Chapter 1

Global Index Grammars and Descriptive Power

JOSÉ M. CASTAÑO

ABSTRACT. We review the properties of Global Index Grammars (GIGs), a grammar formalism that uses a stack of indices associated with productions and has restricted context-sensitive power. We show how the *control* of the derivation is performed and how this impacts in the descriptive power of this formalism both in the string languages and the structural descriptions that GIGs can generate.

1.1 Introduction

The notion of *mild context-sensitivity* was introduced in Joshi (1985) as a possible model to express the required properties of formalisms that might describe Natural Language (NL) phenomena. It requires four properties:[1] a) *constant growth* property (or the stronger *semilinearity* property); b) polynomial parsability; c) *limited cross-serial* dependencies, i.e. some limited context-sensitivity d) proper inclusion of context free languages. The canonical NL problems which exceed context free power are: *multiple agreements*, *reduplication*, *crossing dependencies*.[2]

Many formalisms have been proposed to extend the power of context-free grammars using *control* devices, where the control device is a context-free grammar (see Dassow et al. (1997) regarding control languages). The appeal of this approach is that many of the attractive properties of context-free languages may be inherited (e.g. polynomial parsability, semilinearity, closure properties). Those models can be generalized such that additional *control* levels[3] can be added. They

[1]See for example, Joshi et al. (1991), Weir (1988).

[2]However other phenomena (e.g. *scrambling*, Georgian Case and Chinese numbers) might be considered to be beyond certain *mildly context-sensitive* formalisms.

[3]The corresponding automaton models use embedded or an additional constrained stack. In such case, the generalization and hierarchy of levels is obtained using additional levels of embeddedness, or additional stacks (cf. Weir (1988), Cherubini et al. (1996)).

Proceedings of Mathematics of Language 8 2003
R. T. Oehrle & J. Rogers (editors).

form hierarchies of levels of languages, where a language of level k properly includes a language of level $k-1$. For example in Weir (1992), *Mildly Context-sensitive Languages* (MCSLs) are characterized by such a geometric hierarchy of control grammar levels (see also, Khabbaz (1974), Seki et al. (1991), Cherubini et al. (1996)). Those generalizations provide more expressive power but at a computational cost: the complexity of the recognition problem is dependent on the language level: for a MCSL level-k it is in $O(n^{3 \cdot 2^{k-1}})$.

In Castaño (2003), we introduced Global Index Grammars (GIGs) and the corresponding languages, GILs. We presented a Chomsky-Schützenberger representation theorem for GILs. We showed that GIGs have enough descriptive power to capture the three phenomena mentioned above (*reduplication, multiple agreements*, and *crossed agreements*) in their generalized forms. GILs include such languages as $\{ww^+ \ | w \in \Sigma^*\}$, $\{a^n b^m (c^n d^m)^+ \mid n \ m \geq 1\}$ and $\{\ a^n (b^n c^n)^+ \mid n \geq 1\}$ which are beyond the power of Tree Adjoining Languages and beyond the power of any level-k control language. Recognition of the language generated by a GIG is in **bounded** polynomial time: $O(n^6)$, however for bounded state grammars with unambiguous indexing it is $O(n)$.

The equivalent model of automata was presented in Castaño (2003b). Also an algorithm to construct an LR parsing table for GILs was presented there. The automaton model and the grammar can be used to prove that the family of GILs is an Abstract Family of Languages using the same techniques to prove it for CFLs (cf. Castaño (2003b)). GILs have also the semilinear property, a proof can be easily built following the proof presented in Harju et al. (2001) for counter automata. Therefore GILs have at least three of the four properties required for Mildly context sensitivity: a) semi-linearity b) polynomial parsability c) proper inclusion of context free languages. The fourth property, *limited cross-serial dependencies* does not hold of GILs given they contain the MIX (or Bach) language.

The goal of this paper is to show how the properties of GILs are related to the peculiarities of the *control* device that regulates the derivation. Though this mechanism looks similar to the control device in Linear Indexed Grammars (LIGs, cf. Gazdar (1988)), its behavior differs relative to the trees generated by both formalisms.

GIGs offer additional descriptive power as compared to LIGs (and weakly equivalent formalisms) regarding the canonical NL problems mentioned above, and the same computational cost in terms of asymptotic complexity. They also offer additional descriptive power in terms of the structural descriptions they can generate for the same set of string languages, being able to produce *dependent paths*.[4] However those dependent paths are not obtained by encoding the depen-

[4] For the notion of dependent paths see for instance Vijay-Shanker et al. (1987) or Joshi (2000).

dency in the path itself.

This paper is organized as follows: Section 2 reviews Global Index Grammars and their properties, we give examples of its weak descriptive power and we discuss how *control* of the derivation is performed in GIGs. Section 3 discusses the strong descriptive power of GIGs.

1.2 Global Index Grammars

1.2.1 Linear Indexed Grammars

Indexed grammars (IGs, Aho (1968)), and Linear Index Grammars, (LIGs; LILs) Gazdar (1988), have the capability to associate stacks of indices with symbols in the grammar rules. IGs are not semilinear. LIGs are Indexed Grammars with an additional constraint in the form of the productions: the stack of indices can be "transmitted" only to one non-terminal. As a consequence they are semilinear and belong to the class of MCSGs.

A **Linear Indexed Grammar** is a 5-tuple (V, T, I, P, S), where V is the set of variables, T the set of terminals, I the set of $indices$, S in V is the start symbol, and P is a finite set of productions of the form, where $A, B \in V$, $\alpha, \gamma \in (V \cup T)^*$, $i \in I$:

a. $A[..] \to \alpha\ B[..]\ \gamma$ b. $A[i..] \to \alpha\ B[..]\ \gamma$ c. $A[..] \to \alpha B[i..]\ \gamma$

Example 1 $L(G_{wcw}) = \{wcw\ |w \in \{a, b\}^*\}$,
$G_{ww} = (\{S, R\}, \{a, b\}, \{i, j\}, S, P)$ and P is:

1.$S[..] \to aS[i..]$ 2.$S[..] \to bS[j..]$ 3.$S[..] \to cR[..]$ 4.$R[i..] \to R[..]a$
5.$R[j..] \to R[..]b$ 5. $R[] \to \epsilon$

1.2.2 Global Indexed Grammars

GIGs use the stack of indices as a global control structure. This formalism provides a global but restricted context that can be updated at any local point in the derivation. GIGs are a kind of *regulated rewriting* mechanism (cf. Dassow and Păun (1989)) with global context and history of the derivation (or ordered derivation) as the main characteristics of its regulating device. The introduction of indices in the derivation is restricted to productions that are in Greibach normal form (i.e. in which the right hand side starts with a terminal). An additional constraint that is imposed on GIGs is strict leftmost derivation whenever indices are introduced or removed in the derivation.

Definition 1 *A GIG is a 6-tuple $G = (N, T, I, S, \#, P)$ where N, T, I are finite pairwise disjoint sets and 1) N are nonterminals 2) T are terminals 3) I a set of stack indices 4) $S \in N$ is the start symbol 5) $\#$ is the start stack symbol (not in I,N,T) and 6) P is a finite set of productions, having the following form:*

a.1 $A \underset{\epsilon}{\rightarrow} \alpha$	*(epsilon rules)*	*or the equivalent*	$A \rightarrow \alpha$
a.2 $A \underset{[y]}{\rightarrow} \alpha$	*(epsilon with constraints)*	*or in LIG format:*	$[y..]A \rightarrow [y..]\alpha$
b. $A \underset{x}{\rightarrow} a\ \beta$	*(push)*		$[..]A \rightarrow [x..]a\ \beta$
c. $A \underset{\bar{x}}{\rightarrow} \alpha$	*(pop)*		$[x..]A \rightarrow [..]\alpha$

Note the difference between *push* (type b) and *pop* rules (type c): *push* rules require the right-hand side of the rule to contain a terminal in the first position. *Pop* rules do not require a terminal at all. That constraint on *push* rules is a crucial property of GIGs. Derivations in a GIG are similar to those in a CFG except that it is possible to modify a string of indices. We define the *derives* relation $\Rightarrow$ on *sentential forms*, which are strings in $I^*\#(N \cup T)^*$ as follows. Let β and γ be in $(N \cup T)^*$, δ be in I^*, x in I, w be in T^* and X_i in $(N \cup T)$.

1. If $A \underset{\mu}{\rightarrow} X_1...X_n$ is a production of type (a.) (i.e. $\mu = \epsilon$ or $\mu = [x]$, $x \in I$) then:
   $\delta\#\beta A\gamma \underset{\mu}{\Rightarrow} \delta\#\beta X_1...X_n\gamma$ or $x\delta\#\beta A\gamma \underset{\mu}{\Rightarrow} x\delta\#\beta X_1...X_n\gamma$
2. If $A \underset{\mu}{\rightarrow} aX_1...X_n$ is a production of type (b.) or *push*: $\mu = x, x \in I$, then:
   $\delta\#wA\gamma \underset{\mu}{\Rightarrow} x\delta\#waX_1...X_n\gamma$
3. If $A \underset{\mu}{\rightarrow} X_1...X_n$ is a production of type (c.) or *pop* : $\mu = \bar{x}, x \in I$, then:
   $x\delta\#wA\gamma \underset{\mu}{\Rightarrow} \delta\#wX_1......X_n\gamma$

The reflexive and transitive closure of $\Rightarrow$ is denoted, as usual by $\overset{*}{\Rightarrow}$. We define the language of a GIG, G, $L(G)$ to be: $\{w|\#S \overset{*}{\Rightarrow} \#w$ and w is in $T^*\}$

The main difference between IGs, LIGs and GIGs, corresponds to the interpretation of the *derives* relation relative to the behavior of the stack of indices. In IGs the stacks of indices are distributed over the non-terminals of the right-hand side of the rule. In this way the same *control* words can be associated with multiple paths. This allows dependent paths. In LIGs, indices are associated with only one non-terminal at right-hand side of the rule. Thus there is only one stack affected at each derivation step, with the consequence that LILs are semi-linear. GIGs share this *uniqueness* of the stack with LIGs: there is only one stack to be considered per derivation. Unlike LIGs and IGs, the stack of indices is independent of non-terminals in the GIG case. *Push* rules (type b) are constrained to start the right-hand side with a terminal as specified in (6.b) in the GIG definition. The *derives* definition requires a *leftmost* derivation for those productions (*push* and *pop* rules) that affect the stack of indices.

The following example shows that GILs contain a language not contained in LILs, nor in the family of MCSLs. This language is relevant for modeling coordination in Natural Language as observed, for example, in Gazdar (1988).

Example 2 (Multiple Copies) .$L(G_{wwn}) = \{ww^+ \mid w \in \{a,b\}^*\}$

$G_{wwn} = (\{S,R,A,B,C,L\}, \{a,b\}, \{i,j\}, S, \#, P)$ and where P is:

$S \to AS \mid BS \mid C \quad C \to RC \mid L \quad R \underset{\bar{i}}{\to} RA \quad R \underset{\bar{j}}{\to} RB \quad R \underset{[\#]}{\to} \epsilon$

$A \underset{i}{\to} a \quad B \underset{j}{\to} b \quad L \underset{\bar{i}}{\to} La \mid a \quad L \underset{\bar{j}}{\to} Lb \mid b$

The derivation of *ababab*:

$\#S \Rightarrow \#AS \Rightarrow i\#aS \Rightarrow i\#aBS \Rightarrow ji\#abS \Rightarrow ji\#abC \Rightarrow ji\#abRC \Rightarrow i\#abRBC \Rightarrow \#abRABC \Rightarrow \#abABC \Rightarrow i\#abaBC \Rightarrow ji\#ababC \Rightarrow ji\#ababL \Rightarrow i\#ababLb \Rightarrow \#ababab$

The next example shows the MIX (or Bach) language. Gazdar (1988) conjectured the MIX language is not an IL. GILs are semilinear, therefore ILs and GILs are incomparable under set inclusion.

Example 3 (MIX language) .

$L(G_{mix}) = \{w | w \in \{a,b,c\}^* \ and \ |a|_w = |b|_w = |c|_w \geq 1\}$

$G_{mix} = (\{S,D,F,L\}, \{a,b,c\}, \{i,j,k,l,m,n\}, S, \#, P)$ where P is:

$S \to FS \mid DS \mid LS \mid \epsilon \quad F \underset{i}{\to} c \quad F \underset{j}{\to} b \quad F \underset{k}{\to} a$

$D \underset{\bar{i}}{\to} aSb \mid bSa \quad D \underset{\bar{j}}{\to} aSc \mid cSa \quad D \underset{\bar{k}}{\to} bSc \mid cSb \quad D \underset{l}{\to} aSb \mid bSa$

$D \underset{m}{\to} aSc \mid cSa \quad D \underset{n}{\to} bSc \mid cSb \quad L \underset{\bar{l}}{\to} c \quad L \underset{\bar{m}}{\to} b \quad L \underset{\bar{n}}{\to} a$

The following language cannot be generated by LIGs. It is mentioned in Vijay-Shanker et al. (1987) in relation to the definition of composition in Steedman (1985) Categorial Grammars, which permits composition of functions with unbounded number of arguments and generates tree sets with dependent paths.

Example 4 (Dependent branches) .

$L(G_{sum}) = \{\ a^n b^m c^m d^l e^l f^n \mid n = m + l \geq 1\}$,

$G_{sum} = (\{S,R,F,L\}, \{a,b,c,d,e,f\}, \{i\}, S, \#, P)$ where P is:

$S \underset{i}{\to} aSf \mid R \quad R \to F\ L \mid F \mid L \quad F \underset{\bar{i}}{\to} b\ F\ c \mid b\ c \quad L \underset{\bar{i}}{\to} d\ L\ e \mid d\ e$

The derivation of $aabcdeff$:

$\#S \Rightarrow i\#aSf \Rightarrow ii\#aaSff \Rightarrow ii\#aaRff \Rightarrow ii\#aaFLff \Rightarrow i\#aabcLff \Rightarrow \#aabcdeff$

1.2.3 Control of the derivation in LIGs and GIGs

Every LIL can be characterized by a language $L(G, C)$, where G is a labelled grammar (cf. Weir (1992)), $G = (N, T, L, S, P)$, and C is a control set defined by a CFG (a Dyck language):
$\{a_1...a_n| < S, \epsilon > \overset{*}{\Rightarrow} < a_1, w_1 > ... < a_n, w_n >, a_i \in T \cup \{\epsilon\}, w_1, ..., w_n \in C\}$.

In other words, *control* strings w_i are not necessarily *connected* to each other. Those control strings are encoded in the derivation of each *spine* as depicted in figure 1.1 at the left, but every substring encoded in a *spine* has to belong to the control set language. Those control words describe the properties of a path (a *spine*) in the tree generated by the grammar G, and every possible *spine* is independent.

We defined the language of a GIG G, $L(G)$ to be: $\{w|\#S \overset{*}{\Rightarrow} \#w$ and w is in $T^*\}$. We can obtain an explicit control language modifying the derivation as follows. First modify the derives relations such that *pop* productions rewrite the complement of the index: $x\delta\#wA\gamma \underset{\mu}{\Rightarrow} \bar{x}x\delta\#wX_1......X_n\gamma$

Then, define the language of a GIG G, to be the control language $L(G, C)$: $\{w|\#S \overset{*}{\Rightarrow} \delta\#w$ and w is in T^*, δ is in C $\}$, and C is defined to be the Dyck language over the alphabet $I \cup \bar{I}$ (the set of stack indices and their complements).

It is easy to see that no control substring obtained in a derivation subtree is necessarily in the control language, as is depicted in the figure 1.1 at the right. In other words, the control of the derivation can be distributed over different paths, however those paths are connected transversally by the leftmost derivation order.

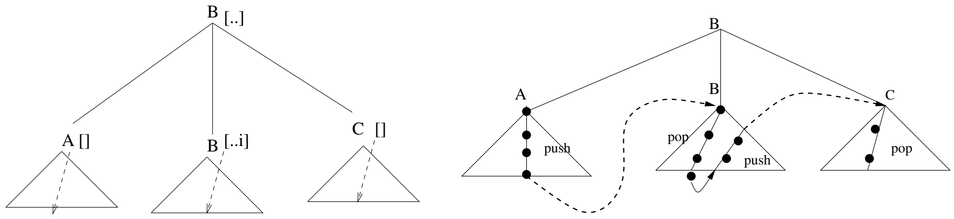

Figure 1.1: LIGs: multiple spines (left) and GIGs: leftmost derivation

1.3 GIGs and structural descriptions

Gazdar (1988) introduces Linear Indexed Grammars and discusses their applicability to Natural Language problems. This discussion is addressed not in terms of weak generative capacity but in terms of strong-generative capacity. Similar approaches are also presented in Vijay-Shanker et al. (1987) and Joshi (2000) (see

Miller (1999) concerning weak and strong generative capacity). In this section we review some of the abstract configurations that are argued for in Gazdar (1988).

1.3.1 The palindrome language

CFGs can recognize the language $\{ww^R|w \in \Sigma^*\}$ but they cannot generate the structural description depicted in figure 1.2 (we follow Gazdar's notation: the left-most element within the brackets corresponds to the top of the stack):

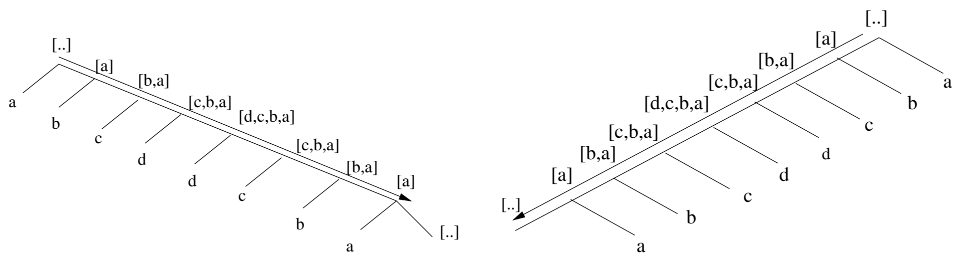

Figure 1.2: Non context-free structural descriptions for the language $\{ww^R|w \in \Sigma\}$ Gazdar (1988)

Gazdar suggests that the configuration at the left would be necessary to represent Scandinavian unbounded dependencies. Such an structure can be obtained using a GIG (and of course a LIG). But the exact mirror image of that structure, (i.e. the structure at the right) cannot be generated by a GIG because it would require *push* productions with a non terminal in the first position of the right-hand side.

However GIGs generate a similar structural description as depicted in figure 1.3 at the left. In such structure the dependencies are introduced in the leftmost derivation order. The English adjective constructions that Gazdar argues can motivate the LIG derivation, are generated by the following GIG grammar. The corresponding structural description is shown in figure 1.3.

Example 5 (Comparative Construction) .

$G_{adj} = (\{AP, NP, \bar{A}, A\}, \{a, b, c\}, \{i, j\}, AP, \#, P)$ *where P is:*

$$AP \to AP\ NP \qquad AP \to \bar{A} \qquad \bar{A} \to \bar{A}\ A$$
$$A \underset{i}{\to} a \qquad A \underset{j}{\to} b \qquad A \underset{k}{\to} c \qquad NP \underset{\bar{i}}{\to} a\ NP$$
$$NP \underset{\bar{j}}{\to} b\ NP \qquad NP \underset{\bar{k}}{\to} c\ NP$$

It should be noted that the operations on indices are reversed as compared to the LIG case shown in right figure of 1.2. On the other hand, it can be noticed also

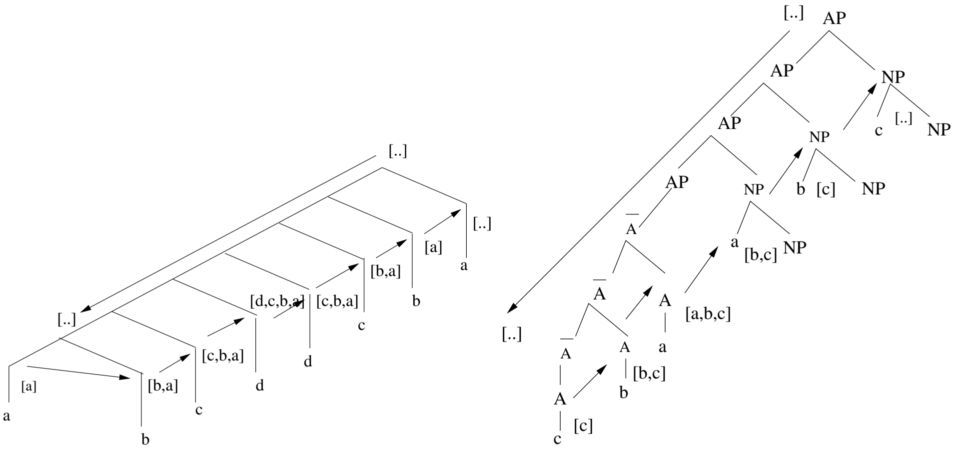

Figure 1.3: GIG structural descriptions for the language ww^R

that the introduction of indices is dependent on the presence of lexical information and its *transmission* is not carried through a top-down *spine*, as in the LIG case. The arrows show the leftmost derivation order that is required by the operations on the stack.

1.3.2 The Copy Language

Gazdar presents the two possible LIG structural descriptions for the copy language depicted in figure 1.4.

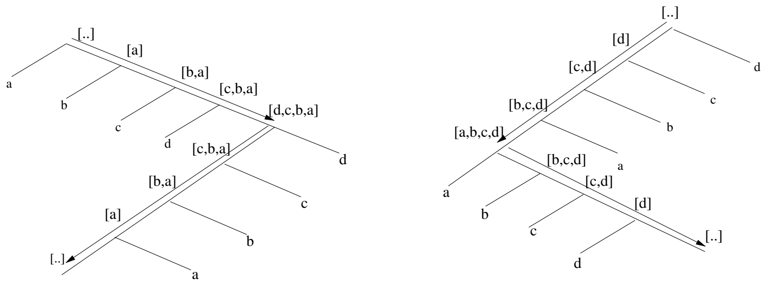

Figure 1.4: LIG structural descriptions of the copy language Gazdar (1988)

The structural description at the left in figure 1.4 can be obtained using GIGs, but not the one at the right. However Gazdar argues that the tree structure shown in figure 1.5 at the left, could be more appropriate for some Natural Language phenomenon that might be modeled with a copy language. Such structure cannot be generated by a LIG, but can be generated by an IG.

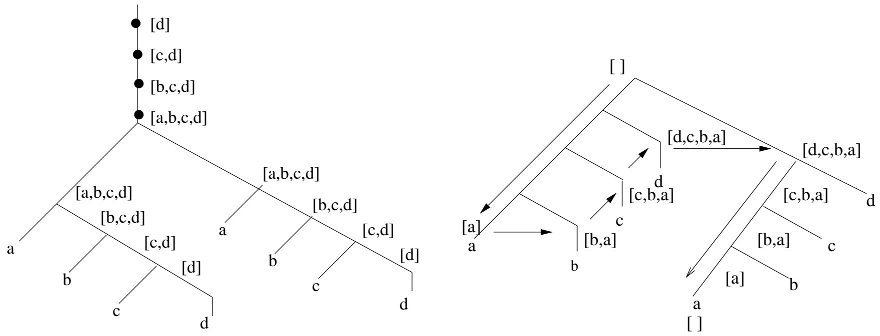

Figure 1.5: An IG structural description of the copy language Gazdar (1988) (left) and a GIG structural description (right)

GIGs cannot produce this structural description either, but they can generate the one presented in the figure 1.5 at the right, where the arrows depict the leftmost derivation order. GIGs can also produce similar structural descriptions for the language of multiple copies (the language $\{ww^{+}|\ w \in \Sigma^{*}\}$ as shown in figure 1.6, corresponding to a grammar like the one shown in example 2.

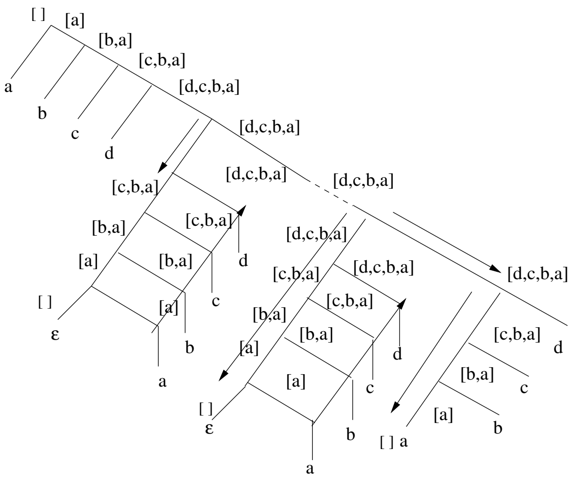

Figure 1.6: A GIG structural description for the multiple copy language

1.3.3 Multiple dependencies

There is no discussion of the applicability of multiple dependency structures in Gazdar (1988). The relevant structures that can be produced by a LIG are depicted in figures 1.7 and 1.8 (left). GIGs can generate the same structures as in 1.7 and the somewhat equivalent in 1.8 at the right.

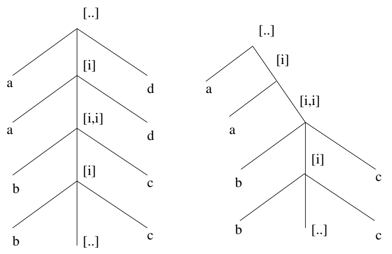

Figure 1.7: LIG and GIG structural descriptions of 3 and 4 dependencies

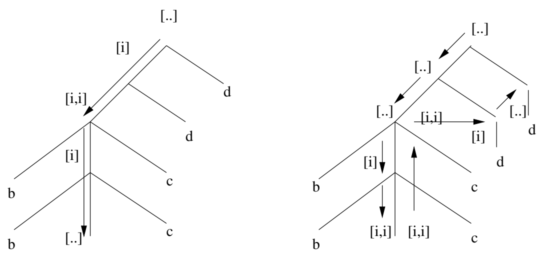

Figure 1.8: A LIG (left) and a GIG (right) structural descriptions of 3 dependencies

Also, GIGs can produce other structures that cannot be produced by a LIG, as we show in figure 1.9, including those corresponding to G_{sum} discussed above.

1.3.4 Conclusions

We have reviewed GIGs and GILs and their most important properties. We showed that the descriptive power of GIGs is beyond CFGs. CFLs are properly included in GILs by definition. We showed also that GIGs include some languages that are not in the LIL/TAL family nor in the MCSLs as characterized in Weir (1992). The similarity between GIGs and LIGs, strongly suggests that LILs might be included in GILs. We presented a comparison of the structural descriptions that LIGs and GIGs can be generate. We have shown that GIGs generate structural descriptions for the copy and multiple dependency languages which can not be generated by LIGs. Finally, we have shown also that the extra power that characterizes GIGs,

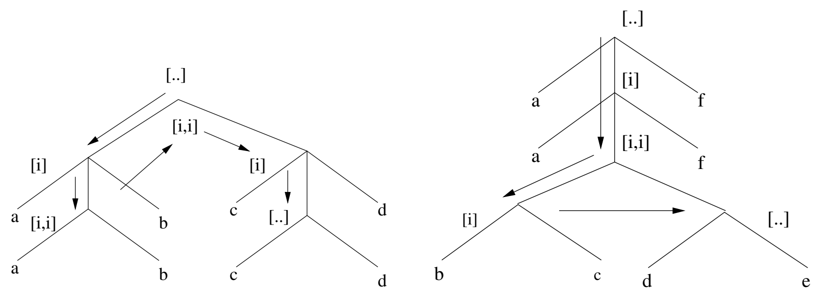

Figure 1.9: A GIG structural descriptions of 4 dependencies (left) and the example of grammar G_{sum} (right)

corresponds to the ability of GIGs to generate dependent paths without *copying* the stack but *distributing* the control in different paths.

Acknowledgments: Thanks to J. Pustejovsky for his continuous support and encouragement on this project. Many thanks also to the anonymous reviewers who provided many helpful comments. This work was partially supported by NLM Grant R01 LM06649-02.

Bibliography

Aho, A. V. (1968). Indexed grammars - an extension of context-free grammars. *Journal of the Association for Computing Machinery*, **15**(4):647–671.

Castaño, J. (2003). GIGs: Restricted context-sensitive descriptive power in bounded polynomial-time. In *Proc. of Cicling 2003, Mexico City, February 16-22*.

Castaño, J. (2003b). LR Parsing for Global Index Languages (GILs). In *In Proceeding of CIAA 2003, Santa Barbara,CA.*

Cherubini, A., L. Breveglieri, C. Citrini, and S. Reghizzi (1996). Multipushdown languages and grammars. *International Journal of Foundations of Computer Science*, **7(3)**:253–292.

Dassow, J. and G. Păun (1989). *Regulated Rewriting in Formal Language Theory*. Springer, Berlin, Heidelberg, New York.

Dassow, J., G. Păun, and A. Salomaa (1997). Grammars with controlled derivations. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages, Vol. 2*. Springer, Berlin,.

Gazdar, G. (1988). Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, eds., *Natural Language Parsing and Linguistic Theories*, pp. 69–94. D. Reidel, Dordrecht.

Harju, T., O. Ibarra, J. Karhumäki, and A. Salomaa (2001). Decision questions concerning semilinearity, morphisms and commutation of languages. In *LNCS 2076*, p. 579ff. Springer.

Joshi, A. (1985). Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural description? In D. Dowty, L. Karttunen, and A. Zwicky, eds., *Natural language processing: psycholinguistic, computational and theoretical perspectives*, pp. 206–250. Chicago University Press, New York.

Joshi, A. (2000). Relationship between strong and weak generative power of formal systems. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pp. 107–114. Paris, France.

Joshi, A., K. Vijay-Shanker, and D. Weir (1991). The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, eds., *Foundational issues in natural language processing*, pp. 31–81. MIT Press, Cambridge, MA.

Khabbaz, N. A. (1974). A geometric hierarchy of languages. *Journal of Computer and System Sciences*, **8**(2):142–157.

Miller, P. (1999). *Strong Generative Capacity*. CSLI Publications, Stanford University, Stanford CA, USA.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami (1991). On multiple context-free grammars. *Theoretical Computer. Science*, pp. 191–229.

Steedman, M. (1985). Dependency and coordination in the grammar of dutch and english. *Language*, pp. 523–568.

Vijay-Shanker, K., D. J. Weir, and A. K. Joshi (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of the 25th ACL*, pp. 104–111. Stanford, CA.

Weir, D. (1988). *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. thesis, University of Pennsylvania.

Weir, D. J. (1992). A geometric hierarchy beyond context-free languages. *Theoretical Computer Science*, **104**(2):235–261.