Why to choose dependency rather than constituency for syntax: a formal point of view

Sylvain Kahane

Modyco - Université Paris Ouest Nanterre & CNRS
sylvain@kahane.fr

Abstract

In this paper we want to come back again to the question of the equivalence or non-equivalence between dependency and constituency. Many arguments have been proposed. In this presentation, we will focus on formal arguments concerning the respective descriptive capacities of these two ways of representation. We show that, if headed constituent trees are equivalent to stratified dependency trees, weakened or enriched versions of one of these formalisms cannot be simulated by the other one.[1]

Keywords

Dependency grammar, phrase structure grammar, stratified dependency tree, headed constituent tree.

1 Introduction

The debate concerning the choice between constituency and dependency in syntax is not new. Even if the tendency is reversing, the last decades of linguistic modeling have been dominated by constituency-based models and the discussion of the advantages and disadvantages of the

[1] This paper is dedicated to the 80th birthday of Igor Mel'čuk with whom I have had so many discussions about the arguments to prove why the syntactic representation is a dependency tree. (Contrarily to Igor, I do not think that the syntactic structure is as simple as a tree even if I am convinced that it is dependency-based. And I do not think that the fact that the core of the syntactic representation is dependency-based excludes a constituency-based level of representation: see for instance, Gerdes & Kahane (2006), the paper we wrote for his 70th birthday.) The most decisive argument of Igor is that there are dependency trees in his brain when he speaks and that he can see them. I never doubt that it is true, but I never considered that as a proof. One of the main reasons is that Igor's brain is really not representative of an ordinary speaker's brain. It is the brain of one of the most brilliant linguist of all the times and moreover the brain of a polyglot who has developed special skills to learn languages. His brain has become the first complete implementation of the Meaning-Text theory! Igor, I am sure we will continue to improve your implementation for many years.

two ways of representation have been essentially sustained by syntacticians working with dependencies. Several kinds of arguments have been developed.

A first range of arguments does not concern the syntactic representations themselves, but the way they are dealt with by the brain. For instance, Tesnière does not justify the introduction of dependency by linguistic arguments but by a "mentalist" argumentation: "Every word that is part of a sentence ceases to be isolated as it is in the dictionary. **The mind perceives** connections between a word and its neighbors. The totality of these connections forms the scaffold of the sentence." (Tesnière 1959, Chapter 1, 3rd paragraph; I underline) (see also Note 1). There is of course no way to prove that today, but such arguments must not be rejected too quickly. We now have more information on the functioning of the brain and we can suppose that our memory consists of a network of knowledge and that our main mental activity is to connect things together. Such arguments have been used to defend dependency (Hudson 1993, 2007).

A second range of arguments concern the correspondences between the syntactic representation and other levels of representation. These arguments must be separated from the previous ones even if they are related: they do not concern the process but instead the correspondence rules considered from a purely declarative point of view. These arguments concern the advantages of a dependency-based structure for the interface with text (the linear rules) and for the interface with the meaning (the syntax-semantics interface). Concerning linearization, it is well known that dependency syntax can deal with non-projective structures (see for instance Mel'čuk & Pertsov 1987, Gerdes & Kahane 2007), while constituency-based models need the addition of complex mechanisms like transformations, movements and so on (see Gerdes 2006 for an argumentation). For the syntax-semantics interface, the Meaning-Text theory is probably one of the first theories to integrate it in a formal linguistic model. It has been shown that a dependency tree is closer to a semantic representation and that dependency-based approaches are more adapted to processing of valency constraints and of multi-word expressions and particularly lexical functions (Mel'čuk 1988, Iordanskaja & Polguère 1988, Polguère 1990, Milićević 2007). These arguments are probably the most important and convincing in the comparison between constituency and dependency, but this is not the point we want to develop here.

A third range of arguments concern the expression power of the different formalisms: are there syntactic configurations that can be described in terms of dependency-based structures and not in terms of constituency-based structures and vice versa? This is the point we want to develop here, considering the problem more from a mathematical point of view rather than form a linguistic one. In this paper, we will first (Section 2), starting from a question of syntax arising in algebra and not in natural languages, explore the extent the two representational formats are equivalent. Subsequently (Section 3), we study the limits of this equivalence according by weakening or enriching the one or the other format.[2]

[2] I would like to address again some words to Igor Mel'čuk. Even if this paper contains principally results in mathematics, this paper has been written for linguists and particularly for linguists like Igor Mel'čuk, who knows the importance for linguists to work with mathematicians and to have themselves a mathematical culture. I take advantage of this anniversary to thank Igor for the great interest he has in our collaboration. I am eternally grateful to him for all I learn from him and from the enormous help he has given me to enter the world of linguistics.

2 Representation of the structure of a calculation

We start our presentation with the study of the structure of an elementary algebraic calculation. We want to show that the problem of the choice of a syntactic representation is not a question of linguistics and to appreciate, outside the internal debates of this discipline, the qualities of the different possible representations of the structure of a calculation.

Let us consider the formula (7 × 19) + (5 × 31). To perform this calculation, we must perform the two multiplications before summing their results. It can be represented by the following diagram:

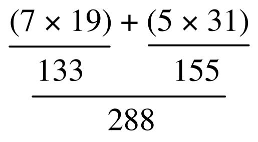

Figure 1. Diagram associated to a calculation

Each intermediate calculation involves two numbers and one operator. The structure of the whole calculation can be represented in the following way:

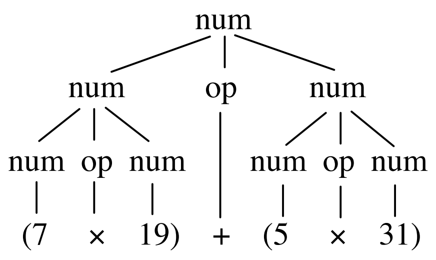

Figure 2. Constituent structure of a calculation

Such a structure is called a *constituent tree* (CT).[3] The elements of the formula (we omit the parentheses) are the leaves of the tree and the internal vertices represent the constituents of the formula. Each internal vertex receives a label indicating the nature of the constituent –

[3] The immediate constituent analysis has been theorized by Leonard Bloomfield (1933). But contrary to Chomsky, Bloomfield did not draw any CTs. Above all, he gave more importance to the notion of head and to the asymmetry of the relation between the head and the other constituents than to the notion of constituent itself. His definition of the notion of constituent is first given in the chapter on morphology where he defines the morpheme. In the chapter on syntax, it is said that "Syntactic constructions are constructions in which none of the immediate constituents is a bound form. [...] The actor-action construction appears in phrases like: *John ran, John fell, Bill ran, Bill fell, Our horses ran away*. [...] The one constituent (*John, Bill, our horses)* is a form of a large class, which we call nominative expressions; a form like *ran* or *very good* could not be used in this way. The other constituent (*ran*, *fell*, *ran away*) is a form of another large class, which we call finite verb expressions; a form like *John* or *very good* could not be used in this way." We think that in some sense Bloomfield should rather be seen as a precursor of the notion of *connection* (called here *construction*) and dependency than as the father of constituency. I think it is really Chomsky who considered that the immediate constituent relation was more important than the head-daughter relation and who based the syntactic representation on a CT. And Chomsky popularized the constituency-based syntactic representation with his formalization of grammar rules by context-free grammars.

number or operator. Some numbers are the explicit numbers in the formula, the others are the result of a calculation. The CT of Fig. 2 shows how the whole calculation is done. It is similar of the diagram of Fig. 1: the CT can be converted into this diagram by replacing the labels number by their numeric value and the vertices operator by horizontal lines modeling the calculation.

A calculation where a unary operator intervenes like the square root in 1 + √5 can be represented in the same way:

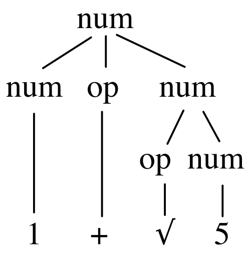

Figure 3. Constituent tree of another calculation

Considering our two examples, we see that the constituents number can be decomposed in two or three sub-constituents. The element that decides the internal structure of a constituent is the operator, which can be unary, like √, or binary, like + and ×.

The element that controls the internal structure of a constituent is called its *head*. An element X *governs* the sub-constituents of the constituent whose head is X and conversely a constituent Y *depends* on an element X if Y is a sub-constituent of the constituent whose head is X.

Another tree-like representation, called a *dependency tree* (DT), can thus be adopted by replacing each constituent by its head and by placing each element under its governor:

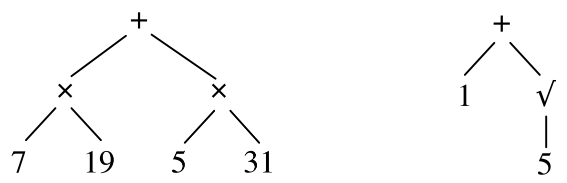

Figure 4. Dependency structures of two calculations

The CT can be easily reconstructed from the DT. They are still present in the DT: the constituents are the projections of the sub-trees of the DT. A *sub-tree* is a tree formed by all the vertices that are under a given vertex, this vertex included; the *projection* of a sub-tree is the segment of the formula composed by all the elements labeling this sub-tree. The following figure shows how the CT can be recovered from the DT, simply by considering vertices and projections as constituents (cf. Lecerf 1961 or Kahane 1997 for a formalization).

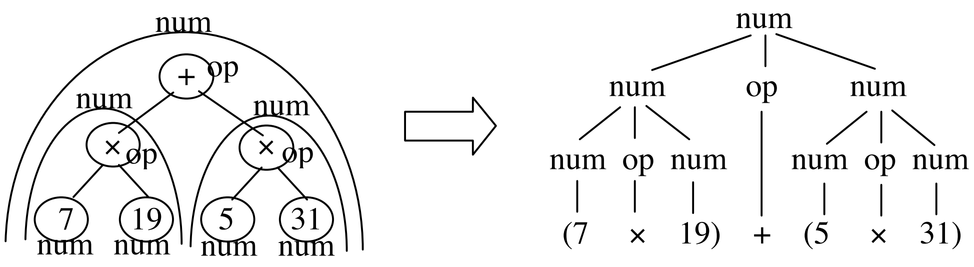

Figure 5. Changing from a dependency tree to a constituent tree

On the other hand it is necessary to add the information concerning the head of each constituent in order to change from a CT to a DT. Thus a DT, though it is simpler (it has half the number of vertices), contains more information than a CT!

So far we were only interested by the hierarchical structure and, even if it was not specified, the trees we have considered were not ordered.[4] A CT is *ordered* if the daughters of each vertex are linearly ordered, which induces a linear order on the leaves of the tree. To imagine an unordered tree, one must think of the tree as a mobile that is suspended by its root, daughters of a vertex freely moving in the air. CTs are generally considered as more interesting for the study of linear precedence constraints because they are naturally ordered by the planar representation imposed by the sheet on which they are drawn. But, as we want to show now, it can be more advantageous to deal with linear precedence constraints starting with a DT rather than a CT, particularly when the linear order does not come immediately from the hierarchical structure.

Before moving to natural language utterances, let us recall that dependency structures for formulae have been introduced following the work of the Polish logician Jan Łukasiewicz, who showed that there are several ways of encoding a formula linearly. A first way is the one we use when we write the formula (7 × 19) + (5 × 31), where each binary operator is placed between its two dependents. This writing needs parentheses in order not to be ambiguous. Another writing, called Polish or prefixed writing, consists of reading the tree by always collecting the governor before its dependents and by collecting all the dependents from left to right: + × 7 19 × 5 31. This formula is not ambiguous, despite the absence of parentheses, as soon as the arity of each operator is known. The *arity* of an operator is the number of its arguments and then of its dependents in the DT. The *postfixed* or *inverted Polish reading*, where the governor is collected after its dependents, is particularly appropriate for calculus and is used by automatic calculators, including some consumer calculating machine. Indeed all you have to carry out the calculation 7 19 × 5 31 + × is to pile the number as they come and to apply each operator to the one or two numbers preceding it (depending on its arity) as soon as it appears and to replace it and the consumed numbers by the obtained number:

[4] Traditionally DTs are by default unordered, like Tesnière's representations, while CTs are ordered, as in Chomsky's work. But since the beginning of the 1980s, linguists, such as Gazdar *et al.* (1985), have strived – and this is true even of constituency-based formal grammars – to distinguish between the syntactic hierarchy (or immediate dominance) from the linear order (the same was done a couple of decades before by Tesnière (1959) in dependency grammar). Present phrase structure grammars, like HPSG (Pollard & Sag 1994), handle the immediate constituency constraints and the linear precedence constraints separately.

7			
19			
×	*133*	*133*	
	5		
	31		
	×	*155*	
		+	*288*

Figure 6. Calculation from a postfixed writing[5]

These various strategies to go from a hierarchical structure to a linear order can be seen again in natural languages and we can consider that grammar of languages come from the need to encode information non linear by essence linearly because the speech signal we produce when speaking is linear.[6]

3 Equivalence and non-equivalence

Section 2 showed on an example the equivalence between DT and headed CT. This section is more technical and clarifies the exact nature of this equivalence. We will first state that the simulation of any CT needs an enrichment of the DT formalism, which gives us stratified DTs (Section 3.1). If headed CTs and stratified DTs are equivalent, it is not true that weakenings (Section 3.2 and 3.3) or enrichments (Section 3.4) of these formalisms necessarily have an equivalent counterpart.

3.1 Headed CT vs. stratified DT

As seen before, any headed CT induces a DT. When transforming a headed CT into a DT, all constituents having the same lexical head are aggregated. In a headed CT, every constituent

[5] This figure tries to represent the evolution of the calculation during the process. Each time a calculation is carried out and elements are suppressed from the pile we create a new colon to show the new content of the pile.

[6] I want to stress that the syntactic structure of a calculation is fundamentally unordered (although for non-commutative operators like subtraction or division, the order of numbers is relevant). The linear order is only needed because we want to describe the calculation by a linear writing. The same is partly true for natural languages. Each language imposes particular order constraints: some languages, like Korean or Turkish, are *head-final*, that is the head of each constituent is at its end (Tesnière (1959:22-33) says they are *centripetal* because when reading the sentence we go toward the center of the sentence which is the root of the DT), while others, like Welsh, are *head-initial* (or *centrifugal* in Tesnière's terms). And others, like English or French are *head-medial*, the head of a constituent being in the middle of its dependents. But in these different languages, there is a similar hierarchical structure for sentences and the different orders in different natural languages correspond to different conventions of "reading" of this hierarchical structure: a prefixed reading for head-initial languages and a postfixed reading for head-final languages. This is why, following Tesnière and Mel'čuk, I really think we need to separate the syntactic structure (called the *structural order* by Tesnière) from the linear order and to consider unordered DTs. Moreover, there are non-configurational languages (sometimes abusively called free-order languages), like Russian or Walpiri, where the linear order has a certain independence from the syntactic structure and is used to express information related to the communicative structure.

having *x* as lexical head is called a *projection* of *x*. The biggest one is called the *maximal projection* of *x* and the others are called the *intermediate projections* of *x*. Only maximal projections can be recovered from the DT. As a consequence, the same DT can be induced by different headed CTs.

But it is possible to enrich the DT formalism to recover the intermediate projections and to totally simulate a headed CT. The canonical solution consists of introducing different levels for the dependencies of a same node. It can be done by stratifying nodes and attaching the dependents to different strata (the third structure in Fig. 7) or by adding types to the dependents of each node (the fourth structure in Fig. 7, where the two dependents are typed as int(ernal) and ext(ernal)). We call such a structure a *stratified DT*.

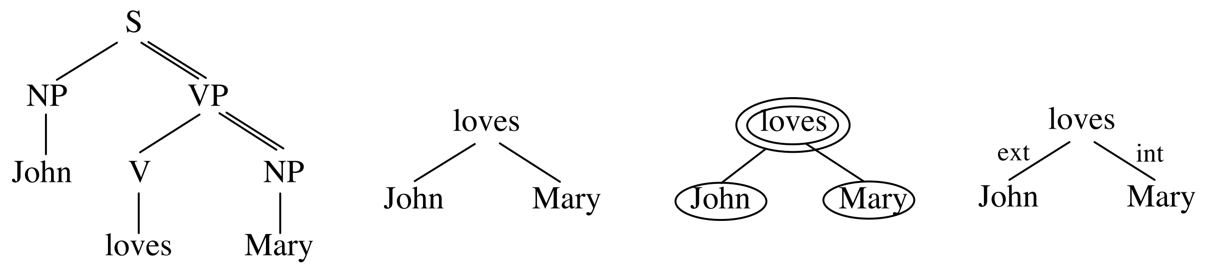

Figure 7. A headed CT with the corresponding DT and stratified DT (with two conventions for the stratified DT)

The first structure in Fig. 7 is a headed CT: the head of a constituent is the sub-constituent marked by a double line. For instance, the head of S is VP. The *lexical head* is obtained by looking recursively for the head until we find a lexical item, that is a leaf of the CT. The lexical head of S is the lexical item *loves*.

The second structure in Fig. 7 is the corresponding DT, describing the dependency relations between the lexical items. It can be obtained by aggregating each constituent with its lexical head; for instance, S, VP and V nodes are aggregating with their lexical head *loves*.

The third structure in Fig. 7 is a first representation of the corresponding DT: here, the different strata are explicitly represented by embedded bubbles (see Kahane (1997) for a formalization of *bubble trees*). Each of these strata corresponds to a different projection of the same lexical item and each dependent is attached to the stratum corresponding to the projection it belongs to: the subject *John* is attached to the stratum corresponding to S and the object *Mary* is attached to the stratum belonging to VP.

The fourth structure of Fig. 7 is an alternative representation of the same stratified DT. It can be interpreted only if we know the obliqueness order on the types of dependents, which allows us to construct different projections for the same node. For instance, if int < ext, *loves* will have two projections: a projection with the dependents of obliqueness ≤ int containing only *Mary* (that is VP) and a projection containing the dependent of obliqueness ≤ ext containing both *John* and *Mary* (that is S). We use the labels *internal* and *external* (used by today works in Generative Grammar), but we can replace them by bar1 and bar2 (with bar1 < bar2) to directly obtain X-bar categories for our constituents (and follow the conventions proposed by Jackendoff (1977) for headed CTs). It is clear that we can reconstruct the original headed CT, including the labels on nodes.

3.2 When DTs cannot simulate plain CTs

It is generally admitted that "he who can do more can do less". But this piece of wisdom is not true for formal structures: if DTs can simulate headed CTs, they cannot simulate plain CTs, that is CTs without heads.[7] In other words, a DT can simulate a headed CT but it cannot simulate a weaker structure like a CT.[8]

Let us explain that with an example. Consider the following CT for the sentence *The dogs are barking*:

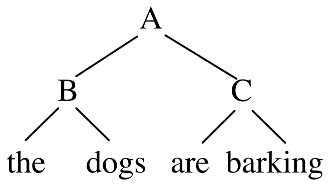

Figure 8. A non-headed CT

Without any information about the heads of constituents B and C, it is not possible to decide which of the lexical items *the* and *dogs* must be connected to which of the lexical items *are* and *barking* and every combination is possible (according to different choices of heads). In other words, it is not possible to draw a DT for this sentence without adding information concerning the head of each constituent and hence no DT can simulate this plain CT.

3.3 When (headed) CTs cannot simulate dependency-based structures

If headed CTs can simulate DTs, they cannot simulate connection graphs, that is a "dependency structure" where "dependencies" are not directed but are only symmetric connections between two nodes (see Gerdes & Kahane 2011 for a formalization of connection structures). In other words, a headed CT can simulate a DT but it cannot simulate a weaker structure like a connection graph, and there is no way to weaken a (headed) CT for that.

Consider a connection graph for *Peter and Mary*:

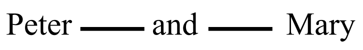

Figure 9. A connection graph

This non-directed graph can be seen as a subspecified DT, like the following ones:

---

[7] We use the notion of *simulation* with a technical sense. A set of structures A *simulates* a set of structures B if there is a canonical transformation from A to B where each structure in B is the transformation of at least one structure in A.

[8] *Weaker* must be understood in a purely mathematical sense. A mathematical structure S is weaker than a mathematical structure S' if S contains less information than S' and S can be structurally enriched to obtain S'. For instance, a non-directed graph is *weaker* than a directed graph: you can enrich a non-directed graph by directing each connection (= edge, in mathematical terminology) and obtain a directed graph.

$$\begin{tikzcd}\text{Peter} \arrow[r] & \text{and} \arrow[r] & \text{Mary} & & \text{Peter} & \text{and} \arrow[l] \arrow[r] & \text{Mary}\end{tikzcd}$$

Figure 10. Two DTs corresponding to the same connection graph

But this connection graph cannot be simulated by a constituency-based structure without specifying some hierarchy on the connections. Indeed, as just illustrated, a CT cannot define connections or dependencies without head marking. And a headed CT defines dependencies and not only connections. It is thus impossible to just simulate a connection graph with a constituency-based structure.

This non-equivalence between connection graphs and CTs is very representative, I think, of the difference in philosophy between dependency- and constituency-based approaches. A connection graph defines a set of syntactic units: in our example, *and* is connected with both *Mary* and *Peter*, without privileging either of the two connections.[9] But a CT must privilege one of the two connections and decide to regroup *and* and *Mary* or to regroup *Peter* and *and*. If not, the phrase *Peter and Mary* will remain unanalyzed, without internal syntactic structure.[10]

A last remark concerns connections and constituency. Tesnière was one of the first to propose a formal representation of coordination. He considered that in coordination, two elements (or more) occupy the same syntactic position. The combination between these two elements, called *junction* by Tesnière, is thus an operation of combination orthogonal to the dependency (which is restricted to the combination between a governor and a dependent). He advocated a symmetrical analysis of coordination and proposed to represent junction by horizontal links, orthogonal to the vertical dependency links. Consider the following French sentence and its syntactic structure:

[9] This difference in philosophy is well illustrated by the introduction to constituency given by Gleason (1961:129-130): “We may, as a first hypothesis, consider that each of the words [of a utterance] has some statable relationships to each other word. If we can describe these interrelationships completely, we will have described the syntax of the utterance in its entirety. […] We might start by marking those pairs of words which are felt to have the closest relationship.” So far what Gleason writes is compatible with a dependency-based approach. But he adds the following assumption without any justification: “We will also lay down the rule that each word can be marked as a member of only one such pair.” and declares the method of finding the best among all the possible pairings to be “the basic problem of syntax”. On the contrary, we can exploit the fact that natural syntactic units are not pairwise disjoint. This defines several possible fragmentations of a utterance into constituents. Formally each of these fragmentations is a CT, even if only one of these CTs is recognized as the CT of the sentence by grammarians working in constituency-based approaches. Gerdes & Kahane (2011) show that this multiplicity of fragmentations naturally defines a DT and that consequently DTs are richer structures in the sense that they contain informatively a multiplicity of fragmentations and not only one (even if the hierarchy in a DT allows to privilege one, which is the CT of maximal projections). See also Osborne (2005), Groß & Osborne (2012) for a similar exploitation of DTs; they show that a DT defines many more syntactic units (which they call *catenae*) than a CT and that most of these syntactic units can be as relevant for linguistic modeling as the syntactic constituents privileged by constituency-based approaches.

[10] I am not discussing here whether or not a symmetrical analysis is more relevant than an asymmetrical one (see Kahane 2012 for a discussion). I just take this example to illustrate the non-equivalence of two formalisms. There are well-known arguments (see for instance Mel’čuk 1988) showing that *and Mary* is more “cohesive” than *Peter and*.

(1)

Alfred	*a*	*acheté*	*un*	*livre*	*et*	*un*	*cahier*	*neufs*
Alfred	has	bought	a	book	and	a	notebook	new(plural)

'Alfred bought a new book and a new notebook'

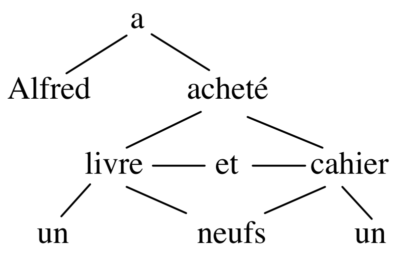

Figure 11. A syntactic structure with dependencies and junctions (from Tesnière 1959:340)

One has argued several times that, by introducing junction, Tesnière was proposing a syntactic representation mixing constituency and dependency (Hudson 1980, Sangati 2012). And even that coordination needs a constituency-based representation. But junctions are just non-directed connections and such connections cannot be simulated by a constituency-based formalism. There is no more constituency in junction than in dependency.[11]

3.4 Equivalence and non-equivalence of enriched structures

A particularity of DTs is that they presuppose that each constituent has exactly one lexical head. But in constituent structures, it is possible to consider constituents with co-heads. It has been for instance proposed for the symmetrical analysis of coordination where conjucts are considered as co-heads (Jackendoff 1977). It is possible to enrich the dependency formalism to deal with co-heads.

Let us for instance, consider the sentence *The little boy ran*. We can consider that the choice between D and N for the head of a DP/NP is not relevant[12] and that D and N are co-co-heads (indicated by double lines):

[11] Tesnière's conventions do not allow to formalize nested coordinations like in *I would like to find someone who speaks French or Italian and German*, where two interpretations are possible: [French or Italian] and German vs. French or [Italian and German]. But this can be formalized by bubble trees (see Kahane 1997, 2012 and the next section), which are not constituents either.

[12] Again, it is not our purpose to discuss here if there exist or not constituents with co-heads and to decide what is a head of NP/DP. It is well known that there are arguments both in favor of D and N (see Abney 1987 for the beginning of the debate about "DP-hypothesis"). My conclusion is that none of the two lexical items controls alone the distribution of the phrase and that not considering one of the two items as the only head is a possible solution. As it can be seen here, this solution is compatible with a dependency-based formalism and is not so complicated (for people accepting that there can be something else than trees in syntax).

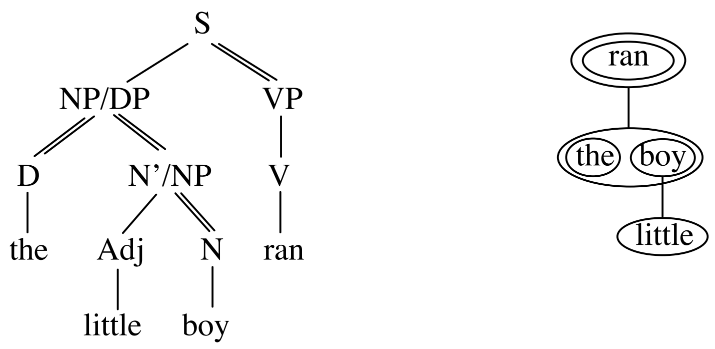

Figure 12. A co-headed CT and the corresponding bubble DT

This co-headed CT can be simulated by an extension of stratified DTs, where co-heads are placed in a bubble. We call such a structure a *bubble DT* (see Kahane 1997 for a formal definition and a formalization of the correspondence between co-headed CTs and bubble DTs). The whole bubble *the boy* depends on the governor *ran*, while *little* depends only on *boy*.

One might state that bubble DTs combine both dependency and constituency, but that would be a false interpretation of these structures. First, the content of the bubble *the boy* is not a constituent of the sentence because these two lexical items are not adjacent. Second, the grouping of *the* and *boy* and the link between this group and their governor *ran* might be interpreted as a 3-node connection. What a bubble DT encodes is the fact that it is possible that lexical items could not be only pairwise connected. Neither *the ran* nor *boy ran* is an acceptable fragment of the sentence, so none of these pairings is acceptable: the verb *ran* needs both D and N to validate the connection.

The last example we want to discuss concerns circular connections. There are cases where three lexical items can all be pairwise connected: A and B, B and C but also A and C. For instance, Hudson (2007) defends, for an equi-construction like *Peter wants to come*, that *Peter* depends both on *wants* and *come*.[13] This is very easy to formalize with a dependency graph (Fig. 13). It need merely accept nodes with two governors. The same analysis can be done in constituency-based systems, but in contrast to dependency-based approaches, constituency needs to employ empty nodes and coindexation (Peter$_i$ wants [ε_i to come]), which is much more costly.

[13] Mel'čuk (1988, 2003) considers that the dependency between *Peter* and *to come* is a semantic dependency and must not appear in the syntactic structure. Nevertheless, the condition on this semantic relation is syntactic: the subject of *want* is shared with the potential *subject* of its verbal complement. For instance, for obtaining a semantic relation with the second argument of the verbal complement, we need to passivize it: *Peter wants to be recognized* (see Kahane (2011) for a formalization introducing such syntactic dependencies in the syntax-semantics interface rules). There are other cases where such configurations have been proposed. Gerdes & Kahane (2011) consider examples like *the rise of nationalism in Catalonia*, where the phrase *in Catalonia* can be connected to both *rise* and *nationalism* without significant change of meaning. The most puzzling case is certainly *wh*-words. As shown by Tesnière (1959), *wh*-words are both pronouns and complementizers and must for this reason occupy two positions in the syntactic structure (see Kahane 2002 for an in-depth argumentation for Tesnière's analysis). Generative grammars propose a very similar constituency-based analysis with the coindexation of a complemetizer and an argumental position.

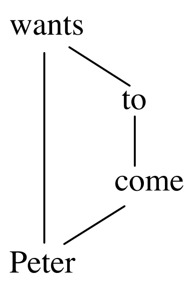

Figure 13. Dependency graph

4 Conclusion

We have explored the equivalence between dependency-based and constituency-based approaches to syntax, showing that headed CTs and stratified DTs can simulate each other. It implies that, in many cases, the two kinds of structures are possible representations and the choice is rather a matter of fashion. Due to the influential work of Chomsky (1957), constituency-based representations have dominated linguistics during the second half of the 20th century. But now that the models are more sophisticated, taking into account the semantics and the diversity of the lexicon (including complex discontinuous constructions), the trend seems to be reversed. In this paper, we have focused on some details of the formal equivalence showing that dependency-based formalisms can have some advantages by not privileging certain syntactic units (the X-bar constituents) at the expense of others. Moreover, the dependency formalism can be easily enriched to take into account co-heads or multi-node dependencies. If one believes, like I do, that syntactic structures are more complicated than the structures that current models handle, dependency-based approaches could actually be more promising than constituency-based ones, allowing consideration of more syntactic units (without privileging constituents too much), relaxation of the hierarchy (with non-directed connection) and more complex relations (with 3-node connections or multi-governed position).

Acknowledgements

I would like to thank Tim Osborne, Paola Pietrandrea and Alain Polguère for useful comments and corrections on the first version of this paper. Exceptionally, one of my papers has not been revised by Igor Mel'čuk. That's the price of the surprise.

Bibliography

Bloomfield L. (1933), *Language*, New York.

Chomsky N. (1957), *Syntactic Structure*, MIT Press, Cambridge.

Gazdar G., Klein E., Pullum G., Sag I., *Generalized Phrase Structure Grammar,* Harvard University Press.

Gerdes K. (2006), Sur la non-équivalence des représentations syntaxiques : comment la représentation en X-barre nous amène au concept du mouvement, *Les Cahiers de Grammaire*, *30*, 175-192.

Gerdes K., Kahane S. (2007), Phrasing It Differently, in L. Wanner (ed.), *Selected lexical and grammatical issues in the Meaning-Text Theory*, Benjamins, 297-335.

Gerdes K., Kahane S. (2011), Defining dependency (and constituency), *Proceedings of the 1st international conference on Dependency Linguistics* (*Depling*), Barcelona, 17-27.

Gleason H. A. (1965), *An Introduction to Descriptive Linguistics*. New York: Holt, Rinehart & Winston, 503 pp. Revised edition 1961.

Hays D. (1964), Dependency theory: A formalism and some observations, *Language*, *40:4*, 511-525; Rand Corporation, 1960, RM-2646 (Grouping and dependency theories).

Hudson R. (1980), Constituency and Dependency, *Linguistics*, *18:3/4*, 179-198; A Second Attack on Constituency: A Reply to Dahl, *Linguistics*, *18:5/6*, 489-504.

Hudson R. (1993), Do We Have Heads in Our Minds?, in Corbett G., Fraser N., McGlashan S. (eds), *Heads in Grammatical Theory*, Cambridge Univ. Press, 266-291.

Hudson R. (2007), *Language Networks: The New Word Grammar*, Oxford University Press.

Iordanskaja L., Polguère A. (1988), Semantic processing for text generation, Technical report, ORA, Canada.

Jackendoff R. (1977) *X-bar Syntax. A Study of Phrase Structure*, MIT Press, Cambridge.

Kahane S. (1997), Bubble trees and syntactic representations, *Proc. 5th Meeting of the Mathematics of Language (MOL5)*, DFKI, Saarbrücken, 70-76.

Kahane S. (2001), A fully lexicalized grammar for French based on Meaning-Text theory, *Computational Linguistics, Proc. CICLing 2001*, Mexico, Springer Verlag, 18-31.

Kahane S. (2002), A propos de la position syntaxique des mots qu-, in P. Le Goffic (éd.), *Interrogation, indéfinition, subordination, Verbum, XXIV:4*, 399-435.

Kahane S. (2012), De l'analyse en grille à la modélisation des entassements, in S. Caddeo, M.-N. Roubaud, M. Rouquier, F. Sabio, *Hommage à Claire Blanche-Benveniste*, Presses de l'Université de Provence, 12 p.

Kahane S., Mel'čuk I. (1999), La synthèse sémantique ou la correspondance entre graphes sémantiques et arbres syntaxiques – Le cas des phrases à extraction en français contemporain*, Traitement Automatique des Langues, 40:2*, 25-85.

Lecerf Y. (1961) Une représentation algébrique de la structure des phrases dans diverses langues naturelles, *Comptes Rendus de l'Académie des Sciences de Paris, 252*, 232-34.

Łukasiewicz J. (1930), Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls, *Comptes rendus des séances de la Société des Sciences et des Lettres de Varsovie*, *23*, 51-77. English transl.: H. Weber (1967), Philosophical Remarks on Many-Valued Systems of Propositional Logics, in S. McCall (ed), *Polish Logic 1920-1939*, Clarendon Press, Oxford.

Mel'čuk I. (1988), *Dependency Syntax: Theory and Practice*, The SUNY Press, Albany, N.Y.

Mel'čuk I. (2003), Levels of Dependency in Linguistic Description: Concepts and Problems. In V. Agel, L. Eichinnger, H.-W. Eroms, P. Hellwig, H. J. Herringer, H. Lobin (eds): *Dependency and Valency. An International Handbook of Contemporary Research,* vol. 1, Berlin - New York, W. de Gruyter, 188-229.

Mel'čuk I., Pertsov N. (1987), *Surface Syntax of English. A Formal Model within the Meaning-Text Framework*, Benjamins, Amsterdam.

Milićević J. (2007) *La paraphrase — Modélisation de la paraphrase langagière*, Peter Lang.

Osborne T. (2005), Beyond the Constituent: A DG Analysis of Chains, *Folia Linguistica*, 39, 251-297.

Osborne T., Groß T. (2012), Constructions are catenae: Construction Grammar meets Dependency Grammar. *Cognitive Linguistics*, 23:1, 163-214.

Polguère A. (1990), *Structuration et mise en jeu procédurale d'un modèle linguistique déclaratif dans un cadre de génération de texte*, Thèse de doctorat, Université de Montréal.

Pollard C, Sag I. (1994), *Head-driven Phrase Structure Grammar*, Chicago University Press.

Sangati F. (2012), *Decomposing and Regenerating Syntactic Trees*, PhD thesis, University of Amsterdam.

Tesnière L. (1959), *Éléments de syntaxe structurale*, Kincksieck, Paris.