accepted by a DI-automaton.

Proof sketch:

I. "only if":(to facilitate a comparison this part follows closely Aho's corresponding proof for indexed grammars and nsa's (theorem 5.1) in (Aho, 1969))

If L is a DI-language, then there exists a DI-grammar G=(N,T,F,P,S) with L(G)=L. For every DI-grammar an equivalent DI-grammar in a normal form can be constructed in which each rule has the form A---~BC, A-~a, A---~Bf or Af---~B, with A•N; B,C•(N-{S}), a•T, f~F; and e • L(G), only if S---~e is in P. (The proof is completely analogous to the corresponding one for indexed grammars in (Aho, 1968) and is therefore omitted).

Thus, we can assume without loss of generality that G is in normal form.

A DI-automaton D such that N(D)=L(G) is constructed as follows:

Let D=(q, QI7T, IS, l,d,,Z~$,¢,#), with I=1~N~(S,¢,#), QI=(qA|AdF), l=F, Z~$ where is constructed in the following manner for all a•T:

1. (q,0,5Bc)εδ(q,a,5A), if A→BC ∈ P,
2. (q,1,§)εδ(q,a,5A),
3. (qA,0,§)εδ(q,a,5A) for all A ∈ Γ,
4. (qA,1,§)εδ(q,a,5A,B) for all A ∈ Γ and all B∈Γ,
5. i.(q,0,5Bc)εδ(qA,5A,5A) and
ii.(q,0,5Bc)εδ(qA,5A,5A) for all A ∈ Γ with Af→B ∈ P,
6. (q,0,§)εδ(q,a,5A)
7. (q,0,-1)εδ(q,a,5A,B) for all B ∈ Γ ~ {$
8. (q,0,§)εδ(q,5A,5A) if and only if S→e is in P.

LEMMA 1.1

If

(i) Af1...fk =n=> a1...am
is a valid leftmost derivation in G with k≥0, m≥1 and AεN, then for n≥1, Zβ1...βkε((N∪{$})*, αε(N∪{$})*, με(N∪{F∪{$}):-

(ii) (q,a1...am#,1,α^AZβ1...βkμ#)

σD*(q,a1...am#,m+1,α^Zβ1...βkμ#).

Proof by induction on n (i.e. the number of derivation steps):

If n=1, then (i) is of the form A=>a where aεT and k=0, since only a rule of the form A→a can be applied because of the normal form of G and since in DI-grammars (unlike in indexed grammars) unconsumed indices can not be swallowed up by terminals. Because of the construction of δ, (ii) is of the form

(q,a#,1,α^AZβ1...βkμ#)

σD(q,a#,1,α^AZβ1...βkμ#)

σD(q,a#,2,α^AZβ1...βkμ#)

=σ(q,a#,2,α^AZβ1...βkμ#).

Suppose Lemma 1.1 is true for all n<n' with n'<1.

A leftmost derivation Af1...fk =n'=a1...am can have the following three forms according as A is expanded in the first step:

1)Af1...fk±1...fk =n'=Bf1...fCf1...fk

2)Af1...fk =n'=a1...am with n'<n.

3)Af1...fk =n'=Bf2...fkm =n'=a1...am
with n′<n and (Af1→B∈P).

From the inductive hypothesis and from 1.-8. above, it follows

1') (q,a1...am#,1,α^AZβ1...βkμ#)

σD(q,a1...am#,1,α^AZβ1...βkμ#)

σD(q,a1...am#,i+1,α^AZβ1...βkμ#)

σD*(q,a1...am#,i+1,α^AZβ1...βkμ#)

σD*(q,a1...am#,m+1,α^Zβ1...βkμ#).
LEMMA 1.2

If for $Z_1^{1}...Z_k^{1} e (N \cup \{\varepsilon}\)^*, $a \in (N \cup \{\varepsilon}\)^*, and $m \geq 1$, $\mu e (N \cup \{\varepsilon\}^*)$,

$$D(q_0,a_1...a_m^\#,a_1...a_m^\#,1,\varepsilon^{Z_1^{1}...Z_k^{1}})$$

then in G the derivation is valid

$A_1^{1}...A_k^{1}=\Rightarrow a_1...a_m^\#$

II.2: If

$A_1^{1}...A_k^{1}=\Rightarrow a_1...a_m^\#$

is a leftmost derivation in G, then the following transition of $D$ is valid

$$D^+(q_0,a_1...a_m^\#,1,\varepsilon^{A_1^{1}...A_k^{1}})$$

The proofs by induction of I.1 and II.2 (unlike the proofs of the corresponding lemmata for nsa's and indexed grammars (s.Aho, (1969)) are as elementary as the one given above for I.1 and are omitted.

The DI-automaton concept can be used to show the inclusion of the class of DI-languages in the class of context-sensitive languages. The proof is structurally very similar to the one given by Aho (Aho, 1968) for the inclusion of the indexed class in the context-sensitive class: For every DI-automaton $A$, an equivalent DI-automaton $A'$ can be constructed which accepts its input $w$ if and only if $A$ accepts $w$ and which in addition uses a stack the length of which is bounded by a linear function of the length of the input $w$. For $A'$ a linear bounded automaton $M$ (i.e. the type of automaton characteristic of the context-sensitive class) can be constructed which simulates $A'$. For reasons of space the extensive proof can not be given here.

4 Some Remarks on the Complexity of DI-Recognition

The time complexity of the recognition problem for DI-grammars will only be considered for a subclass of DI-grammars. As the restriction on the form of the rules is reminiscent of the Chomsky normal form for context-free grammars (CFG), the grammars in the subclass will be called DI-Chomsky normal form (DI-CNF) grammars. A DI-grammar $G=(N,T,F,P,S)$ is a DI-CNF grammar if and only if each rule in $P$ is of one of the following forms where $A,B,C \in N-\{S\}$, $f \in F$, $a \in T$, $S \rightarrow a$, $f \notin L(G)$,

(a) $A \rightarrow BC$, (b) $A \rightarrow BF$, (c) $A \rightarrow BCf$, (d) $A \rightarrow BF$, (e) $A \rightarrow a$

The question whether the class of languages generated by DI-CNF grammars is a proper or improper subclass of the DI-languages will be left open.

In considering the recognition of DI-CNF grammars an extension of the CKY algorithm for CFGs (Kasami, 1965; Younger, 1967) will be used which is essentially
inspired by an idea of Vijay-Shanker and Weir in (Vijay-Shanker and Weir, 1991).

Let the \(n(n+1)/2\) cells of a CKY-table for an input of length \(n\) be indexed by \(i\) and \(j\) (\(1 \leq i \leq n\)) in such a manner that cell \(Z_{i,j}\) builds the top of a pyramid the base of which consists of the input \(a_1...a_n\).

As in the case of CFGs a label \(E\) of a node of a derivation tree (or a code of \(E\)) should be placed into cell \(Z_{i,j}\) only if in \(G\) the derivation \(E = \ast \Rightarrow a_1...a_j\) is valid. Since nonterminal nodes of DI-derivation trees are labeled by pairs \((A, \mu)\) consisting of a nonterminal \(A\) and an index stack \(\mu\) and since the number of such pairs with \((A, \mu) = \ast \Rightarrow w\) can grow exponentially with the length of \(w\), intractability can only be avoided if index stacks can be encoded in such a way that substacks shared by several nodes are represented only once.

Vijay-Shanker and Weir solved the problem for linear indexed grammars (LIGs) by storing for each node \(K\) not its complete label \(A_f f_2...f_n\), but the nonterminal part \(A\) together with only the top \(f_1\) of its index stack and an indication of the cell where the label of a descendant of \(K\) can be found with its top index symbol continuing the stack of its ancestor \(K\). In the following this idea will be adopted for DI-grammars, which, however, require a supplementation.

Thus, if the cell \(Z_{i,j}\) of the CKY-table contains an entry beginning with \(\mu = \ast \Rightarrow a_1...a_j\) with \(\mu = f_1 f_2...f_n \in \mathbb{F}\) is valid, and further that the top index symbol \(f_2\) on \(\mu_1\) (i.e. the continuation of \(f_1\)) is in an entry of cell \(Z_{p,q}\) beginning with the nonterminal \(B\). If, descending in such a manner and guided by pointer quadruples like 
\[
\langle B, f_2, p, q, \rangle,
\]

an entry of the form \(\langle C, f_n, \rangle,..\rangle\) is met, then, in the case of a LIG-table, the bottom of stack has been reached. So, entries of the form 
\[
\langle A, f_1, (B, f_2, p, q) \rangle
\]

are sufficient for LIGs.

But, of course, in the case of DI-derivations the bottom of stack of a node, because of index distribution, does not coincide with the bottom of stack of an arbitrary index inheriting descendant, cf.

Rather, the bottom of stack of a DI-node coincides with the bottom of stack of its rightmost index inheriting descendant. Therefore, the pointer mechanism for DI-entries has to be more complicated. In particular, it must be possible to add an "intersemial" pointer to a sister path. However, since the continuation of the unary stack (like of \(C_f\) in (13)) of a node without index inheriting descendants is necessarily unknown at the time its entry is created in a bottom up manner, it must be possible to add an intersemial pointer to an entry later on.

That is why a DI-entry for a node \(K\) in a CKY-cell requires an additional pointer to the entry for a descendant \(C\), which contains the end-of-stack symbol of \(K\) and which eventually has to be supplemented by an intersemial continuation pointer. E.g. the entry

\[
\langle B_1, f_2, (D, f_3, p, q), (C, f_4, r, s) \rangle \in Z_{i,j}
\]

indicates that the next symbol \(f_3\) below \(f_2\) on the index stack belonging to \(B_1\) can be found in cell \(Z_{p,q}\) in the entry for the nonterminal \(D\); the second quadruple \((C, f_4, r, s)\) points to the descendant \(C\) of \(B_1\) carrying the last index \(f_1\) of \(B_1\) and containing a place where a continuation pointer to a neighbouring path can be added or has already been added.

To illustrate the extended CKY-algorithm, one of the more complicated cases of specifying an entry for the cell \(Z_{i,j}\) is added below which dominates most of the other cases in time complexity:

\[
\begin{align*}
&\text{FOR } i := n \text{ TO } 1 \text{ DO} \\
&\text{FOR } j := 1 \text{ TO } n \text{ DO} \\
&\quad \text{FOR } k := i \text{ TO } j-1 \text{ DO} \\
&\quad\quad \text{ FOR each rule } A \Rightarrow A_1 A_2; \\
&\quad\quad\quad \text{ if } \langle A_1, f_1, (B_1, f_2, p_1, q_1), (C_1, f_3, s_1, t_1) \rangle \in Z_{i,k} \\
&\quad\quad\quad\quad \text{ for some } B_1, C_1 \in N, f_1, f_2 \in \mathbb{F}, \\
&\quad\quad\quad\quad\quad p_1, q_1 \text{ (is } i \leq \mu_1 \leq k \text{ and } q_1 < \mu_1), \\
&\quad\quad\quad\quad\quad s_1, t_1 \text{ (is } i \leq \mu_1 \leq k \text{ and } s_1 < \mu_1), \\
&\quad\quad\quad\quad\quad \text{ and } \langle A_2, f_2, \rangle \in \mathbb{F} \\
&\quad\quad\quad\quad\quad \text{ then } 1. \text{ if } \langle B_1, f_1, (B_2, f_2, p_2, q_2), X \rangle \in \mathbb{F} \\
&\quad\quad\quad\quad\quad\text{ for some } B_2 \in N, f_2 \in \mathbb{F}, p_2, q_2 \text{ with } i \leq \mu_2 \leq k, \text{ and if } q_1 \leq p_2, \text{ then } \\
&\quad\quad\quad\quad\quad\quad \quad \quad X = \text{ } (C, f_0, u, v) \text{ for some } C \in N, f_0 \in \mathbb{F}, u, v \text{ (is } i \leq \mu_2 \leq q_1), \\
&\quad\quad\quad\quad\quad\quad \quad \quad \text{ then } \langle A_2, f_2, \rangle \in \mathbb{F} \text{ for some } f_2 \in \mathbb{F} \text{ and } \\
&\quad\quad\quad\quad\quad\text{ else } \langle C_1, f_3, \rangle \in \mathbb{F} \\
&\quad\quad\quad\quad\quad\quad\quad \quad \text{ for some } \langle C_1, f_3, \rangle \in \mathbb{F} \text{ (is } i \leq \mu_1 \leq k, \text{ and } \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \quad \quad \langle A_2, f_2, \rangle \in \mathbb{F} \text{ (is } i \leq \mu_1 \leq k, \text{ and } \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \langle A_2, f_2, \rangle \in \mathbb{F} \text{ (is } i \leq \mu_1 \leq k, \text{ and } }
\end{align*}
\]

(14) 

LIG-Percolation vs. DI-Percolation

\[
\begin{align*}
&\text{LIG-Percolation} \quad A_f f_2...f_n \\
&\quad / \quad / \\
&\quad B_f f_2...f_n \quad B_2 \\
&\quad / \quad / \\
&\quad C_f (\text{bottom of stack}) \\
&\quad / \quad / (\text{stack continuation}) \\
&\quad D_f f_2...f_n \\
&\quad / \quad / \\
&\quad B_f f_2...f_n \quad B_2 f_2...f_n \\
&\quad / \quad / \\
&\quad C_f (\text{bottom of stack}) \\
&\quad / \quad / (\text{stack continuation})
\end{align*}
\]
then
\[ Z_{s1,t1}=Z_{s1,t1}\cup\{<C_1,f_3,(A_2,f_2,k+1,j),->\} \]

The pointer \((A_2,f_2,k+1,j)\) in the new entry of \(Z_{i,j}\) points to the cell of the node where the end of stack of the newly created node with nonterminal \(A\) can be found. The same pointer \((A_2,f_2,k+1,j)\) appears in cell \(Z_{s1,t1}\) as "supplement" in order to indicate where the stack of \(A\) is continued behind the end-of-stack of \(A_1\). Note that supplemented quadruples of a cell \(Z_{i,j}\) are uniquely identifiable by their form \(<N,f_1,(C,f_2,r,s),->\), i.e. the empty fourth component, and by the relation \(j_s=r\). Supplemented quadruples cannot be used as entries for daughters of "active" nodes, i.e. nodes the entries of which are currently being constructed.

Let \(a_1...a_n\) be the input. The number of entries of the form \(<B,f_1,(D,f_2,p,q),(C,f_3,r,s)\>\) \((f_1,f_2,f_3\in F, B,C,D\in N, 1\leq p,q,r,s, j\leq n)\) in each cell \(Z_{i,j}\) will then be bounded by a polynomial of degree 4, i.e. \(O(n^4)\). For a fixed value of \(i,j,k\), steps like the one above may require \(O(n^8)\) time (in some cases \(O(n^{12})\)). The three initial loops increase the complexity by degree 3.

References

[Aho, 1968] A. V. Aho. Indexed Grammars, J. Ass. Comput. Mach. 15, 647-671, 1968.
[Aho, 1969] A. V. Aho. Nested Stack Automata, J. Ass. Comp. Mach. 16, 383-, 1969.
[Gazdar, 1988] G. Gazdar. Applicability of Indexed Grammars to Natural Languages, in: U.Reyle and C.Rohrer (eds.):Natural Language Parsing and Linguistic Theories, 69-94, 1988.
[Joshi, Vijay-Shanker, and Weir, 1989] A. K. Joshi, K. Vijay-Shanker, and D. J. Weir. The convergence of mildly context-sensitive grammar formalisms. In T. Wasow and P. Sells (Eds.), The processing of linguistic structure. MIT Press, 1989.
[Kasami, 1965] T. Kasami. An efficient recognition and syntax algorithm for context-free languages. (Tech. Rep. No. AF-CRL-65-758). Bedford, MA: Air Force Cambridge Research Laboratory, 1965.
[Pereira, 1981] F. Pereira. Extrapolation Grammars, in: American Journal of Computational Linguistics, 7, 243-256, 1981.
[Pereira, 1983] F. Pereira. Logic for Natural Language Analysis, SRI International, Technical Note 275, 1983.

[Rizzi, 1982] L. Rizzi. Issues in Italian Syntax, Dordrecht, 1982.
[Stabler, 1987] E. P Stabler. Restricting Logic Grammars with Government-Binding Theory, Computational Linguistics, 13, 1-10, 1987.
[Takeshi, 1973] Hayashi Takeshi. On Derivation Trees of Indexed Grammars. Publ.RIMS, Kyoto Univ., 9, 61-92, 1973.
[Vijay-Shanker, Weir, and Joshi, 1986] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Tree adjoining and head wrapping. 11th International Conference on Comput. Ling. 1986.
[Vijay-Shanker, Weir, and Joshi, 1987] K. Vijay-Shanker, D. J. Weir, A. K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. 25th Meeting Assoc.Comput. Ling., 104-111. 1987.
[Vijay-Shanker and Weir, 1991] K. Vijay-Shanker and David J. Weir. Polynomial Parsing of Extensions of Context-Free Grammars. In: Tomita, M.(ed.) Current Issues in Parsing Technology, 191-206, London 1991.
[Weir and Joshi, 1988] David J. Weir and Aravind K. Joshi. Combinatory Categorial Grammars: Generative power and relationship to linear context-free rewriting systems. 26th Meeting Assoc.Comput. Ling., 278-285, 1988.
[Younger, 1967] D. H. Younger. Recognition and parsing context-free languages in time \(n^3\). Inf. Control, 10, 189-208.