論　文

On generating the word order of coordinated structures

Thomas Michael Gross

要　旨

自然言語における並列関係を記述するのは極めて難しい。オスボーヌ(2003a,b) が提案した説によると，文の構成素は語順を無視する樹型図にまとまってから，語順を生成する樹型図に出力される。それらの樹型図とその生成プロセスに特に着目し，説明する。結果として，出力された樹型図は時として１つではなく，類似した構造の集合体となる。その集合体のなかに分析された構造が存在したら，その分析は正しかったことを意味する。

キーワード：依存文法，並列関係，樹型図

0. Preliminary remarks

Coordination poses some very difficult problems for syntactic analysis. In Gross (2002), I proposed a theory of coordination that is based on the concept of conjoining sentential elements and pruning superfluous elements. This was called the big-conjunct approach, and the major idea was that such a theory bases its working on the assumption that what is coordinated are always sentences. However, in doing so, I did not refute the possibility that the analysis of coordination could be based on a small-conjunct approach. I did mention though, that I could not think of any possibility.

During this year, however, Timothy Osborne (2003a,b) will propose in a forthcoming

dissertation and various papers a theory of coordination that has two features: first, it is based on Dependency Grammar, and second it proposes a small-adjunct approach. For linguistic laymen and even for linguists who are not immediately involved with syntactic phenomena such as coordination, it may perhaps be difficult to grasp what it would mean if such a theory would really work. As someone who has researched syntactic phenomena within the framework of Dependency Grammar, I have but to regard a theory such as the one Osborne proposes, as one of the major events in linguistics − given however that the theory works.

It is of course impossible to introduce Osborne's theory of coordination in detail, so I will stick to a general outline. However, I will focus on a specific segment of Osborne's theory, namely his representational devices, and how and whether they manage to depict the sentence structure. It should be recalled that determining the sentence structure or the syntactic structure of utterances involving coordination is what constitutes the major problem when addressing coordination. In other words, what a suitable theory of coordination should achieve is the production of acceptable syntactic structures − acceptable in strict terms of linguistic structuring and acceptable in terms of intuition. A theory that produces structures that are linguistically acceptable but counterintuitive is not very convincing. On the other hand, a theory that produces structures that are linguistically unacceptable but nonetheless intuitive, is difficult to imagine. My main focus in this paper will be on how Osborne's theory deals with the actual sentence structures that involve coordination.

1. Basic tenets of Dependency Grammar

Timothy Osborne (2003a) bases his theory on Dependency Grammar (henceforth: DG). In Gross (1999) and various papers, I have outlined how a DG is modeled. Some mainstays of DG should be highlighted: 1. In a DG words are connected to other words by a hierarchical, i.e. anti-symmetric, relationship called dependency. Phrases are not subject of the immediate syntactic mechanisms of a DG, although they can play major roles in more removed areas. 2. A DG does not allow empty nodes. Dependency connections between two words may not involve one or more elements that are not present on the surface structure. This condition is difficult to paraphrase in DG terms because there is no distinction between a surface structure and other structural levels. It follows however, immediately from the next major condition for DGs. 3. A DG is monostratal. That means that DGs refute any concept of deep structure and as a corollary also any concept of mechanisms that convert one structure type into another structure type. In other words, DGs rule out transformational processes that convert a deep

structure into a surface structure. It does not matter much whether a non-surface structure is called deep structure or base or any other term. As empty nodes are generated by assumed interactions between different levels of a polystratal syntax empty nodes cannot exist in a DG. There is, however, considerable leeway in other areas. This includes in particular dependency determination procedures and dependency representations. As for the first issue, DGians are in considerable disagreement over what constitutes a proper dependency determination procedure. Such a procedure is required in order to first determine which elements depend on other elements. Conventionally three different types of procedure are distinguished, although the borders are − more often than not − blurred: 1. morphological dependency, 2. syntactic dependency, and 3. semantic dependency. Morphological dependencies are thought of as cases where only the form of one word is determined by another word. An example would be case of nouns as required by prepositions such as for instance German *wegen des Wetters* (*due to the weather*) where the preposition *wegen* always demands genitive case. However, even if *wegen* would require a different case, the relation between the preposition and the noun phrase would not change: it would still indicate a reason. A syntactic dependency can be thought of as for instance the dependency of a direct object on a verb. In German, for example, direct objects take accusative case, but accusative case is formally distinct from other cases only in masculine nouns, while feminine, neuter, and plural nouns have identical nominative and accusative forms. If a non-masculine noun occurs as a direct object, it can only be inferred that it takes accusative case, but it cannot be confirmed by looking at the actual form. In this respect the concept of syntactic dependency overrides that of morphological dependency, since the latter could not indicate accusative case. Semantic dependency is the concept of which most DGians have dissimilar notions. It is often used to indicate relationships between words that are not connected by a morphological or a syntactic dependency, or where the direction of morphological and syntactic dependencies runs opposite to its semantics. An example for the latter instance may be German *gebackene Kuchen* (*baked cakes*) where the past participle *gebackene* is morphologically and syntactically dependent on the noun *Kuchen*. However, at the same time, *Kuchen* is semantically the direct object of the participle, since past participles of transitive verbs − which *backen* is − always indicate passive voice.

The other area of disagreement is how dependency relations should be represented. In DG, the most used devices are dependency trees or stemmata (I will henceforth prefer the latter term). One notable exception is Richard Hudson who uses curved arrows to indicate

dependencies. In Gross (1992; 1999), I have detailed the various problems that different approaches to the generation of stemmata incur. Without going into details, it is evident that it matters a great detail if stemmata are assumed to work as dependency determination devices or just as representations of previously determined dependencies. In Gross (1992), I have named as *intuitionists* those DGians who assume that stemmata have validity prior to dependency determination procedures. Thus, intuitionists argue about whether a given stemma T is the correct structure of a sentence S. Furthermore, intuitionists do not explicate how stemmata are being generated. Unfortunately, also Tesniére, the father of modern DG, must be subsumed under the intuitionist label, in spite of his saying that language is unidirectional and one-dimensional (1959: 33, chap. 5, sentence 11). Stemmata are two-dimensional devices, and whoever desires to use these devices while analyzing a medium that is inherently characterized by one-dimensionality must be able to show on request how he arrives at a two-dimensional device from observations of a one-dimensional phenomenon. At various occasions I have argued that stemmata are concepts *a posteriori*, i.e. the analysis of one-dimensional phenomena must come before the application of two-dimensional representation devices. This includes terminology: terminology for stemmata generation may not be used for dependency determination. Because this issue will be at the heart of the topic addressed later, I will quickly show how stemmata can be arrived at: *gebackene Kuchen* consists of two words. The participle *gebackene* is originally a derivative adjective since all participles are adjectives. It requires a flexeme − here *+e* − which must match features inherent in the noun. Therefore, the noun *Kuchen* to some extenlt determines the form of the participle. This is an anti-symmetric relation since gebackene does not determine the form of the noun *Kuchen* in any way. All dependency relations are anti-symmetric relations. Thus, it is safe to assume that *gebackene* depends on *Kuchen*. The above-made claims constitute the dependency determination procedure. It is possible to utilize a certain formalism, but is nevertheless sufficient to just spell it out as I did above. The string *gebackene Kuchen* contains a further property: linear order. The participle occurs before the noun. This is an instance of the first dimension - which is the only dimension that can be directly observed. The order in the first dimension will be represented by *linear indices*. Words prior to other words receive indices with a value less than the later words. Since *gebackene* is the first word in the string, it receives the linear index {1}. The noun *Kuchen* receives the linear index {2}, since it is the second word. This may be written as in (1).

(1) gebackene$_1$ Kuchen$_2$

From the dependency determination procedure it is known that the participle depends on the noun. This information is neither directly accessible in the string nor in (1). However, it is possible to assign to the depending word the value of the linear index of the word on which the former depends. Since *gebackene* depends on *Kuchen* and since the linear index of *Kuchen* is {2}, this value is assigned to *gebackene* as a *dependency index*. The noun is not dependent on any other element in this string, so it will be assigned value {x}. Finite elements, i.e. those elements which are not dependent on any other element in a sentence, will be assigned value {0}. Thus, we gain structure (2).

(2) $_2$gebackene$_1$ $_X$Kuchen$_2$

In a series of definitions which cannot be detailed here but which are explicated in Gross (1992; 1999), dimensional relators, vectors, planes, and connecting branches are introduced, and as a result the stemma of (2) should look like the one in figure 1.

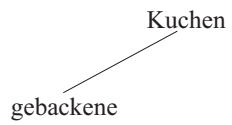


Fig. 1: Stemma of (2)

Note in particular that the stemma in figure 1 not only represents the dependency relation between the words, but also the linear sequence of the words. Because it has the latter property, stemmata such as the one in figure 1 are also called *projective* stemmata, since they correctly project the word order back into the first dimension.

2. Brief outline of Osborne's theory of coordination

Timothy Osborne's theory of coordination is based on several crucial assumptions (2003a,b): 1. Phenomena inherently pertaining to coordination must be distinguished from those that do not. Distinguished from pure coordination must be phenomena such as VP-ellipsis, N-ellipsis, pseudo-gapping, sluicing and others. 2. Coordination as such is not one mechanism but consists of two distinct phenomena. They are called *string coordination* and *gapping*. 3.

String coordination is to be distinguished into *forward string coordination* (FSC) and *backward string coordination* (BSC). 4. Osborne's theory employs two types of stemmata, and one type depicts − or claims to depict − only the second dimension (dependency), while the other type depicts not only the first dimension (linear word order), but also the second, and a third dimension. Osborne calls the first dimension *precedence*, the second *dominance*, and the third *behindance*. Although conventionally, stemmata are regarded as output representations, i.e. they are static and depict the result of a linguistic behavior, sometimes input stemmata are used. Input stemmata − or *pre-linearization* trees as Osborne calls them − depict only dependency relations but not linear word order. Linearization is achieved at the output stage.

Osborne makes a strong case for the distinction of different phenomena: those that are coordination and those that are not. His case is equally strong for the distinction of two different types of coordination: string coordination and gapping. Coordinated elements in string coordination are called conjoined, and those in gapping structures are called matched. Here are some examples:

(3)　old [men] and [women]
(4)　[men] and [women] from Mars
(5)　He orders wine, and she beer.

(3) is FSC, (4) is BSC, and (5) is an instance of gapping. The bracketed elements in (3) and (4) are the coordinated elements. In (3), an element occurs before the coordination which can refer to both conjoined elements, i.e. the men are old, and the women are old. This is called the *shared material*. In FSC such as (3), shared material is located in front of the coordination. In BSC such as (4) the shared material is located after the coordination; here it is *from Mars*.

In (5), the situation is more complicated: conventionally it is assumed that a verb − here it should be *orders* − has gapped between *she* and *beer*. Therefore, also conventionally, it is further assumed that the coordinated elements in (5) are sentences, namely *he orders wine* and *she orders beer*. It will turn out that Osborne refutes this assumption.

In order to understand Osborne's concept of coordination, it is first necessary to understand Osborne's use of the term *string*. Osborne distinguishes between three different string types: x-strings, y-strings, and z-strings. These terms refer to the dimensions of a post-serialization

tree. In such a tree, x-strings appear in the first dimension (precedence), while y-strings appear in the second dimension (dominance), and z-strings in the third dimension (behindance). All the necessary dependency terminology Osborne introduces is based on stemmatological concepts, i.e. stemmata are considered as an *a priori* to basic dependency terms. It is not required to introduce every dependency term Osborne defines, but in order to understand the concept of string, at least two terms must be introduced: *branch* and *stem*. A branch is the sum of all words and connections between them besides the finite word. It may, however, also be subset thereof. Observe the example in figure 2.

Fig. 2: Branch

In figure 2, the noun *cakes* acts as the finite word since it is not dependent on any other word in the structure. The words *very* and *tasty* as well as the connections between *very* and *tasty*, and between *tasty* and *cakes* can be regarded as one branch. Since the branch definition can apply to *any* word, the word *very* and its connection *to tasty* can also be considered as a branch. However, *tasty* and its connections − either upward or downward − cannot be considered as a branch, because *very* depends on *tasty*. I.e. the definition applies to any word W *and to all* words downward from W. A *stem* is a word W that dominates all words that can be arrived at by tracing from W to any other word along the connections. Thus, in the branch *very* − *tasty* − *cakes* the word *cakes* is the stem of *very* and *tasty*. According to Gross (1999), the words *very* and *tasty* are general successors of *cakes*.

Now it is possible to introduce the concept of strings. X-strings are sequences of words arranged in the first dimension, where only precedence matters. For example, in sentence (5), *wine* and *she* is a possible x-string. The only relevant property here is that wine appears before *and*, and *and* before *she*. However, no dependency relations have to hold.

A y-string is a sequence of words with a single stem arranged in the second dimension, where only dependency matters. For example, the stemma in figure 2 contains three y-strings: *very tasty*, *tasty cakes*, and *very tasty cakes*.

A z-string is a set of words arranged in the third dimension. The third dimension, behindance, is defined via the term *behind*. A sequence B is behind sequence A if 1. A precedes B, 2. A does not depend on B, and B does not depend on A, and 3. A and B form one branch with respect to the word they depend on. For all practical purposes, this relation only occurs in coordination. In sentences (3) and (4), *women* is behind *men*, and in (5), *she* is behind *he,* and *beer* is behind *wine*. Thus, in (3) and (4) there is one z-string respectively, namely *men* – *women*.

As soon as words occur in a behindance relation, post-serialization trees require *planes*. In Gross (1999), they are called *layers*. All words that are not behind other words occur on plane P^1. Words that are directly behind words on P^1, are located on P^2 asf.

Actual coordination is then accomplished by three axioms. For SC the Axiom of String Coordination demands that conjoined y-strings that qualify as x-strings are subject to coordination. Conjunction is posited by the Axiom of Conjunction: y-strings fulfilling the same syntactic function are conjoined. For gapping the Axiom of Matching demands that single y-strings are matched if they fulfill the same syntactic function.

3. Converting trees

After the – very – brief outline of Osborne's theory of coordination, it is now possible to turn to the question how Osborne's theory represents the sentences (3-5) as stemmata. First, the theory produces input stemmata (Osborne's pre-linearization trees). They are shown in figure 3.

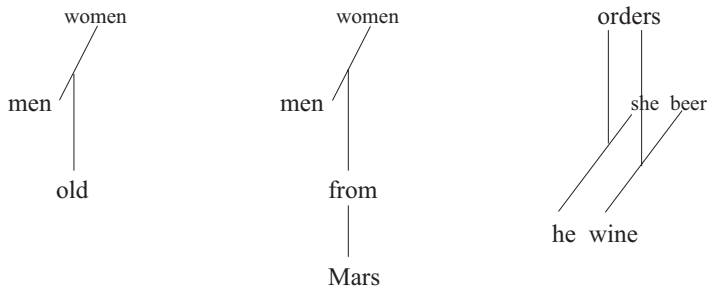
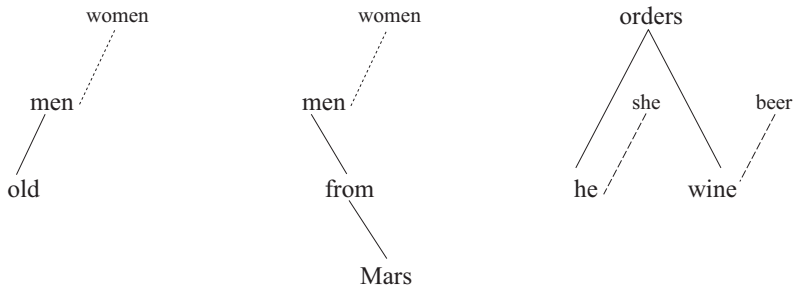


Fig. 3: Pre-linearization trees of (3-5) as (a-c)

A closer look at these pre-linearization trees reveals that in (a) and (b) the shared material depends on the − not yet − conjoined elements, while in (c) the − not yet − matched elements depend on the shared element. These trees are then converted into the following post-linearization trees.



(d) (e) (f)

Fig. 4: Post-linearization trees of (3-5) as (d-f)

The question that needs to be addressed in this section is how exactly the conversion from pre-linearization trees to post-linearization trees works. In order to understand that process, it is first necessary to look at the pre-linearization trees. Since they are pre-linear, they reveal no information on how the elements contained in these trees are ordered word by word. Osborne posits that in order to convert for instance tree (a) in figure 3 into tree (d) in figure 4 a linearization procedure receives (a) as input, assigns positions for the words contained in (a) and as a result outputs (d). Osborne (2003a) calls this procedure L and describes it as a "mechanism of the grammar that positions and spells out the nodes of D(ependency)-hierarchies". One primary principle of L is the Completeness Principle (CP): the plane P^1 must be a complete D(ependency)-hierarchy. Other planes however, need not be complete hierarchies. This means that if words located on a plane other than P^1 are omitted, the remaining words on P^1 must constitute a complete − and supposedly acceptable − dependency structure. However, if words on P^1 are omitted the remaining words on other planes are not bound by the CP.

Since there are two distinct phenomena of coordination, namely conjunction and matching, Osborne proposes two different linearization rules. The linearization rule for SC states 1. that L views all words positioned between conjoined words as coordinated material, and 2. that

for each departure from P[1] at least one coordinator must occur. It is necessary here to look at an example:

(6)　　Tom bought and fixed an old car.

Sentence (6) is an instance of both FSC and BSC. *Tom* is shared material in forward position, *an old car* shared material in backward position. A pre-linearization tree should look like the one in figure 5.

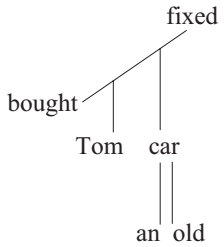


Fig. 5: Pre-linearization tree of sentence (6)

Condition 1 of the linearization rule for SC states that any word between conjoined words cannot be shared material but must be part of one conjunct. Figure 5 shows that candidates for conjunction are the verbs *bought* and *fixed*. If L should produce a sequence where either the branch *Tom* or the branch *an old car* are positioned between the conjoined words, then these branches cannot be shared material. This means that sentence (6) is one－but only one correct output for the tree in figure 5. Sentence (7), however, would violate condition 1:

(7)　　*Tom bought an old car and fixed.

Thus, L arrives at the post-linearization tree in figure 6.

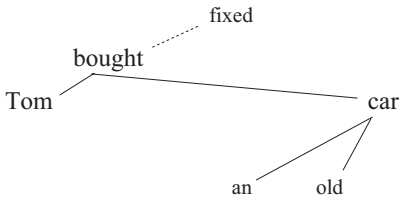


Fig. 6: Post-linearization tree of sentence (6)

The coordinator *and* is inserted directly after L departs from P^1 which is directly after the verb bought.

The linearization rule for gapping states 1. that L may jump between planes only immediately after the final node of a matched y-string or immediately after the final node of P^1, and 2. that for each departure from P^1 at least one coordinator must appear.

Gapping, however, must also take care to observe the Gapping Principle: L may view only those y-strings as matched that are dominated by a non-matched +Pred node. A +Pred node is any word having predicate potential such as e.g. infinitives, predicate AdjPs, predicate NPs, predicate PPs asf. Here too, an example is required:

(8) Bill forced you, and Larry me to read a book on syntax.

The linearization rule for gapping states in condition 1 that L may jump between planes only immediately after the final node of a matched y-string or immediately after the final node of P1. Figure 7 shows a pre-linearization tree for sentence (8).

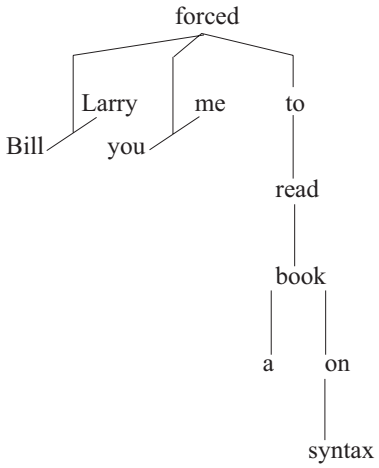


Fig. 7: Pre-linearization tree of sentence (8)

Subject to condition 1 are final nodes of matched y-strings and of plane P^1. As the tree in figure 7 shows, matched y-strings are *Bill − Larry* and *you − me*. Since in (8), L departs from P^1 after *you* which is the final node of a matched y-string, (8) is a − possible − output for the tree in figure 8. However, the next sentence would also be a valid output candidate for the

same tree.

(9)　　Bill forced you to read a book on syntax and Larry me.

In (9), L departs from P^1 after the final node of P^1, namely *syntax*. The post-linearization tree of sentence (8) should then look like that in figure 8, the post-linearization tree of sentence (9) should look like that in figure 9.

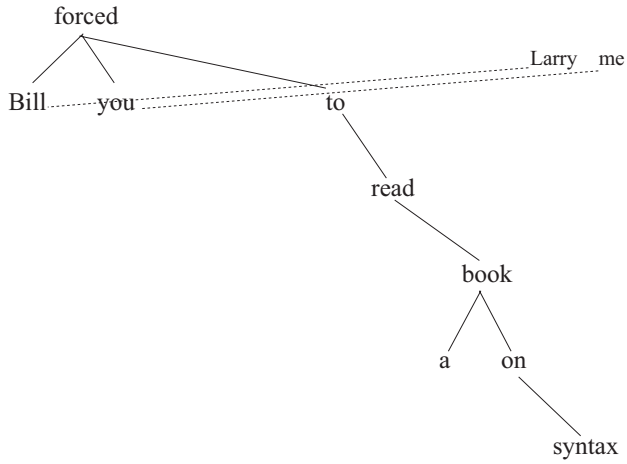


Fig. 8: Post-linearization tree of sentence (8)

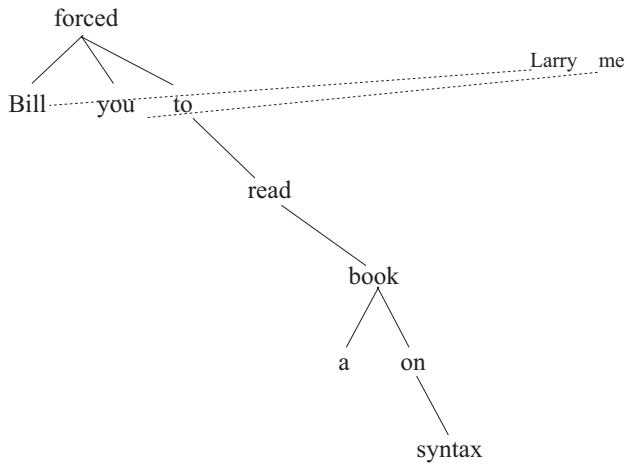


Fig. 9: Post-linearization tree of sentence (9)

4. A closer look at pre-linearization trees and the format of L

The brief outline of how pre-linearization trees are converted into post-linearization trees using the concept of a linearization mechanism L that operates according to the posited linearization rules seems to handle the data and to be intuitive at the same time.

However, it is necessary to reconsider in particular the pre-linearization trees and ask what exactly they formalize. In section 1 I briefly outlined how a theory of stemmata would need to be designed in order to generate stemmata by first making observations of linguistic phenomena. This means in particular that linguists find linguistic utterances as traces of linguistic behavior and describe their structures. In the utterances linguists commonly find *in natura* one property always applies: the utterances have a linear order. Consequently, linear order is a property that can be employed when trying to generate a stemma. Since linear order alone is insufficient, the linguist must add information on dependency. In the theory I outlined above, this is done by assigning dependency indices to the words after conducting a dependency determination procedure.

In contrast, Osborne's pre-linearization trees contain no information on linear order, but instead they contain information on dependency. It is difficult to imagine how observation of natural language data had to be conducted in order to only get information on dependency but not on linear order, but for the sake of the argument I shall accept this as a given. It is, however, necessary to ask what exactly an pre-linearization tree contains as information.

Observing tree (a) in figure 3, it is possible to ask what it actually depicts. It seems to claim nothing more than

(10.1) old_i / men

(10.2) old_i / women

where the slash symbol denotes dependency. Thus (10.1) can be read as "*old* depends on *men*", and (10.2) as "*old* depends on *women*". The index denotes identity, i.e. one and the same word *old* depends at the same time on *men* and *women*. If it was not the same word, (10) would derive sentence (11):

(11) old men and old women

The first problem (10) incurs is that it can be read as a double dependency. Although, at

least as far as I see it, nothing prevents double dependencies from occurring, many linguists find them undesirable and usually rule them out but posing respective constraints. Mel'čuk (1979: 12; 1988: 23), for example, defines as three properties of a stemma that 1. every element must be connected, 2. an element may be depend only on one element, and 3. one element may not be dependent on another element. Condition 1 is evident, and condition 3 implies that there must be a root − a finite element. Condition 2 constitutes a problem for a dependency determination procedure that produces statements such as those made in (10). Clearly it is violated since *old* depends on *men* and on *women*. While the question remains whether these conditions are at all necessary, because one could ask why formal prescriptions should rule out a certain segment of natural language utterances, the necessity remains that even if condition 2 was cancelled, a linearization mechanism would then be able to also produce a post-linearization tree such as the one in figure 10.

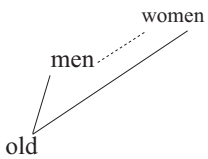


Fig. 10: Undesirable post-linearization tree of sentence (3)

This means that L, the linearization mechanism, must be able to make the necessary distinctions in order to prevent trees such as in figure 10 from occurring. This can be achieved quite easily if the following condition is formulated:

(C-1)　If a word X depends on another word Y *and* on another word Z then Y and Z must not be positioned on the same plane.

Since dependency connections in post-serialization trees such as Osborne proposes must run within a plane, i.e. dependency connections between words in different planes are prohibited, C-1 is sufficient to rule out a stemma such as the one in figure 10.

Then, it is necessary to again look at (10) and think about what L has to achieve. Stemmata are generated by − very simple − calculations on linear and dependency indices. Pre-linearization trees or their reformulations such as the statements in (10) contain information on the value of dependency indices, but not on the linear indices. In the normal procedure

of stemma generation, linear indices are known, and dependency indices have to be assigned after determination of dependency relations. Thus, (10) implies that the dependency index of *old* is equal to the linear indices of *men* and *women*. This statement is identical to saying that *old* is part of a double dependency relation. Therefore it is possible to rephrase (10) as (12).

(12.1) $_{men,women}old_{?}$
(12.2) $_{X}men_{?}$
(12.3) $_{X}women_{?}$

The task to be conducted by L is to assign linear indices. In order to do this, L has to refer evidently to basic compilation rules. For instance, *old* is an adjective, and English adjective must be positioned in front of nouns or respective substitutive expressions if they depend on them. Since *old* depends on a noun, it must be positioned in front of that noun. Therefore, *old* must be positioned either in front of *men* or in front of *women*.
Furthermore, old must also be positioned on plane P^1 which follows from the reading that both the men and the women can be old. Obviously, elements on planes with a higher plane index cannot refer back to elements on planes with a lower plane index. Therefore, if *old* depends both on *men* and *women*, it follows from C-1 that either *men* must be positioned on P^1 and women on P^2, or vice versa. Then, *old* must always be positioned on P^1 in order to refer to the element positioned on P^2. Therefore condition C-1 must be reformulated:

(C-I) If a word X depends on another word Y *and* another word Z then Y and Z must be positioned on different planes, and X must be positioned on the plane with the lower plane index.

If L adheres to condition C-I, then it will produce the next series of sequences:

(13.1) $_{2}old_{1,1}\ _{0}men_{1,2}\ _{2}women_{2,3}$
(13.2) $_{2}old_{1,1}\ _{0}women_{1,2}\ _{2}men_{2,3}$

The first index *after* a word denotes the plane index. Thus, in (13.1) *old* and *men* are positioned on P^1 and *women* on P^2. In (13.2), *old* and *women* are positioned on P^1, and *men* on P^2. The second index *after* a word denotes the actual linear index. Since P^1 must observe

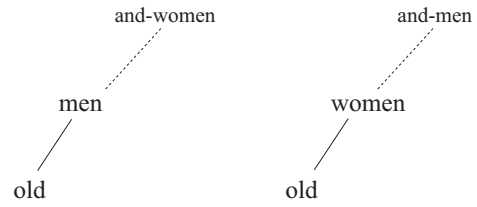
the CP, i.e. it must be a complete dependency structure. Thus, *men* receives dependency index {0} in (13.1), while *women* receives it in (13.2). Since according to C-I, *old* must be positioned on P^1, it follows that its dependency index must equal to the linear index of its governor which must also be positioned on P^1.

Problematic is the issue of the dependency index of the noun on P^2. Since in Osborne's post-linearization trees for SC a dotted line connects z-strings, let the element on P^2 receive as a proxy index a dependency index equal to the value of its predecessor on the same z-string. This index is held in bold script, so it can be distinguished from regular dependency indices. However, the sequences in (13) are not yet acceptable structures since in generating them L has not observed the second condition of Osborne's linearization rule for SC which states that for every departure from P^1 L must position a coordinator before P^2 starts. In particular, Osborne states that the coordinator attaches to the first word on a plane other than P^1. Applying this condition leads to (14).

(14.1) $_{2}\text{old}_{1,1}\ _{0}\text{men}_{1,2}\ \mathbf{_{2}}\text{and-women}_{2,3}$

(14.2) $_{2}\text{old}_{1,1}\ _{0}\text{women}_{1,2}\ \mathbf{_{2}}\text{and-men}_{2,3}$

L will then be able to output each possible structure of (14), as shown in figure 11.



(g) (h)

Fig. 11: Post-linearization trees of (14) as (g) and (h)

Sentence (4) shall be checked as well. Tree (b) in figure 3 contains the next information:

(15.1) from_{i} / men

(15.2) from_{i} / women

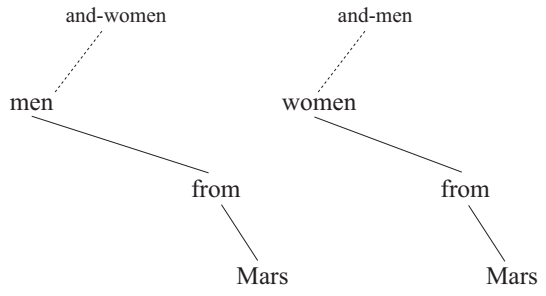
(15.3) Mars / from

Applying condition C-I and the second condition of the linearization rule for SC leads to (16):

(16.1) $_0$men$_1$ $_1$and-women$_{2,2}$ $_1$from$_{1,3}$ $_3$Mars$_{1,4}$

(16.2) $_0$women$_1$ $_1$and-men$_{2,2}$ $_1$from$_{1,3}$ $_3$Mars$_{1,4}$

Note the plane indices of the string *from — Mars* must be located on P^1 since from is dependent on both *men* and *women*. Adhering to C-I, it must be positioned on P^1. However, L must also position it after the noun on P^2 because otherwise *from Mars* is only considered as a dependent of the first noun, but not the second. Since from depends on both nouns, it must be positioned on P^1 and after the noun on P^2. Thus, L will produce the post-linearization trees shown in figure 12.



(i) (j)

Fig. 12: Post-linearization trees of (16) as (i) and (j)

Last, sentence (5) shall be checked, too. Tree (c) in figure 3 contains the following information:

(17.1) he$_{f1}$/ orders

(17.2) she$_{f1}$/ orders

(17.3) wine$_{f2}$/ orders

(17.4) beer$_{f2}$/ orders

The situation in (17) is reversed with respect to what was indicated in (10) and (15). There

one and the same word was dependent on two different words. In (17), all words are dependent on one and the same word. However, the words *he* and *she* are dependent on orders in the same way, i.e. they have the same syntactic function − here given as {f1}. This function is different from the syntactic function of wine and beer which also depend on orders. Osborne has anticipated this phenomenon and formulated his axiom of conjunction and his axiom of matching. This shall be globally reformulated here as condition C-II:

(C-II) If two words X and Y depend on another word Z, and if X and Y fulfill the same syntactic function F, then X and Y must not be positioned on the same plane.

According to C-II, L will produce the following structures when also applying the linearization rules for gapping:

(18.1) $_{2}\text{he}_{1,1}\ _{0}\text{orders}_{1,2}\ _{2}\text{wine}_{1,3}\ _{1}(\text{and-})\text{she}_{2,4}\ _{3}\text{beer}_{2,5}$

(18.2) $_{2}\text{she}_{1,1}\ _{0}\text{orders}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}(\text{and-})\text{he}_{2,4}\ _{3}\text{wine}_{2,5}$

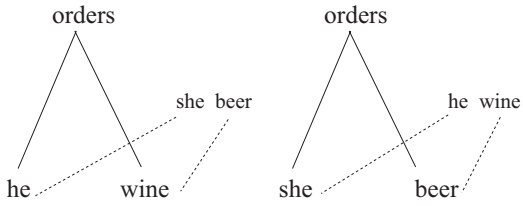
(18.3) $_{2}\text{he}_{1,1}\ _{0}\text{orders}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}(\text{and-})\text{she}_{2,4}\ _{3}\text{wine}_{2,5}$

(18.4) $_{2}\text{she}_{1,1}\ _{0}\text{orders}_{1,2}\ _{2}\text{wine}_{1,3}\ _{1}(\text{and-})\text{he}_{2,4}\ _{3}\text{beer}_{2,5}$

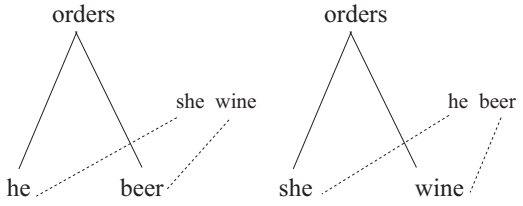
The difference to the outputs (14) and (16) which were semantically equivalent is that in (18) the sequences are not all semantically equivalent. (18.1) and (18.2) are equivalent but different from (18.3) and (18.4). However, (18.3) and (18.4) are again equivalent.

Also note that the second condition of Osborne's linearization rule for gapping is far more lenient than indicated by Osborne's formulation. A coordinator *can* occur, but it does not *need* to occur. However, there are cases in which a coordinator *must* occur. I will not go into this issue here.

The post-serialization trees for the sequences generated from (17) are shown in figure 13.



(k) (l)



(m) (n)

Fig. 13: Post-linearization trees of (18) as (k-n)

The correct tree for sentence (5) is (k).

5. Pre-serialization trees as probability sets

If the conversion of pre-serialization trees to post-serialization trees by L works as described in the previous section, then the problem arises that L will not output one post-serialization tree based on the same pre-serialization tree but many. As long as the basic compiling rules work, L will output at least acceptable sentences. Whether these basic compiling rules work, and how the format and constraints of such a compiler can be formulated will not be addressed here.

A problem pertinent to the proper working of L, however, is the following: the sequences (18) were generated under the assumption that L applied C-II and the linearization rule for gapping to (17). However, L cannot decide whether it must apply the linearization rule for gapping *or* the one for SC. It is true that (17) cannot be compiled and sequenced properly by L applying the linearization rules for SC. But how would the next sentence be sequenced?

(19)　　Bill ordered wine, and Mary beer.

The formalization of the pre-linearization tree is structurally equivalent to (17) with the exceptions that the pronouns are substituted by proper nouns, and that the verb is not in present tense. Applying the gapping rules, L will output sequences structurally equivalent to (18):

(20.1)　　$_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{wine}_{1,3}\ _{1}\text{(and-)Mary}_{2,4}\ _{3}\text{beer}_{2,5}$

(20.2)　　$_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Bill}_{2,4}\ _{3}\text{wine}_{2,5}$

(20.3)　　$_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Mary}_{2,4}\ _{3}\text{wine}_{2,5}$

(20.4)　　$_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{wine}_{1,3}\ _{1}\text{(and-)Bill}_{2,4}\ _{3}\text{beer}_{2,5}$

But L can now also apply the SC rules, and output the following sequences:

(21.1)　　$_{3}\text{Bill}_{1,1}\ _{1}\text{and-Mary}_{2,2}\ _{0}\text{ordered}_{1,3}\ _{3}\text{wine}_{1,4}\ _{4}\text{and-beer}_{2,5}$.

(21.2)　　$_{3}\text{Bill}_{1,1}\ _{1}\text{and-Mary}_{2,2}\ _{0}\text{ordered}_{1,3}\ _{3}\text{beer}_{1,4}\ _{4}\text{and-wine}_{2,5}$

(21.3)　　$_{3}\text{Mary}_{1,1}\ _{1}\text{and-Bill}_{2,2}\ _{0}\text{ordered}_{1,3}\ _{3}\text{wine}_{1,4}\ _{4}\text{and-beer}_{2,5}$.

(21.4)　　$_{3}\text{Mary}_{1,1}\ _{1}\text{and-Bill}_{2,2}\ _{0}\text{ordered}_{1,3}\ _{3}\text{beer}_{1,4}\ _{4}\text{and-wine}_{2,5}$

All sequences in (21) are semantically equivalent. However, they are not equivalent to any sequence in (20). This means that the pre-linearization tree of sentence (19) will output eight different sequences. As long as sentence (19) is among them, L works properly.

It is evident that L has to process a number of possible structures which can be calculated from the number of words that either depend on more than one other word, or that depend on the same word while having equivalent syntactic functions. In the sentences (3) and (4) there was only one word that depended on two different words, *old* in (3), and *from* in (4). The number of possible outputs is thus 2. In (5), however, two sets of two words each with equivalent syntactic functions respectively were established. The number of possible outputs is $2\times2=4$. In sentence (19), not only gapping but also SC rules could be applied. Thus, the number of possible outputs is $2\times2\times2=8$.

Quite evidently, the number of possible outputs will increase geometrically with the number of words to be conjoined or matched. The next sentence

(22) Bill ordered wine and beer, Mary beer.

must be generated from the next pre-linearization statements:

(23.1) Bill_{f1} /ordered
(23.2) Mary_{f1} /ordered
(23.3) wine_{f2} /ordered
(23.4) beer_{f2} /ordered
(23.5) beer_{f2} /ordered

Note that *beer* is not indexed with the identity index; it thus stands for two different words. However, semantically these words are indeed identical. The present conditions I and II do not capture this scenario. According to C-II, *beer*[1] and *beer*[2] will never occur on the same plane. While this is actually desired, the sequence *beer and beer* is not. However, this is such a basic violation of coordination that one need not formalize a constraint for it − if one is a human being. L, however, is not, and therefore, the next condition shall be formalized:

(C-0) Two identical words X and Y without further dependents may not be coordinated.

The expression without further dependents rules in utterances such as *beer from Germany and beer from Japan*.
L will now be able to output the next sequences applying the gapping rules, C-0 and C-II.

(24.1) $_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{wine}_{1,3}\ _{3}\text{and-beer}_{2,4}\ _{1}\text{(and-)Mary}_{3,5}\ _{3}\text{beer}_{3,6}$
(24.2) $_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{3}\text{and-wine}_{2,4}\ _{1}\text{(and-)Mary}_{3,5}\ _{3}\text{beer}_{3,6}$
(24.3) $_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Mary}_{2,4}\ _{3}\text{beer}_{2,5}\ _{5}\text{and-wine}_{3,6}$
(24.4) $_{2}\text{Bill}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Mary}_{2,4}\ _{3}\text{wine}_{2,5}\ _{5}\text{and-beer}_{3,6}$
(24.5) $_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{wine}_{1,3}\ _{3}\text{and-beer}_{2,4}\ _{1}\text{(and-)Bill}_{3,5}\ _{3}\text{beer}_{3,6}$
(24.6) $_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{3}\text{and-wine}_{2,4}\ _{1}\text{(and-)Bill}_{3,5}\ _{3}\text{beer}_{3,6}$
(24.7) $_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Bill}_{2,4}\ _{3}\text{beer}_{2,5}\ _{5}\text{and-wine}_{3,6}$
(24.8) $_{2}\text{Mary}_{1,1}\ _{0}\text{ordered}_{1,2}\ _{2}\text{beer}_{1,3}\ _{1}\text{(and-)Bill}_{2,4}\ _{3}\text{wine}_{2,5}\ _{5}\text{and-beer}_{3,6}$

Note that the words *beer* do not appear on the same planes. Since C-0 prevents SC from

occurring the sequences (24) are all those that L can output. What L in effect does is to produce a set of possible sequences of which one must be the one that has been actually encountered. In this respect, pre-linearization trees are probabilistic devices.

A probabilistic mechanism is something very desirable for DG, since DG has inherent problems with the treatment of word order: in general a DG cannot predict the word order of utterances not yet encountered since a DG lacks an abstract stratum. This means that DGs usually do not implement a level of description in which a rather fixed syntactic structure is viewed as a basic structure to which transformation rules are applied in order to incur word order changes. Since − as it was outlined in section 1 − DGs are monostratal they do not employ such a description level. Thus, without for instance functional terminology, a DG cannot handle word order issues very well.

The constraints a DG has to work under, limit the theoretical accessibility of such a necessary abstract level of description. While constituency grammars such as GB-Theory or Minimalism work *top-down*, DGs usually work *bottom-up*. DGs access the actual data first, and then make assumptions about the sentence structure. Constituency grammars work from the assumption that an abstract sentence structure exists, and interpret the language data in a way as to fit this assumed abstract sentence structure.

This insight into the workings of different grammar theories is, however, also applicable to the pre-linearization trees Osborne proposes. In order to arrive at pre-linearization trees which contain only information on dependency connections, first an actual utterance has to be analyzed. This analysis will necessarily also contain information on linear order, since utterances of natural languages cannot but have linear order. This means that the analysis will produce information on linear order (linear indices) and dependency (dependency indices). In order to arrive at a pre-linearization tree, the information on linear order is simply disregarded.

This means that a linguist must have made assumptions on dependency connections *before* she proposes a pre-linearization tree. Therefore, considering a sentence such as (3), the linguist finds four words: *old*, *men*, *and*, and *women*. In order to arrive at a post-serialization tree such as (g) in figure 12, the linguist must then first assume that the coordinator *and* does not constitute a single node in the tree. This assumption must be based on different insights that just the assurance that in doing so, a viable theory will emerge. Epistemologically speaking, Osborne's theory of coordination does not explicate the workings of coordinators and coordination, but only assumes a specific way of coordination to work. It is organized

in a slightly circular way, since the theory cannot emerge if established detection procedures are conducted, but rather it must reorganize the data recovered by detection procedures under the assumption that some words have completely different syntactic roles than others.

It is however, widely accepted that there are at least two main classes of words (or word-forms): *function words* and *content words*. Modern constituency grammars − including Universal Grammar − view this distinction as a core tenet. If DGians want or need to implement this distinction they have to consider the implications for the theory of DG. It will ultimately lead away from the focus on data empirics and towards a model, quite similar in idea but not in form to what constituency grammars already offer.

The notion of pre-linearization trees as probabilistic devices that can output a set of structured utterances of which one utterance must be the one under observation is helpful. It can help to bring modern linguistics in line with other scientific areas that are involved in the research of human cognitive abilities and where statistics and probabilities play a central role.

Literature:

Gross, Thomas (1992): Konstruktive Stemmatologie. In: Papiere zur Linguistik, Nr. 47. p115-139.

— (1999): Theoretical Foundations of Dependency Syntax. Műnchen: iudicium.

— (2002): Coordination as a linguistic speed-up mechanism. In: Language and Culture No.7. Institute for Language Education, Aichi University. p39-68.

Mel'čuk, Igor (1979): Studies in Dependency Syntax. Albany.

Osborne, Timothy (2003a): The Third Dimension. Ph.D.-Dissertation. Pennsylvania State University.

— (2003b): String Coordination: A 3d Dependency Approach to Coordination. In: Civilization 21, No.10. Association for International Communication, Aichi University. p91-111.

Tesniére, Lucien (1959): Elements de syntaxe structurale. Paris: Klingensieck. Grundzűge der strukturalen Syntax. Stuttgart: Klett-Cotta. 1980. Engel, Ulrich (ed.)