Modeling Coordination by Means of Operations on Strings and on Derivation Trees

Carlos Martín-Vide[*]
Gheorghe Păun[†][‡]

Abstract

Several operations on strings are introduced, as models of the phenomenon of coordination in natural languages. Their relationships with other string operations is investigated. On this basis, the closure properties of families in the Chomsky hierarchy are obtained. In particular, we prove that the family of context-free languages is not closed under all but one of these operations. This special case concerns the coordination defined only between strings with a common syntactic structure (both strings have derivations described by identical trees, modulo the coordinated subwords). Some interpretations of these results are mentioned.

1 Coordination; Some Variants

The idea we start from is that in a given coordinated structure all the conjuncts are of the same type and status, and the coordination as a whole is of the same type and status as its subparts. This means that it is not possible to define only one head in the construction (coordination is not a projection of the conjunction, and there is no dependence between the two conjuncts), making it impossible to apply any general principle of hierarchical construction of the sentence (X-bar, for instance).

Coordination is basically a recursive phenomenon, because it builds phrase structures (trees associated to strings) of any length. Several studies of coordination start from the following generalization due to Chomsky, Chomsky (1957):

> If S_1 and S_2 are grammatical sentences, and S_1 differs from S_2 only in that X appears in S_1 where Y appears in S_2 (i.e., $S_1 = \ldots X \ldots$ and $S_2 = \ldots Y \ldots$), and X and Y are constituents of the same type in S_1 and S_2, respectively, then S_3 is a sentence, where S_3 is the result of replacing X by $X + and + Y$ in S_1 (i.e., $S_3 = \ldots X + and + Y \ldots$).

[*]Research Group on Mathematical Linguistics and Language Engineering (GRLMC)

[†]Institute of Mathematics of the Romanian Academy

[‡]Research supported by the Secretaría de Estado de Universidades e Investigación – SAB95-0357.

The usual treatment of coordination phenomenon starts from the idea that two categories can be catenated with a conjunction giving a larger category of the same type. The classical rule of this description obeys the context-free requirements, where X can be any linguistic category or nonterminal: $X \to X$ and X.

This schema can produce the well-known coordination cases between equal categories, which are common in all languages. Several problems arise if we include the treatment of the following cases of coordination:

1. Coordination of *unlike* categories. The general schema for this kind of sentences is similar to $Z \to Y$ and X, as appearing in the following examples Sag, Gazdar, Wasow, and Weisler (1985):
a. Pat is stupid and a liar (AP and NP).
b. Pat is a Republican and proud of it (NP and AP).
c. Pat is healthy and of sound mind (AP and PP).
d. That was a rude remark and in very bad taste (NP and PP),
where X and Y are different categories or nonterminals and Z is any category resulting from both X and Y. The problem in this schema is to explain how Z is constructed, because it is neither equal to X nor equal to Y (and therefore the rule is not recursive).

2. Binary coordination between many pairs of nonterminals. The following sentences are typical for a language of the form $\{a^n b^m c^n d^m \mid n, m \geq 1\}$, possessing crossed dependencies which cannot be produced by means of context-free rules without regulation:
a. John sent a letter and a postcard to Mary and to Paul, respectively.
b. The boys and the girls eat apples and bananas, respectively.
c. The boys and the girls run and walk through the garden, respectively.
d. $*$John and Mary sings and dances, respectively.

3. Non-constituent coordination and gapping phenomena. English and other languages contain a number of coordinate constructions where the conjuncts are not constituents in the normal sense but are sequences of constituents. The general term for such constructions is *non-constituent coordination*:
a. Mary studies art, John, music and Paul, history.
b. Harry has sent a letter to Mary, and John, a postcard to Paul.
c. Paul composed, and John posted, a letter to Mary.

Conjunctions can be performed in ordered pairs, where the order of the elements is fixed. The members of a couple may be the same (*o ... o* (Spanish), *et ... et* (French)) or different (*both ... and* (English)). One calls *binary coordination* the structure which has two conjuncts, and *multiple coordination* the structure with more than two. This may express a restriction over the set of conjunctions: *but* cannot appear in multiple coordination, and neither the couples with different words, as in the case of *both ... and*. All languages use this kind of structures, and this assumption suggests a unified treatment of the phenomenon. We try to do this in terms of formal operations on strings and languages.

2 Formal Language Prerequisites

As usual, V^* denotes the free monoid generated by the alphabet V with respect to the operation of concatenation; λ stands for the empty string, $V^+ = V^* - \{\lambda\}$, and $|x|$ is the length of $x \in V^*$. The set of all prefixes of $x \in V^*$ is denoted by $Pref(x)$. The *right derivative* of a language $L \subseteq V^*$ with respect to a string $x \in V^*$ is defined by: $\partial^r_x(L) = \{y \in V^* \mid yx \in L\}$. The *shuffle* of two strings $x, y \in V^*$ is defined by: $x ⧢ y = \{u_1v_1 \dots u_nv_n \mid n \geq 1, x = u_1 \dots u_n, y = v_1 \dots v_n, u_i, v_i \in V^*, 1 \leq i \leq n\}$. For $L_1, L_2 \subseteq V^*$ we write $L_1 ⧢ L_2 = \bigcup_{x \in L_1, y \in L_2}(x ⧢ y)$.

A context-free grammar is denoted by $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the set of productions, written as $A \rightarrow x$, $A \in N$, $x \in (N \cup T)^*$. The derivation relation is denoted by $\Longrightarrow$, its reflexive and transitive closure by $\Longrightarrow^*$; the language generated by G is denoted by $L(G)$.

The families of regular, linear, context-free, context-sensitive, and recursively enumerable languages are denoted by REG, LIN, CF, CS, and RE, respectively.

A *gsm* ("generalized sequential machine") is a system $g = (K, V_1, V_2, s_0, F, \delta)$, where K is the set of *states*, V_1 is the *input* alphabet, V_2 is the *output* alphabet, $s_0 \in K$ is the *initial state*, $F \subseteq K$ is the set of *final states*, and δ is a mapping (called *transition mapping*) from $K \times V_1$ to the set of finite subsets of $2^{V_2^* \times K}$. We extend δ to $K \times V_1^*$ as follows:

$$\delta(s, \lambda) = (\lambda, s),$$
$$\delta(s, ax) = \{(yx', s') \mid (x', s') \in \delta(s'', x), (y, s'') \in \delta(s, a)\},$$

for all $s \in K, a \in V_1, x \in V_1^*$. Then, for $w \in V_1^*, L \subseteq V_1^*$, we define

$$g(w) = \{z \in V_2^* \mid (z, s_f) \in \delta(s_0, w), \text{ for some } s_f \in F\},$$
$$g(L) = \bigcup_{w \in L} g(w).$$

A gsm is said to be λ-free if $\delta(s, a) \subseteq V_2^+ \times K$, for all $a \in V_1$, $s \in K$.

It is known that the families in the Chomsky hierarchy are closed under λ-free gsm mappings.

A morphism $h : V^* \rightarrow V^*$ is said to have *limited erasing on a language* $L \subseteq V^*$ (in short, we say that h is *limited on* L) if there is a constant k such that $|x| \leq k|h(x)|$ for all $x \in L, x \neq \lambda$.

All families in the Chomsky hierarchy are closed under limited morphisms.

For further notions and results in formal language theory we use here, we refer to Rozenberg and Salomaa (1997). We only recall here the important notion of a *derivation tree.*

Given a context-free grammar $G = (N, T, S, P)$, a tree τ with the nodes labelled by elements of $N \cup T \cup \{\lambda\}$, is a derivation tree with respect to G if:

1. the root of τ is labelled by S,
2. if the descendents of a node labelled by some $A \in N$ are $\alpha_1, \alpha_2, \dots, \alpha_k, k \geq 1, \alpha_i \in N \cup T$, then the production $A \rightarrow \alpha_1\alpha_2 \dots \alpha_k$ is in P,

3. if a node is labelled by λ, then it is the only descendent of a node labelled by some $A \in N$ and $A \to \lambda$ is a production in P,

4. the nodes labelled by elements of $T \cup \{\lambda\}$ have no descendents, all nodes labelled by elements of N have at least one descendent.

The nodes labelled by elements of $T \cup \{\lambda\}$ constitute the *frontier* of τ (the other nodes – excepting the root – are said to be internal nodes); we denote by $fr(\tau)$ the string in T^* identified by the frontier (we assume τ placed with the root above and we read $fr(\tau)$ from left to right, on the frontier nodes).

For a node ν in τ, we denote by $\tau(\nu)$ the subtree of τ with the root in ν, by $e(\nu)$ the label of ν, and by $fr(\tau(\nu))$ the subword of $fr(\tau)$ corresponding to the frontier of $\tau(\nu)$.

It is known that for every string w generated by a context-free grammar G there is a derivation tree τ with respect to G such that $fr(\tau) = w$; if G is unambiguous, then τ is unique.

3 Two Basic Coordination Operations on Strings

Intuitively, for two strings $x, y \in V^*$ with a common prefix, $x = ux'$, $y = uy'$, the coordination of x, y leads to a string $z = ux'y'$ (or $z = uy'x'$, if the order is not relevant). We consider here two variants of this operation, depending on whether the common prefix is maximal or not.

To start with, let us denote by $mp(x, y)$ the longest common prefix of x, y:

$$\begin{aligned} mp(x,y) = u \quad \text{iff} \quad & x = ux', y = uy' \text{ and there is no } u' \in V^* \\ & \text{such that } x = u'x'', y = u'y'' \text{ and } |u'| > |u|. \end{aligned}$$

Then, the *free prefix coordination* of x, y is defined by:

$$C_{fp}(x,y) = \{ux'y' \mid x = ux', y = uy', \text{ for some } u, x', y' \in V^*\},$$

whereas the *maximal prefix coordination* of x, y is defined by:

$$C_{mp}(x,y) = \{ux'y'\} \text{ iff } x = ux', y = uy' \; u = mp(x,y), x', y' \in V^*.$$

Observe that $C_{fp}(x,y)$ can contain several strings, that always $C_{mp}(x,y) \subseteq C_{fp}(x,y)$ and that $C_{mp}(x,y) \neq \emptyset$ (at least $\lambda \in Pref(x) \cap Pref(y)$; if $mp(x,y) = \lambda$, then $C_{mp}(x,y) = xy$).

Each operation $C_\alpha, \alpha \in \{fp, mp\}$, is extended in the natural way to languages:

$$C_\alpha(L) = \bigcup_{x,y \in L} C_\alpha(x,y).$$

In order to settle the closure properties of languages in the Chomsky hierarchy with respect to operations C_α, we use the following two auxiliary results, relating C_α to known operations on languages.

Lemma 1. *If FL is a family of languages closed under union, concatenation with symbols, intersection with regular languages, and right derivative, then FL closed under $C_\alpha, \alpha \in \{fp, mp\}$, implies FL closed under intersection.*

Proof. For a family FL as above, consider two languages $L_1, L_2 \in FL$, $L_1, L_2 \subseteq V^*$, and construct:

$$L = L_1\{cc_1\} \cup L_2\{cc_2\},$$

where c, c_1, c_2 are symbols not in V. For the regular language $R = V^*\{cc_1c_2\}$ we obtain:

$$L_1 \cap L_2 = \partial_{cc_1c_2}(C_\alpha(L) \cap R), \alpha \in \{fp, mp\}.$$

($\subseteq$) If $x \in L_1 \cap L_2$, then $x_1 = xcc_1 \in L_1, x_2 = xcc_2 \in L_2$. Clearly, $mp(x_1, x_2) = xcc_1c_2 \in C_\alpha(L) \cap R$. Moreover, $x = \partial^r_{cc_1c_2}(xcc_1c_2)$.

($\supseteq$) Take $x \in \partial^r_{cc_1c_2}(C_\alpha(L) \cap R)$. Then $xcc_1c_2 \in C_\alpha(L) \cap R$, hence there are $x_1 \in L_1\{cc_1\}, x_2 \in L_2\{cc_2\}$ such that $xcc_1c_2 \in C_\alpha(x_1, x_2)$. Because xcc_1c_2 contains one occurrence of c_1 and one of c_2, we must have $x_1 = x'_1cc_1, x'_1 \in L_1$, and $x_2 = x'_2cc_2, x'_2 \in L_2$. Because xcc_1c_2 contains only one occurrence of c, it follows that $mp(x_1, x_2) = x_3c$. Consequently, $x'_1 = x'_2 = x_3 = x$. This implies that $x \in L_1 \cap L_2$.

From the closure properties of FL, we obtain that $L \in FL$; if $C_\alpha(L) \in FL$, then also $L_1 \cap L_2 \in FL$. ♢

Lemma 2. *If FL is a family of languages closed under shuffle, λ-free gsm mappings, and limited morphisms, then FL is closed under $C_\alpha, \alpha \in \{fp, mp\}$.*

Proof. Let FL be a family as above and take $L \in FL, L \subseteq V^*$. Consider a new symbol, $c \notin V$. For each $a \in V$, take a new symbol, a', and denote $V' = \{a' \mid a \in V\}$. Consider the morphisms

$$h : V^* \longrightarrow V'^*, \text{ defined by } h(a) = a', \text{ for all } a \in V,$$
$$h' : (V \cup \{c\})^* \longrightarrow V^*, \text{ defined by } h'(a) = a, \text{ for } a \in V, \text{ and } h'(c) = \lambda.$$

Consider also the regular language

$$R = \{aa' \mid a \in V\}^*\{c^2\}V^*V'^*$$

and the gsm

$$g = (\{s_0, s_1, s_2\}, V \cup V' \cup \{c\}, V \cup \{c\}, s_0, \{s_2\}, \delta),$$

with the mapping δ defined by

$$\begin{array}{ll}
\delta(s_0, a) = \{(a, s_0)\},\ a \in V, & \delta(s_1, c) = \{(c, s_2)\}, \\
\delta(s_0, a') = \{(c, s_0)\},\ a \in V, & \delta(s_2, a) = \{(a, s_2)\},\ a \in V, \\
\delta(s_0, c) = \{(c, s_1)\}, & \delta(s_2, a') = \{(a, s_2)\},\ a \in V.
\end{array}$$

Then we obtain:

$$C_{fp}(L) = h'(g(((L \sqcup\!\sqcup \{c\}) \sqcup\!\sqcup (h(L) \sqcup\!\sqcup \{c\})) \cap R)). \qquad (*)$$

($\subseteq$) If $x \in C_{fp}(L)$, then $x = ux'y'$, for some $x_1 = ux', y_1 = uy'$, both in L. Then $ucx' \in L \sqcup\!\sqcup \{c\}, h(u)ch(y') \in h(L) \sqcup\!\sqcup \{c\}$. If $u = a_1a_2 \ldots a_k, k \geq 0, a_i \in V, 1 \leq i \leq k$, then $a_1a_1'a_2a_2' \ldots a_ka_k'ccx'h(y') \in ((L \sqcup\!\sqcup \{c\}) \sqcup\!\sqcup (h(L) \sqcup\!\sqcup \{c\})) \cap R$.

The gsm g works as follows:

– scanning a prefix $b_1d_1'b_2d_2' \ldots b_rd_r' \in (VV')^*$, one produces $b_1cb_2c \ldots b_rc$,

– when reading cc, these symbols are left unchanged,

– from now one, each $a \in V$ remains unchanged, and each $a' \in V'$ is replaced by a.

Therefore, $g(a_1a_1' \ldots a_ka_k'ccx'h(y')) = a_1c \ldots a_kcccx'y'$. Then, the morphism h' removes all occurrences of c, hence we get the string $a_1 \ldots a_kx'y' = x$.

($\supseteq$) Take a string x in the set in the right-hand side of the ($*$). There are $x_1 \in L$, $x_2 \in L, x_3 \in R$ such that x is obtained as in ($*$) from x_1, x_2, x_3. Denote:

$$\begin{aligned} x_1' &= u_1cx_1'' \in L \sqcup\!\sqcup \{c\}, \text{ for } x_1 = u_1x_1'', \\ x_2' &= h(u_2)ch(x_2'') \in h(L) \sqcup\!\sqcup \{c\}, \text{ for } x_2 = u_2x_2'', \\ x_3 &= a_1a_1' \ldots a_ka_k'ccx_3'x_3'', \text{ for } k \geq 0, x_3' \in V^*, x_3'' \in V'^*. \end{aligned}$$

We must have $a_1a_1' \ldots a_ka_k' \in u_1 \sqcup\!\sqcup h(u_2)$, hence $u_1 = u_2 = a_1 \ldots a_k$. Moreover, $x_1'' = x_3'$ and $h(x_2'') = x_3''$. Consequently, $x_1 = ux_1'', x_2 = ux_2''$, for $u = u_1 = u_2$, and $x = ux_1''x_2''$ (by the definition of g and h'). But $ux_1''x_2'' \in C_{fp}(x_1, x_2) \subseteq C_{fp}(L)$.

According to the closure properties of FL, we get $C_{fp}(L) \in FL$ (note that g is λ-free, h' is limited, and that the closure under gsm mappings – even λ-free – implies the closure under the intersection with regular languages).

For the case of C_{mp} we replace the regular language R by:

$$R' = \{aa' \mid a \in V\}^*\{c^2\}(\{bud'v \mid b, d \in V, u \in V^*, v \in V'^*, b \neq d\} \cup V^* \cup V'^*).$$

As above, we obtain:

$$C_{mp}(L) = h'(g(((L \sqcup\!\sqcup \{c\}) \sqcup\!\sqcup (h(L) \sqcup\!\sqcup \{c\})) \cap R').$$

The intersection with R' forces the selection of strings $a_1a_1' \ldots a_ka_k'ccx'h(y')$ as above, for $a_1 \ldots a_kx' \in L$, $a_1 \ldots a_ky' \in L$, with maximal k: either the next symbol (the first one in x' and in y') is different in the two strings, or one of x', y' is empty.

Therefore, $C_{mp}(L) \in FL$, too. $\diamondsuit$

Theorem 1. *The families REG, CS, RE are closed under C_α, LIN and CF are not closed, $\alpha \in \{fp, mp\}$.*

Proof. The families in the Chomksy hierarchy have the closure properties in Lemmas 1, 2, but LIN, CF are not closed under intersection. $\diamondsuit$

4 Some Variants of the Basic Operations

In the definition above, either any common prefix or only the maximal prefix of two strings is considered when coordinating the strings. This might not cover the case when only *certain* prefixes can be accepted. This is modeled by the *regulated prefix coordination* operation, defined as follows.

For a regular language $M \subseteq V^*$ and $x, y \in V^*$, we define:

$$C_{rp}(x,y) = \{ux'y' \mid x = ux', y = uy', \text{ for some } u, x', y' \in V^*, u \in M\}.$$

(Only prefixes belonging to M are taken into consideration.)

Another natural variant is to coordinate substrings of the two strings, identifying both prefixes and suffixes of them. Formally, the *free bilateral coordination* of $x, y \in V^*$ is defined by:

$$C_{fb}(x,y) = \{ux'y'v \mid x = ux'v, y = uy'v, \text{ for some } u, v, x', y' \in V^*\}.$$

Because it is not clear how the *maximal bilateral coordination* should be defined, we do not consider here this case. (For instance, consider $x = abbab$, $y = abbbab$. The maximal common prefix is abb, the maximal common suffix is $bbab$; they overlap, both in x and in y!)

We can, however, define in the usual way the *regulated bilateral coordination*, asking that both u, v in the definition above are elements of a given regular language; we denote by C_{rb} this operation.

Both C_{fb}, C_{rb} can be extended in the natural way to languages.

The next step is to iterate the operations $C_\alpha, \alpha \in \{fp, mp, rp, fb, rb\}$, defining, for $L \subseteq V^*$:

$$C^*_\alpha(L) = \bigcup_{i \geq 0} C^i_\alpha(L),$$

where

$$C^0_\alpha(L) = L, \; C^{i+1}_\alpha(L) = C_\alpha(C^i_\alpha(L)), \; i \geq 0.$$

It is easy to see that Lemma 1 holds true with the same proof for all operations C_α, C^*_α, α as above: with the notation in the proof of Lemma 1, we have:

$$C^*_\alpha(L) \cap R = C_\alpha(L) \cap R, \alpha \in \{fp, mp, fb\},$$

because the intersection with R selects the strings where only one occurrence of c is present. Moreover, for

$$R = M\{cc_1c_2\},$$

we also cover the case of $\alpha \in \{rp, rb\}$.

Theorem 2. *The families LIN, CF are closed under none of the operations C_α, C^*_α, $\alpha \in \{fp, mp, rp, fb, rb\}$.*

Let us examine now the proof of Lemma 2.

Instead of transforming a prefix $a_1a'_1 \ldots a_ka'_k$ of the scanned string into $a_1a_2 \ldots a_k$, the gsm g can also check whether or not $a_1a_2 \ldots a_k \in M$, for a given regular set. (Take a finite automaton for M and simulate it on the symbols $a_1, \ldots, a_k$; the details are left to the reader.) Therefore, Lemma 2 holds true also for C_{rp}.

For the bilateral case, we modify the proof of Lemma 2 as follows:

– consider three new symbols c_1, c_2, c,

– instead of R, consider the regular set:

$$R' = \{aa' \mid a \in V\}^*\{c_1^2\}V^*V'^*\{c_2^2\}\{aa' \mid a \in V\}^*,$$

– instead of g, consider the gsm:

$$g' = (\{s_0, s_1, s_2, s_3, s_4\}, V \cup \{c_1, c_2\}, V \cup \{c\}, s_0, \{s_4\}, \delta'),$$

with the transition mapping defined as suggested by Figure 1.

Then, for $L \subseteq V^*$ we obtain:

$$C_{fb}(L) = h'(g'(((L \sqcup\!\sqcup \{c_1c_2\}) \sqcup\!\sqcup (h(L) \sqcup\!\sqcup \{c_1c_2\})) \cap R')).$$

If the coordination is regulated by a language $M \in REG$, then, as in the case of C_{rp}, we can modify g' above in such a way to check whether or not the used prefix and suffix of the current strings are in M.

Consequently, Lemma 2 holds true also for C_α, $\alpha \in \{fb, rb\}$.

Theorem 3. *The families REG, CS, RE are closed under all operations C_α, $\alpha \in \{fp, mp, rp, fb, rb\}$.*

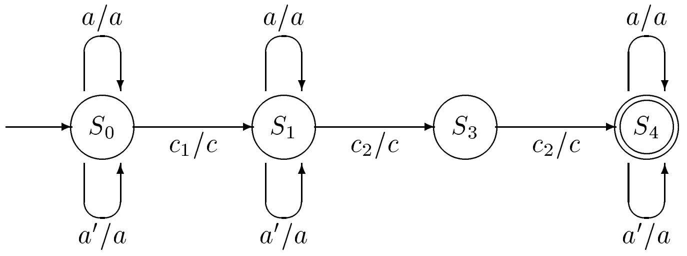

Figure 1

Theorem 4. *The families CS, RE are closed under all operations C^*_α, $\alpha \in \{fp, mp, rp, fb, rb\}$.*

Proof. For RE, the assertion is obvious (consequence of the Turing-Church Thesis). For CS, a straightforward (but long) construction can prove the assertion.

Here is the idea of such a construction for C^*_{fp}. The modifications for the other cases are obvious:

– Start from a context-sensitive grammar G, for a language $L \subseteq V^*$;
– Generate a string $x \in L(G)$;
– Generate one more string $y \in L(G)$;
– Find a common prefix of x, y; let it be u (hence $x = ux', y = uy'$);
– Having x, y, construct $ux'y'$;
– Consider $ux'y'$ in the role of x and go to step 2.

It is clear that the grammar G' obtained in this way generates exactly $C^*_{fp}(L)$. Moreover, G' has a bounded workspace: in order to generate a string w, it uses at most a space of length $2|w|$ (from $x = ux', y = uy'$ we get $w = ux'y'$ and $|xy| \leq 2|ux'y'|$). Consequently, $L(G') \in CS$. ♢

The closure of REG under the operations $C^*_\alpha, \alpha \in \{fp, mp, fb, rb\}$, remains *open.*

5 Syntactically Grounded Coordination

In the previous sections we have defined coordination operations on strings x, y without taking into account the syntactic structure of x, y. In natural languages, we pass from $x = ux', y = uy'$ (or $x = ux'v, y = uy'v$) to $w = ux'y'$ ($w = ux'y'v$, respectively) only when u (u, v, respectively) has (have) the same syntactic structure. This makes necessary to consider the derivation trees of x, y, hence to define the coordination for trees, not for strings.

Take a context-free grammar $G = (N, T, S, P)$.

Two derivation trees τ_1, τ_2 with respect to G are said to be *coordinable* if there is a node ν_1 in τ_1 and a node ν_2 in τ_2 such that if we remove $\tau_1(\nu_1)$ from τ_1 and $\tau_2(\nu_2)$ from τ_2 and we label both ν_1 and ν_2 with the same symbol, then we obtain two identical trees.

For two coordinable trees τ_1, τ_2 with respect to nodes ν_1, ν_2 labelled by X, Y, respectively, we construct a tree τ_3 as follows:

1. Excise $\tau_1(\nu_1)$ from τ_1.
2. Label ν_1 in the remaining tree by a new nonterminal symbol, Z.
3. Add at this node the subtree defined by the context-free rule $Z \rightarrow XY$.
4. Attach $\tau_1(\nu_1)$ to the new node labelled by X and $\tau_2(\nu_2)$ to the new node labelled by Y.

The obtained tree, τ_3, has a terminal frontier.

If we have $fr(\tau_1) = u_1x_1v_1$, $fr(\tau_2) = u_2x_2v_2$, for $x_1 = fr(\tau_1(\nu_1))$, $x_2 = fr(\tau_2(\nu_2))$, then, because τ_1, τ_2 are coordinable with respect to ν_1, ν_2, we have $u_1 = u_2$, $v_1 = v_2$. Then, $fr(\tau_3) = ux_1x_2v$, for $u = u_1 = u_2, v = v_1 = v_2$.

Therefore, $fr(\tau_3) \in C_{fb}(fr(\tau_1), fr(\tau_2))$.

We say that τ_3 has been obtained by coordination from τ_1, τ_2. For a grammar G, we denote by $C(G)$ the language consisting of all strings $fr(\tau)$, for τ being a tree obtained by coordination from two derivation trees with respect to G.

Note that we define $C(G)$ using exactly one coordination for each string in $C(G)$.

Figure 2 presents the idea of coordinable trees and of coordination.

The fact that, when coordinating (the frontier of) trees, the common parts of the frontiers are not only equal but they also have the same syntactic description has a rather powerful (and somewhat surprising) influence on the result: the operation preserves context-freeness.

Theorem 5. *For every context-free grammar G, the language $C(G)$ is context-free.*

Proof. If $G = (N, T, S, P)$, then we construct the grammar $G' = (N \cup \{Z\}, T \cup \{c\}, S, P')$, with

$$\begin{aligned} P' &= P \cup \{A \rightarrow xZy \mid A \rightarrow xXy \text{ and } A \rightarrow xYy \in P, \\ &\quad \text{for some } x, y \in (N \cup T)^*, X, Y \in N\} \\ &\cup \{Z \rightarrow XcY \mid X, Y \in N\}. \end{aligned}$$

Consider also the regular set:

$$R = T^*\{c\}T^*$$

and the morphism $h : (T \cup \{c\})^* \to T^*$ defined by $h(a) = a$, $a \in T$, and $h(c) = \lambda$. We obtain:

$$C(G) = h(L(G') \cap R).$$

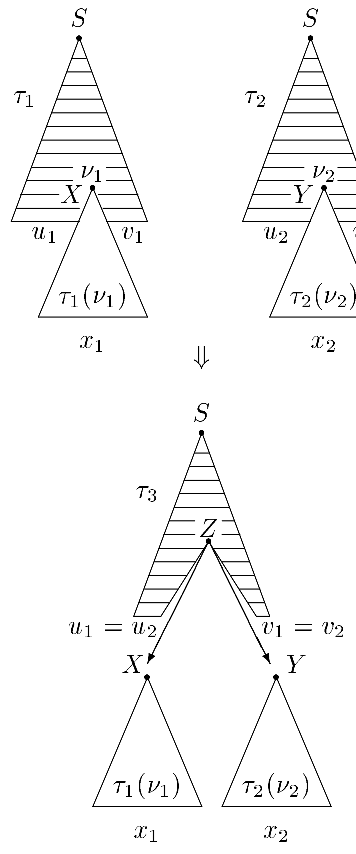

Figure 2

Indeed, because when we coordinate two trees τ_1, τ_2 of G with respect to two nodes ν_1, ν_2 the trees obtained by excising $\tau_1(\nu_1), \tau_2(\nu_2)$ from τ_1, τ_2, respectively, are identical modulo the labelling of ν_1, ν_2, there is a rule $A \to xXy$ used in τ_1 and a rule $A \to xYy$ used in τ_2 (possibly $X = Y$). Therefore, the new rules of P' perform a coordination operation. Removing c by the morphism h, we get the result of coordination, hence $C(G) \subseteq h(L(G') \cap R)$. Because the intersection with R selects from $L(G')$ exactly those strings in whose derivation we have used only once a rule $Z \to XcY$, we have also the converse inclusion.

From the closure properties of CF we obtain $C(G) \in CF$. ◊

Corollary. *If G is a regular grammar, then $C(G)$ is also regular.*

Proof. Exactly as above, starting from G regular, we construct G'. Because all recurrent derivations $A \Longrightarrow^* uAv$ in G' have $u \in T^*, v = \lambda$, according to Theorem 5.5 in Salomaa (1973), it follows that $C(G)$ is a regular language. $\diamondsuit$

For linear grammar, the above results are not true.

Theorem 6. *There is a linear grammar G such that $C(G) \notin LIN$.*

Proof. For the grammar:

$$G = (\{S\}, \{a, b\}, S, \{S \to aSb, S \to ab\}),$$

we clearly obtain:

$$C(G) = \{a^i a^n b^n a^m b^m b^i \mid i \geq 0, n, m \geq 1\},$$

which is not a linear language. $\diamondsuit$

In the coordination operation defined above, we allow not only to have both nodes ν_1, ν_2 identically labelled, but it is also possible to have $\tau_1(\nu_1) = \tau_2(\nu_2)$ (not to mention the weaker condition, $fr(\tau_1(\nu_1)) = fr(\tau_2(\nu_2))$). Coordinating τ_1 with τ_1 looks artificial. Hence, we say that the coordination of τ_1, τ_2 with respect to the nodes ν_1, ν_2 is *nontrivial* if $\tau_1(\nu_1) \neq \tau_2(\nu_2)$. We denote by $NC(G)$ the language consisting of all strings $fr(\tau)$, for τ being obtained by a nontrivial coordination of two derivation trees with respect to G.

The apparently small difference between usual tree coordination and nontrivial coordination turns out to have a surprising effect on the type of the language $NC(G)$.

Theorem 7. *There is a linear grammar G such that $NC(G) \notin CF$.*

Proof. Consider again the grammar G in the proof of Theorem 6. All derivations in G are of the form:

$$S \Longrightarrow aSb \Longrightarrow a^2Sb^2 \Longrightarrow \ldots \Longrightarrow a^nSb^n \Longrightarrow a^{n+1}b^{n+1},\ n \geq 0.$$

Thus, two subtrees of derivation trees with respect to G are different if they have different heights. Therefore, we have:

$$NC(G) = \{a^i a^n b^n a^m b^m b^i \mid i \geq 0, n, m \geq 1, n \neq m\}.$$

Let us assume that $NC(G) \in CF$. Consider a context-free grammar $G' = (N, \{a, b\}, S, P)$ such that $L(G') = NC(G)$. In order to generate the prefix a^i and the suffix b^i in strings $a^i a^n b^n a^m b^m b^i$ of $NC(G)$, we need a derivation $X \Longrightarrow^* a^j X b^j$, $j \geq 1, X \in N$. The rules used in such a derivation are not used when producing substrings $a^n b^n$ or $a^m b^m$, because from X we have to generate substrings of the form $a^k a^n b^n a^m b^m b^l$. If in a subderivation leading to a block $a^n b^n$ or $a^m b^m$ of a string $a^i a^n b^n a^m b^m b^i$, after generating some $a^s Y b^s$, we introduce X from Y, then strings not in $NC(G)$ will be obtained. Therefore, if we remove from P all rules contributing to a derivation $X \Longrightarrow^* a^j X b^j$ as above, then we obtain a grammar

G'' generating strings of the form $a^{i_1}a^nb^na^mb^mb^{i_2}$, for all $n, m \geq 1$, $n \neq m$, and with finitely many values for i_1, i_2. Because the pairs (i_1, i_2) are well specified and finitely many, we can replace by λ each occurrence of a and b in the rules of G'' which contribute to the prefix a^{i_1} and to the suffix b^{i_2}, respectively. In this way, we obtain a context-free grammar G''' such that $L(G''') = \{a^nb^na^mb^m \mid n, m \geq 1, n \neq m\}$.

However, $L = \{a^nb^na^mb^m \mid n, m \geq 1, n \neq m\}$ is not a context-free language. Assume the contrary. It follows that also $L' = \{a^nb^nca^mb^m \mid n, m \geq 1, n \neq m\}$ is context-free. Take a context-free grammar G_0 for L'. All recurrent derivations in G_0 must be of the form $X \Longrightarrow^* a^iXb^i, i \geq 0$. (Any other type of recurrent derivations produces strings not in $a^nb^na^mb^m$, even without imposing restrictions on the relation between n and m.) Replace by λ each occurrence of b in G_0. The obtained grammar, G_0', contain only recurrent derivations of the form $X \Longrightarrow^* a^iX, i \geq 0$. According to Theorem 5.5 in Salomaa (1973), the language $L(G_0')$ must be regular. However, $L(G_0') = h(L')$, for $h(a) = a, h(c) = c, h(b) = \lambda$. Hence, $L'(G_0') = \{a^nca^m \mid n, m \geq 1, n \neq m\}$. This is not a regular language, a contradiction which concludes the proof. $\diamondsuit$

Theorem 8. *There is a regular grammar G such that $NC(G) \notin REG$.*

Proof. Consider the grammar:

$$G = (\{S\}, \{a, b\}, S, \{S \to aS, S \to b\}).$$

We obtain:

$$NC(G) = \{a^ia^nba^mb \mid i \geq 0, n, m \geq 0, n \neq m\}.$$

As in the previous proof, if $NC(G)$ is regular, then a regular grammar generating $\{a^nba^mb \mid n, m \geq 0, n \neq m\}$ can be constructed, which is contradictory, because this language is not regular. It follows that $NC(G) \notin REG$, too. $\diamondsuit$

6 Central Coordination

The case of *The boys eat apples + The girls eat bananas → The boys and the girls eat apples and bananas (respectively)* suggests the following variants of coordination operations.

For $x, y \in V^*$, we define the *free central coordination* by:

$$C_{fc}(x, y) = \{x'y'ux''y'' \mid x = x'ux'', y = y'uy'', \text{ for some } u, x', x'', y', y'' \in V^*\},$$

whereas the *maximal central coordination* is defined by:

$$C_{mc}(x, y) = \{x'y'ux''y'' \mid x = x'ux'', y = y'uy'', \text{ for } u, x', x'', y', y'' \in V^*, \text{ and there is no proper superword of } u \text{ which is common to } x \text{ and } y\}.$$

The proof of Lemma 1 holds true also for C_α, C_α^*, for $\alpha \in \{fc, mc\}$, whereas the proof of Lemma 2 can be modified in order to cover the new operation (see also the remarks before Theorem 3). We get:

Theorem 9. *The families REG, CS, RE are closed under C_α, CS, RE are closed under C_α^*, too, but LIN, CF are closed under none of these operations, $\alpha \in \{fc, mc\}$.*

The central coordination can be naturally defined in the syntactically grounded variant, imposing that the substring u has the same syntactical description in both strings x, y.

We denote by $C_c(G)$ the language of the frontier strings of trees obtained by central coordination of derivation trees in the context-free grammar G. Somewhat surprisingly, for this case we do not have a result like Theorem 4 above.

Theorem 10. *There is a linear grammar G such that $C_c(G) \notin CF$.*

Proof. Consider the grammar:

$$G = (\{S, A, B, C\}, \{a, b, c, d, e\}, S, P),$$
$$P = \{S \to A, S \to B, A \to aAb, B \to cBd, A \to C, B \to C, C \to e\}.$$

All strings of the form $x = a^n eb^n$, $y = c^m ed^m$, $n, m \geq 0$, are in $L(G)$ and they have the (maximal) common subword e. Moreover, in any derivation tree there is the subtree determined by $C \to e$. Thus we have:

$$C_c(G) \cap \{a, c\}^* e \{b, d\}^* = \{a^n c^m e b^n d^m \mid n, m \geq 0\},$$

a language which is not context-free. ♢

One can easily see that for each regular grammar G we have $C_c(G) \in REG$: two derivation trees with respect to G can have in common only a subtree corresponding to a final part of the associated derivations, so the central coordination is, in fact, a "suffix coordination".

7 Concluding Remarks

The following table synthesizes the results concerning the closure properties of families in the Chomsky hierarchy under the (non-iterated) coordination operations considered above (the notations are those used before, U stands for "undefined")

	C_{fp}	C_{mp}	C_{rp}	C_{fb}	C_{rb}	C	NC	C_{fc}	C_{mc}	C_c
REG	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
LIN	No	No	No	No	No	No	No	No	No	No
CF	No	No	No	No	No	Yes	No	No	No	No
CS	Yes	Yes	Yes	Yes	Yes	U	U	Yes	Yes	U
RE	Yes	Yes	Yes	Yes	Yes	U	U	Yes	Yes	U

Several interesting conclusions can be drawn on the basis of the results in this table.

With only one exception, that of unrestricted syntactically grounded coordination, none of these operations preserves the context-free languages.

Coordination is a basic linguistic operation, probably present in all languages. We have considered here many variants, both at the surface level of strings and taking into consideration derivation trees, hence the *structure* of strings. Thus, we can claim, at least statistically, that we have captured the idea of coordination by our definitions. Consequently, we can infer: *natural languages contain non-context-free specific constructions.* Coordination is one of them.

This does not necessarily implies that natural languages, English for instance, are not context-free languages in the restricted sense, as sets of strings, it merely means that *natural languages contain constructions which cannot be handled by the context-free grammar formalism.*

The only case when context-free languages are preserved, that of unrestricted syntactically grounded coordination, can be considered as insufficiently adequate, as it allows trivial coordination. Significantly enough, when nontrivial coordination is considered, the context-freeness is again lost. Moreover, there are linear grammars leading to non-context-free languages by nontrivial syntactically grounded coordination. (The fact that the family of linear languages is closed under none of the considered operations is not a surprise, because this family is not closed under concatenation while coordination involves a sort of concatenation in its definition.)

If context-free grammars are not sufficient, then what else ? This is not an easy question. The family of context-sensitive languages is closed under all coordination operations (except those syntactically grounded, which make no sense in this case as we have no available derivation tree). This confirms the general belief that natural languages lie somewhere at the context-sensitive level of the Chomsky hierarchy, but this is a loose conclusion: context-sensitive grammars are “too powerful”. A standard candidate are the Tree Adjoining Grammars (TAG), of various forms (with constraints, dependencies, etc). How the corresponding families of languages behave with respect to the previous coordination operations (on strings or on trees) remains as a *research topic.* The answer is of a definite interest for the adequacy of TAG’s as models of the syntax of natural language.

References

Chomsky, N. (1957). *Syntactic structures.* Mouton, The Hague.

Rozenberg, G. and A. Salomaa (Eds.) (1997). *Handbook of formal languages.* Springer-Verlag.

Sag, I., G. Gazdar, T. Wasow, and S. Weisler (1985). Coordination and how to distinguish categories. *Natural Language and Linguistic Theory 3*, 117–171.

Salomaa, A. (1973). *Formal languages.* Academic Press, New York.