On the Applicability of Global Index Grammars

José M. Castaño
Computer Science Department
Brandeis University
jcastano@cs.brandeis.edu

Abstract

We investigate Global Index Grammars (GIGs), a grammar formalism that uses a stack of indices associated with productions and has restricted context-sensitive power. We discuss some of the structural descriptions that GIGs can generate compared with those generated by LIGs. We show also how GIGs can represent structural descriptions corresponding to HPSGs (Pollard and Sag, 1994) schemas.

1 Introduction

The notion of *Mildly 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 three properties:[1] a) *constant growth* property (or the stronger *semi-linearity* property); b) polynomial parsability; c) *limited cross-serial* dependencies, i.e. some limited context-sensitivity. The canonical NL problems which exceed context free power are: *multiple agreements*, *reduplication*, *crossing dependencies*.[2]

Mildly Context-sensitive Languages (MCSLs) have been characterized by a geometric hierarchy of grammar levels. A level-2 MCSL (eg. TALs/LILs) is able to capture up to 4 counting dependencies (includes $L_4 = \{a^n b^n c^n d^n | n \geq 1\}$ but not $L_5 = \{a^n b^n c^n d^n e^n | n \geq 1\}$). They were proven to have recognition algorithms with time complexity $\mathsf{O}(\mathsf{n}^6)$ (Satta, 1994). In general for a level-k MCSL the recognition problem is in $\mathsf{O}(\mathsf{n}^{3 \cdot 2^{k-1}})$ and the descriptive power regarding counting dependencies is 2^k (Weir, 1988). Even the descriptive power of level-2 MCSLs (Tree Adjoining Grammars (TAGs), Linear Indexed Grammars (LIGs), Combinatory Categorial Grammars (CCGs) might be considered insufficient for some NL problems, therefore there have been many proposals[3] to extend or modify them. On our view the possibility of modeling coordination phenomena is probably the most crucial in this respect.

In (Castaño, 2003) we introduced Global Index Grammars (GIGs) - and GILs the corresponding languages - as an alternative grammar formalism that has a restricted context sensitive power. We showed that GIGs have enough descriptive power to capture the three phenomena mentioned above (*reduplication, multiple agreements, crossed agreements*) in their generalized forms. Recognition of the language generated by a GIG is in **bounded** polynomial time: $O(n^6)$. We presented a Chomsky-Schützenberger representation theorem for GILs. In (Castaño, 2003c) we presented the equivalent automaton model: LR-2PDA and provided a characterization the-

[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]There are extensions or modifications of TAGs, CCGs, IGs, and many other proposals that would be impossible to mention here.

orems of GILs in terms of the LR-2PDA and GIGs. The family of GILs is an Abstract Family of Language.

The goal of this paper is to show the relevance of GIGs for NL modeling and processing. This should not be understood as claim to propose GIGs as a grammar model with "linguistic content" that competes with grammar models such as HPSG or LFG. It should be rather seen as a formal language resource which can be used to model and process NL phenomena beyond context free, or beyond the level-2 MCSLs (like those mentioned above) or to compile grammars created in other framework into GIGs. LIGs played a similar role to model the treatment of the SLASH feature in GPSGs and HPSGs, and to compile TAGs for parsing. GIGs offer additional descriptive power as compared to LIGs or TAGs 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]

This paper is organized as follows: section 2 reviews Global Index Grammars and their properties and we give examples of its weak descriptive power. Section 3 discusses the relevance of the strong descriptive power of GIGs. We discuss the structural description for the palindrome, copy and the multiple copies languages $\{ww^+ | w \in \Sigma^*\}$. Finally in section 4 we discuss how this descriptive power can be used to encode HPSGs schemata.

2 Global Index Grammars

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. The class of LILs contains L_4 but not L_5 (see above).

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

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

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

1.$S[..] \rightarrow aS[i..]$ *2.*$S[..] \rightarrow bS[j..]$
3.$S[..] \rightarrow cR[..]$ *4.*$R[i..] \rightarrow R[..]a$
5.$R[j..] \rightarrow R[..]b$ *6.* $R[] \rightarrow \epsilon$

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* mechanisms (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 rules that have terminals in the right-hand side. An additional constraint that is imposed on GIGs is strict leftmost derivation whenever indices are introduced or removed by 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 non-terminals 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,*[5] *where*

[5]The notation in the rules makes explicit that operation on the stack is associated to the production and neither to terminals nor to non-terminals. It also makes explicit that the operations are associated to the computation of a Dyck language (using such notation as used in e.g. (Harrison, 1978)). In another notation: a.1 $[y..]A \rightarrow [y..]\alpha$, a.2 $[y..]A \rightarrow [y..]\alpha$, b. $[..]A \rightarrow [x..]a\ \beta$ and c. $[x..]A \rightarrow [..]\alpha$

$x \in I$, $y \in \{I \cup \#\}$, $A \in N$, $\alpha, \beta \in (N \cup T)^*$ *and* $a \in T$.

$a.i\ A \underset{\epsilon}{\rightarrow} \alpha$ *(epsilon)*

$a.ii\ A \underset{[y]}{\rightarrow} \alpha$ *(epsilon with constraints)*

$b.\ A \underset{x}{\rightarrow} a\ \beta$ *(push)*

$c.\ A \underset{\bar{x}}{\rightarrow} \alpha\ a\ \beta$ *(pop)*

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:

   i. $\delta\#\beta A\gamma \underset{\mu}{\Rightarrow} \delta\#\beta X_1 ... X_n\gamma$

   ii. $x\delta\#\beta A\gamma \underset{\mu}{\Rightarrow} x\delta\#\beta X_1 ... X_n\gamma$

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 LIGs, indices are associated with only one non-terminal at right-hand side of the rule. This produces the effect that there is only one stack affected at each derivation step, with the consequence of the semilinearity property of LILs. GIGs share this *uniqueness* of the stack with LIGs: there is only one stack to be considered. Unlike LIGs and IGs the stack of indices is independent of non-terminals in the GIG case. GIGs can have rules where the right-hand side of the rule is composed only of terminals and affect the stack of indices. Indeed *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 rules (*push* and *pop* rules) that affect the stack of indices. The constraint imposed on the *push* productions can be seen as constraining the context sensitive dependencies to the introduction of lexical information. This constraint prevents GIGs from being equivalent to a Turing Machine as is shown in (Castaño, 2003c).

2.2.1 Examples

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 NL.

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: $S \rightarrow AS \mid BS \mid C$ $\quad C \rightarrow RC \mid L$

$R \underset{\bar{i}}{\rightarrow} RA \quad R \underset{\bar{j}}{\rightarrow} RB \quad R \underset{[\#]}{\rightarrow} \epsilon$

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

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, (Castaño, 2003c) therefore ILs and GILs could be 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:

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

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

$D \underset{l}{\rightarrow} aSb \mid bSa \qquad D \underset{m}{\rightarrow} aSc \mid cSa \qquad D \underset{n}{\rightarrow} bSc \mid cSb$
$L \underset{\bar{l}}{\rightarrow} c \qquad L \underset{\bar{m}}{\rightarrow} b \qquad L \underset{\bar{n}}{\rightarrow} a$

The following example shows that the family of GILs contains languages which do not belong to the MCSL family.

Example 4 (Multiple dependencies) $L(G_{gdp}) = \{ a^n(b^nc^n)^+ \mid n \geq 1\}$,
$G_{gdp} = (\{S, A, R, E, O, L\}, \{a, b, c\}, \{i\}, S, \#, P)$ *and* P *is:*

$S \rightarrow AR \qquad A \rightarrow aAE \qquad A \rightarrow a \qquad E \underset{i}{\rightarrow} b$
$R \underset{i}{\rightarrow} b\,L \qquad L \rightarrow OR \mid C \qquad C \underset{\bar{i}}{\rightarrow} c\,C \mid c$
$O \underset{\bar{i}}{\rightarrow} c\,OE \mid c$

The derivation of the string aabbccbbcc shows five dependencies.
$\#S \Rightarrow \#AR \Rightarrow \#aAER \Rightarrow \#aaER \Rightarrow i\#aabR \Rightarrow ii\#aabbL \Rightarrow ii\#aabbOR \Rightarrow i\#aabbcOER \Rightarrow \#aabbccER \Rightarrow i\#aabbccbR \Rightarrow ii\#aabbccbbL \Rightarrow ii\#aabbccbbC \Rightarrow i\#aabbccbbcC \Rightarrow \#aabbccbbcc$

2.3 GILs Recognition

The recognition algorithm for GILs we presented in (Castaño, 2003) is an *extension* of Earley's algorithm (cf. (Earley, 1970)) for CFLs. It has to be modified to perform the computations of the stack of indices in a GIG. In (Castaño, 2003) a graph-structured stack (Tomita, 1987) was used to efficiently represent ambiguous *index* operations in a GIG stack. Earley items are modified adding three parameters δ, c, o:

$[\delta, \mathbf{c}, \mathbf{o}, A \rightarrow \alpha \bullet A\beta, i, j]$

The first two represent a pointer to an active node in the graph-structured stack ($\delta \in I$ and $c \leq n$). The third parameter ($o \leq n$) is used to record the ordering of the rules affecting the stack.

The $O(n^6)$ time-complexity of this algorithm reported in (Castaño, 2003) can be easily verified. The *complete* operation is typically the costly one in an Earley type algorithm. It can be verified that there are at most n^6 instances of the indices (c_1, c_2, o, i, k, j) involved in this operation. The counter parameters c_1 and c_2, might be *state bound*, even for grammars with ambiguous indexing. In such cases the time complexity would be determined by the CFG backbone properties. The computation of the operations on the graph-structured stack of indices are performed at a constant time where the constant is determined by the size of the index vocabulary.

$O(n^6)$ is the worst case; $O(n^3)$ holds for grammars with *state-bound* indexing (which includes unambiguous indexing)[6]; $O(n^2)$ holds for unambiguous context free back-bone grammars with *state-bound* indexing and $O(n)$ for bounded-state[7] context free back-bone grammars with *state-bound* indexing.

3 GIGs and structural description

(Gazdar, 1988) introduces Linear Indexed Grammars and discusses its 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).

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 (we follow Gazdar's notation: the leftmost element within the brackets corresponds to the top of the stack):

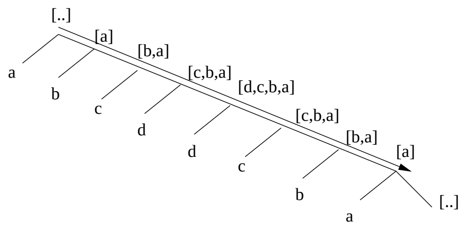

Figure 1: A non context-free structural description for the language ww^R (Gazdar, 1988)

Gazdar suggests that such configuration would be necessary to represent Scandinavian

[6]Unambiguous indexing should be understood as those grammars that produce for each string in the language a unique indexing derivation.

[7]Context Free grammars where the set of items in each state set is bounded by a constant.

unbounded dependencies.Such an structure can be obtained using a GIG (and of course a LIG).

But the mirror image of that structure cannot be generated by a GIG because it would require to allow *push* productions with a non terminal in the first position of the right-hand side. However the English adjective constructions that Gazdar argues that can motivate the LIG derivation, can be obtained with the following GIG productions as shown in figure 2.

Example 5 (Comparative Construction) .

$AP \rightarrow AP\ NP \qquad AP \rightarrow \bar{A} \qquad \bar{A} \rightarrow \bar{A}\ A$

$A \underset{i}{\rightarrow} a \qquad A \underset{j}{\rightarrow} b \qquad A \underset{k}{\rightarrow} c$

$NP \underset{\bar{i}}{\rightarrow} a\ NP \qquad NP \underset{\bar{j}}{\rightarrow} b\ NP \qquad NP \underset{\bar{k}}{\rightarrow} c\ NP$

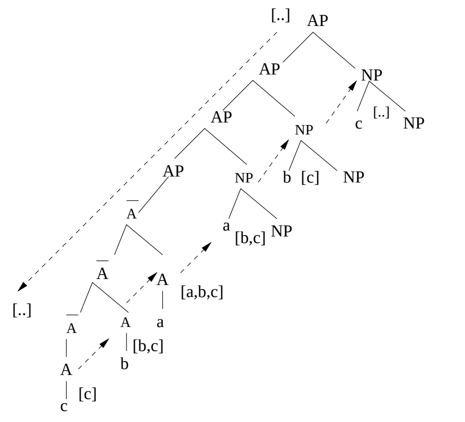

Figure 2: A GIG structural description for the language ww^R

It should be noted that the operations on indices follow the reverse order as in the LIG case. On the other hand, it can be noticed also 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 or TAG cases. The arrows show the leftmost derivation order that is required by the operations on the stack.

3.2 The Copy Language

Gazdar presents two possible LIG structural descriptions for the copy language. Similar structural descriptions can be obtained using GIGs. However he argues that another tree structure 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, and can by an IG (see (Castaño, 2003b) for a complete discussion and comparasion of GIG and LIG generated trees).

GIGs cannot produce this structural description, but they can generate the one presented in figure 3, 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 4, corresponding to the grammar shown in example 2.

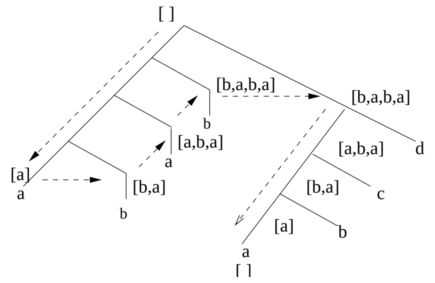

Figure 3: A GIG structural description for the copy language

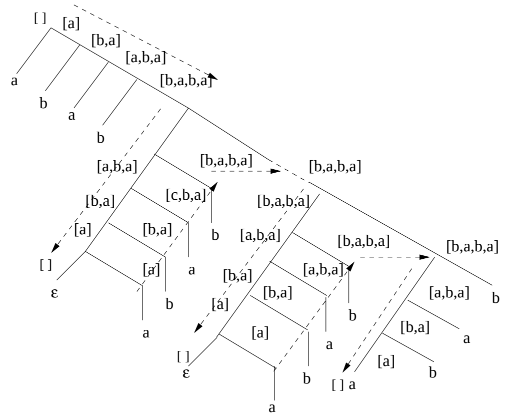

Figure 4: A GIG structural description for the multiple copy language

4 GIGs and HPSGs

We showed in the last section how GIGs can produce structural descriptions similar to those of LIGs, and others which are beyond LIGs and TAGs descriptive power. Those structural descriptions corresponding to figure 1 were correlated to the use of the SLASH feature in GPSGs and HPSGs. In this section we will show how the structural description power of GIGs, is not only able to capture those phenomena but also additional structural descriptions, compatible with those generated by HPSGs. This follows from the ability of GIGs to capture dependencies through different paths in the derivation.

There has been some work compiling HPSGs into TAGs (cf. (Kasper et al., 1995), (Becker and Lopez, 2000)). One of the motivations was the potential to improve the processing efficiency of HPSG, performing HPSG derivations at compile time. Such compilation process allowed to identify significant parts of HPSG grammars that were mildly context-sensitive.

We will introduce informally some slight modifications to the operations on the stacks performed by a GIG. We will allow the productions of a GIG to be annotated with finite strings in $I \cup \bar{I}$ instead of single symbols. This does not change the power of the formalism. It is a standard change in PDAs (cf. (Harrison, 1978)) to allow to push/pop several symbols from the stack. Also the symbols will be interpreted relative to the elements in the top of the stack (as a Dyck set). Therefore different derivations might be produced using the same production according to what are the topmost elements of the stack. This is exemplified with the productions $X \underset{\bar{n}v}{\rightarrow} x$ and $X \underset{[n]v}{\rightarrow} x$, in particular in the first three cases where different actions are taken (the actions are explained in the parenthesis) :

$nn\delta\#wX\beta \underset{\bar{n}v}{\Rightarrow} vn\delta\#wx\beta$ (pop n and push v)

$n\bar{v}\delta\#wX\beta \underset{\bar{n}v}{\Rightarrow} \delta\#wx\beta$ (pop n and $\bar{v}$)

$vn\delta\#wX\beta \underset{\bar{n}v}{\Rightarrow} v\bar{n}vn\delta\#wx\beta$ (push $\bar{n}$ and v)

$n\delta\#wX\beta \underset{[n]v}{\Rightarrow} vn\delta\#wx\beta$ (check and push)

We exemplify how GIGs can generate similar structural descriptions as HPSGs do, in a very oversimplified and abstract way. We will ignore many details and try give an rough idea on how the *transmission* of features can be carried out from the lexical items by the GIG stack, obtaining very similar structural descriptions.

Head-Subj-Schema

Figure 5 depicts the tree structure corresponding to the Head-Subject Schema in HPSG (Pollard and Sag, 1994).

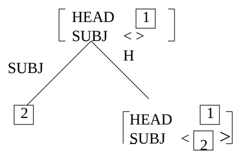

Figure 5: Head-Subject Schema

Figure 6 shows an equivalent structural description corresponding to the GIG productions and derivation shown in the next example (which might correspond to an intransitive verb). The arrows indicate how the transmission of features is encoded in the leftmost derivation order, an how the elements contained in the stack can be correlated to constituents or lexical items (terminal symbols) in a constituent recognition process.

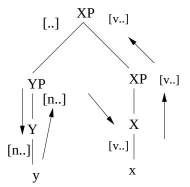

Figure 6: Head-Subject in GIG format

Example 6 (Intransitive verb) $XP \rightarrow YP\ XP$ $XP \rightarrow X$ $YP \rightarrow Y$ $X \underset{\bar{n}v}{\rightarrow} x$ $Y \underset{n}{\rightarrow} y$

$\#XP \Rightarrow \#YPXP \Rightarrow \#yXP \Rightarrow n\#YXP \Rightarrow n\#yX \Rightarrow v\#yx$

Head-Comps-Schema Figure 7 shows the tree structure corresponding to the Head-Complement schema in HPSG.

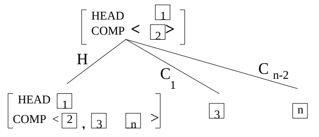

Figure 7: Head-Comps Schema tree representation

The following GIG productions generate the structural description corresponding to figure 8, where the initial configuration of the stack is assumed to be $[n]$:

Example 7 (transitive verb) .
$XP \rightarrow X\ CP$ $CP \rightarrow Y\ CP$ $X \underset{\bar{n}v\bar{n}}{\rightarrow} x$ $CP \rightarrow \epsilon$ $Y \underset{n}{\rightarrow} y$

The derivation:
$n\#XP \Rightarrow n\#XCP \Rightarrow \bar{n}v\#xCP \Rightarrow \bar{n}v\#xYCP \Rightarrow v\#xyCP \Rightarrow v\#xy$

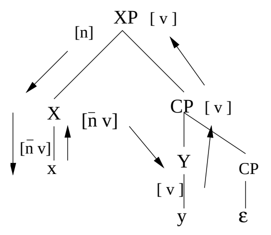

Figure 8: Head-Comp in GIG format

The productions of example 8 (which use some of the previous examples) generate the structural description represented in figure 9, corresponding to the derivation given in example 8. We show the contents of the stack when each lexical item is introduced in the derivation.

Example 8 (SLASH in GIG format) .

$XP \rightarrow YP\,XP \quad XP \rightarrow X\,CP \quad XP \rightarrow X\,XP$
$CP \rightarrow YP\,CP \quad X \underset{\bar{n}v\bar{n}}{\rightarrow} hates \quad CP \rightarrow \epsilon$
$X \underset{\bar{n}\bar{v}}{\rightarrow} know \quad X \underset{\bar{n}v\bar{v}}{\rightarrow} claims$
$YP \underset{n}{\rightarrow} Kim|Sandy|Dana|we$

A derivation of ‘Kim we know Sandy claims Dana hates’*:*
$\#XP \Rightarrow \#YP\,XP \Rightarrow n\#Kim\,XP \Rightarrow$
$n\#Kim\,YP\,XP \Rightarrow nn\#Kim\,we\,XP \Rightarrow$
$nn\#Kim\,we\,X\,XP \Rightarrow \bar{v}n\#Kim\,we\,know\,XP \Rightarrow$
$\bar{v}n\#Kim\,we\,know\,YP\,XP \Rightarrow$
$n\bar{v}n\#Kim\,we\,know\,Sandy\,XP \Rightarrow$
$n\bar{v}n\#Kim\,we\,know\,Sandy\,X\,XP \Rightarrow$
$\bar{v}n\#Kim\,we\,know\,Sandy\,claims\,XP \Rightarrow$
$\bar{v}n\#Kim\,we\,know\,Sandy\,claims\,YP\,XP \Rightarrow$
$n\bar{v}n\#Kim\,we\,know\,Sandy\,claims\,Dana\,XP \overset{*}{\Rightarrow}$
$\#Kim\,we\,know\,Sandy\,claims\,Dana\,hates$

Finally the last example and figure 10 show how coordination can be encoded.

Example 9 (SLASH and Coordination)
$XP \rightarrow YP\,XP \quad XP \rightarrow X\,CP \quad XP \rightarrow X\,XP$
$CP \rightarrow YP\,CP \quad CP \rightarrow \epsilon \quad X \underset{[n\bar{v}n]c}{\rightarrow} visit$
$X \underset{\bar{n}v\bar{n}}{\rightarrow} talk\ to \quad C \rightarrow and \quad CXP \rightarrow XP\,CXP$
$CXP \rightarrow C\,XP \quad X \underset{\bar{n}\bar{v}}{\rightarrow} did \quad YP \underset{n}{\rightarrow} Who|you$

5 Conclusions

We presented GIGs and GILs and showed the descriptive power of GIGs is beyond CFGs. CFLs are properly included in GILs by definition. We showed also that GIGs include

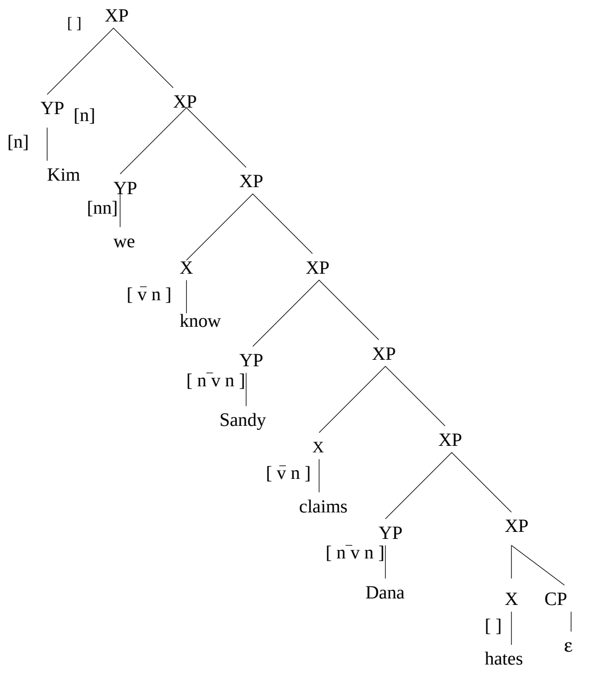

Figure 9: SLASH in GIG format

some languages that are not in the LIL/TAL family. GILs do include those languages that are beyond context free and might be required for NL modelling. The similarity between GIGs and LIGs, suggests that LILs might be included in GILs. We presented a succinct comparison of the structural descriptions that can be generated both by LIGs and GIGs, we have shown that GIGs generate structural descriptions for the copy language which can not be generated by LIGs. We showed also that this is the case for other languages that can be generated by both LIGs and GIGs. This corresponds to the ability of GIGs to generate dependent paths without *copying* the stack. We have shown also that those non-local relationships that are usually encoded in HPSGs as feature transmission, can be encoded in GIGs using its stack, exploiting the ability of Global stacks to encode dependencies through dependent paths and not only through a spine.

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

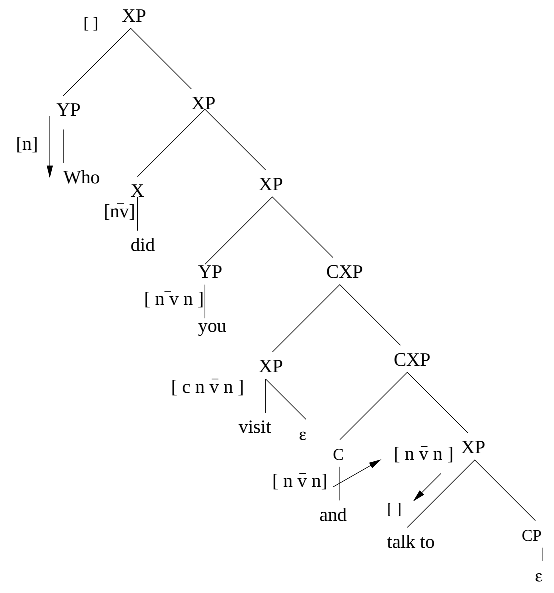

Figure 10: SLASH in GIG format

R01 LM06649-02.

References

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

T. Becker and P. Lopez. 2000. Adapting hpsg-to-tag compilation to wide-coverage grammars.

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

J. Castaño. 2003b. Global index grammars and descriptive power. In R. Oehrle and J. Rogers, editors, *Proc. of Mathematics of Language, MOL 8.* Bloomington, Indiana, June.

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

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

J. Earley. 1970. An Efficient Context-free Parsing Algorithm. *Communications of the ACM*, 13:94–102.

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

M. H. Harrison. 1978. *Introduction to Formal Language Theory.* Addison-Wesley Publishing Company, Inc., Reading, MA.

A. Joshi, K. Vijay-Shanker, and D. Weir. 1991. The convergence of mildly context-sensitive grammatical formalisms. In Peter Sells, Stuart Shieber, and Thomas Wasow, editors, *Foundational issues in natural language processing*, pages 31–81. MIT Press, Cambridge, MA.

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

A. Joshi. 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)*, pages 107–114, Paris, France.

R. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG into TAG. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 92–99. Cambridge, Mass.

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

C. Pollard and I. A. Sag. 1994. *Head-driven Phrase Structure Grammar.* University of Chicago Press, Chicago, IL.

G. Satta. 1994. Tree-adjoining grammar parsing and boolean matrix multiplication. *Computational linguistics*, 20, No. 2.

M. Tomita. 1987. An efficiente augmented-context-free parsing algorithm. *Computational linguistics*, 13:31–46.

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

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