Interpreting and defining connections in dependency structures

Sylvain Kahane
Modyco, Université Paris Nanterre & CNRS
sylvain@kahane.fr

Abstract

This paper highlights the advantages of not interpreting connections in a dependency tree as combinations between words but of interpreting them more broadly as sets of combinations between catenae. One of the most important outcomes is the possibility of associating a connection structure to any set of combinations assuming some well-formedness properties and of providing a new way to define dependency trees and other kinds of dependency structures, which are not trees but "bubble graphs". The status of catenae of dependency trees as syntactic units is discussed.

1 Introduction

The objective of this article is twofold: first, to show that dependencies in a dependency tree or a similar structure should not generally be interpreted only as combinations between words; second, to show that a broader interpretation of connections has various advantages: It makes it easier to define the dependency structure and to better understand the link between different representations of the syntactic structure. We will focus on the possible syntactic combinations (what combines with what), without considering the fact that combinations are generally asymmetric, linking a governor to a dependent. We use the term *connection* to denote a dependency without the governor-dependency hierarchy.

Section 2 presents the notions of combination and connection, which are based on the notion of catena (Osborne et al. 2012). The properties of the set of catenae are studied and used to define an equivalence relation on the set of combinations, which is used to define the connections. Section 3 draws two consequences concerning the interpretation of phrase structure trees and the granularity of dependency structures. Section 4 reverses the process presented in Section 2 and shows how a set of units can be used to define a dependency structure, including structures which are not trees but what we call bubble graphs. We conclude by relating our interpretation of dependency structures to the cognitive process of parsing.

2 How to interpret a dependency tree

After presenting various views on combinations proposed in syntactic theories (section 2.1), we introduce the notion of catenae (Section 2.2), which allows us to introduce a new interpretation of connections (Section 2.3). A formal definition of connections is given in Section 2.4.

2.1 Various views on syntactic combinations

Let us start with an example.

(1) A photo of her room is hanging on the wall.

All syntactic theories agree on the fact that there is a subjectal construction, what Bloomfield (1933) calls an actor-action construction. But theories disagree on what units exactly are involved in this construction. For immediate constituency analysis and phrase structure grammars the combination is between the subject NP/DP *a photo of her room* and the VP *is hanging on the wall*. For dependency grammars, it is often considered that combinations are between words (see especially Mel'čuk (1988), who would consider a combination between *photo* and *is*).[1] Tesnière (1959) considered that connections were between nuclei, a nucleus generally combining a lexical word and function word.[2]

[1] Other dependency approaches make different choices: For Hudson (1984, 1987), who considers the determiner as the head of the "noun phrase", the combination is between *a* and *is*, while for the Universal Dependencies scheme, the combination is between the content words *photo* and *hanging*. We will see in Section 4 that it is possible not to choose between these different views.

[2] In the very beginning of his book, Tesnière (1959[2015]: ch. 1) says: "The sentence is an **organized set**, the constituent elements of which are the **words**. Each word in a sentence is not 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." But later in the same book (ch. 22), he revises this and introduces

For our example, the nuclei in question will be *a photo* and *is hanging*. This can also be compared with linguists who consider that combinations are between chunks (Abney 1991, Vergne 2000), following Frazier & Fodor (1978), who argue that utterances are processed in small chunks of about six words that are then connected to each other. Another view is to consider that the governor of a dependency is a word but that the dependent is a constituent.[3] Beauzée (1765) pointed out more than 250 years ago that it was necessary to consider as word complements both words (which he called initial/grammatical complements) and their projections (which he called total/logical complements): "For instance, in the sentence *with the care required in circumstances of this nature*; […] the preposition *of* is the initial *complement* of the appellative noun *circumstances*; and *of this nature* is its total *complement*; *circumstances* is the grammatical *complement* of the preposition *with*; and *circumstances of this nature* is its logical *complement*." (Beauzée 1765:5, cited by Kahane, forthcoming).[4]

As we will see all these views on syntactic combinations are compatible with dependency syntax.

2.2 Dependency trees and catenae

Dependency trees generally link words, which are considered as the minimal units. Figure 1 shows a surface-syntactic dependency tree (Mel'čuk 1988). We focus on the structure strictly speaking and do not introduce relation labels on dependencies.

(2) Mary looks at a photo of her room.

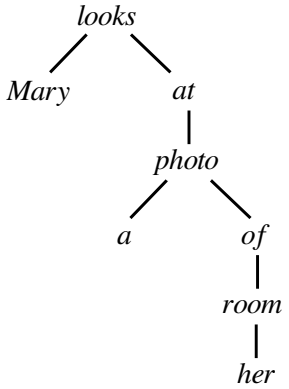

Figure 1. The dependency tree D0 of (2)

It has already been noted by various authors that dependency trees define a large number of units. Osborne et al. (2012) call *catenae* all units which are connected portions of a dependency tree and note that almost all of the units considered in syntax are catenae, starting with the constituents of phrase structure grammars.

The most interesting of these units are the (maximal) projections: To each node *x* of a dependency tree, we can associate its maximal projection, which is the unit formed by the nodes dominated by *x*, including *x* (Lecerf 1960, Beauzée 1765). For instance, the maximal projection of *photo* is *a photo of her room*.[5] An ordered dependency tree is said to be projective iff all its maximal projections are continuous.[6] As showed by Lecerf, an ordered dependency tree is projective iff its dependencies do not cross each other and no dependency covers the root.

Note that some catenae, such as *looks at photo of,* are not very relevant syntactic units. We will see how to avoid to consider such units in Section 4.

2.3 Connections and combinations

The narrow interpretation of a dependency tree is to consider that dependencies correspond to combinations of words, but other interpretations of a dependency can be made. For instance, a connection can be interpreted as a combination between the governor word and the projection of the dependent word (Beauzée 1765). The broadest interpretation that can be made of connections in a dependency tree is to consider that each dependency is potentially

---

the nucleus as "the set which joins together, in addition to the structural node itself, all the other elements for which the node is the structural support, starting with the semantic elements."

[3] The term *constituent* will be always used in reference to immediate constituent analysis and phrase structure grammars.

[4] Tesnière (1959[2015]: ch. 3) also considered that the dependent can be a projection: "We define a *node* as a set consisting of a governor and all of the subordinates that are directly or indirectly dependent on the governor and that the governor in a sense links together into a bundle."

[5] By removing branches formed by some of the dependents of *x*, we obtain partial projections of *x,* which are also catenae. The node *photo* has two partial connections: *a photo* and *photo of her room*.

[6] We call dependency tree only the tree structure. The dependency is generally combined with a linear order on its nodes. The projectivity is a property of the ordered dependency tree.

the combination of any catena containing the dependent node with any catena containing the governor node. For instance, the dependency between *at* and *photo* in our example can be interpreted as a combination between *at, looks at,* or *Mary looks at* on the one hand and *photo, a photo, a photo of her room* on the other hand (Figure 2). It is the latter interpretation that is the most valuable and that we will formalize now.

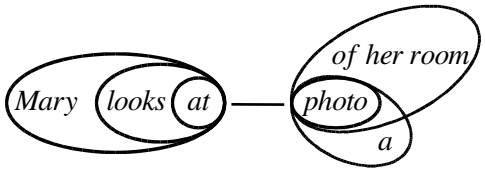

Figure 2. Broader interpretation of a connection

2.4 Formal definition of connections

Let us consider a sentence *s* which is segmented into a set X of (elementary) linguistic units (words, morphemes, etc.). A dependency tree D on X is a tree whose vertices are the elements of X.

The elements of X are linearly ordered by the fact that *s* is a string of elements of X. We call Unit(X) the set of all strings on X which are a subset of *s* (such strings may be a discontinuous subset of *s*). We call Catena(D) the subset of units in Unit(X) that are catenae on D, that is, connected portions of D.

Let us consider a set U of units. For the moment, we are interested in U = Catena(D), but our definitions will be applied to other sets of units in the following sections. A *combination* on U is any pair {A,B} of disjoint units ($A \cap B = \varnothing$) such that $A \cup B$ is a unit. For instance, { *looks at*, *a photo*} is a combination on Catena(D0).

The set of combinations on U is called Combi(U). A dependency tree D defines a set of combinations, which are all the combinations of adjacent catenae, that is Combi(Catena(D)).

We want to group the combinations of Combi(Catena(D)) that correspond to the same connection of the dependency tree D. Two such combinations are said to be *compatible*. The relation of compatibility is noted ≈. For instance, all the combinations illustrated by Figure 2 should be compatible.

For a dependency tree, the compatibility can be defined by different properties. We propose three of them.

$\{A,B\} \approx \{A',B'\}$ iff $\{A \cap A', B \cap B'\}$ is a combination [P1]

iff $\{A \cup A', B \cup B'\}$ is a combination [P2]

iff $A \cap A'$ and $B \cap B'$ are not empty and $A \cup A'$ and $B \cup B'$ are disjoint. [P3]

Consequently, if $\{A,B\} \approx \{A',B'\}$, then $\{A,B\} \approx \{A',B'\} \approx \{A \cap A', B \cap B'\} \approx \{A \cup A', B \cup B'\}$.

The relation of compatibility can be represented by the configuration in Figure 3.

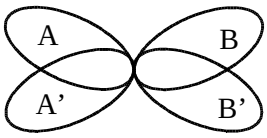

Figure 3. Two compatible combinations {A,B} and {A',B'}

The equivalence between our three characterizations of the compatibility ([P1], [P2], and [P3]) is not completely trivial. The equivalence is due to three more general properties of Catena(D) that we call the Intersection Property, the Acyclicity and the Sticking Property:

- Intersection Property: A set U of units has the Intersection Property iff for every A and B in U such that $A \cap B$ is non empty, then $A \cap B$ is in U;
- Acyclicity: A set U of units is acyclic iff for every A, B, and C in U such that $A \cap B$, $B \cap C$, and $C \cap A$ are non empty (i.e. A, B, and C form a potential cycle), then $A \cap B \cap C$ is non empty;
- Sticking Property: A set U of units has the Sticking Property iff for every A and B in U, if $A \cap B$ is in U, then $A \cup B$ is in U.

Let us show the equivalence.

- [P1] → [P3]: If $\{A \cap A', B \cap B'\}$ is a combination, then, by definition, $A \cap A'$ and $B \cap B'$ are not empty units. Due to the Acyclicity, $A' \cap B$ must be empty: if it was not, then A' , B, and $A \cup B'$ would form a cycle and their intersection, equal to $(A \cap A' \cap B) \cup (A' \cap B \cap B')$, would be non empty, which

is contradictory with the fact that A ∩ B and A' ∩ B' are empty. By symmetry, A ∩ B' must also be empty. We deduce that A ∪ A' is disjoint from B and B' and then from B ∪ B'.

- [P3] → [P2]: It is a direct consequence of the Sticking Property: A ∪ A' and B ∪ B' are units, because A ∩ A' and B ∩ B' are non empty, and A ∪ A' ∪ B ∪ B' is a unit, because A ∪ B and A' ∪ B' are non disjoint units.
- [P2] → [P1]: If {A ∪ A', B ∪ B'} is a combination, then A ∪ A' and B ∪ B' are disjoint. As A ∪ A', A ∪ B ∪ B', and A' ∪ B ∪ B' form a potential cycle, their intersection, which is A ∩ A', is non empty (Acyclicity). Similarly, B ∩ B' is non empty. Finally, (A ∩ A') ∪ (B ∩ B'), which is equal to (A ∪ A') ∩ (B ∪ B') in this particular case (because A ∩ B and A' ∩ B' are empty), is a unit (Intersection Property).

The relation ≈ is reflexive and symmetrical on Unit(X), but it is not transitive whenever X has more than 3 elements.[7] Nevertheless, the relation ≈ is transitive on Combi(Catena(D) and is then an equivalence relation. More generally, the transitivity of ≈ on Combi(U) is a consequence of the Acyclicity and the Sticking Property. Suppose that we have {A,B} ≈ {A',B'} and {A',B'} ≈ {A",B"}. This entails that A ∩ A" is non empty, because it is the intersection of A ∪ A' ∪ A", A ∪ B ∪ B' ∪ B", and A" ∪ B ∪ B' ∪ B", which form a potential cycle. Similarly, B ∩ B" is non empty and then {A,B} ≈ {A",B"}.

An equivalence relation on a set E induces a partition of E. The subsets of E forming this partition are called the equivalence classes on E. An equivalence class is a subset of equivalent elements. The set of equivalence classes on E for an equivalence relation R is called the quotient of E by R and is noted E/R. The elements of Combi(Catena(D)/≈ are the *connections*.

For those who are not familiar with quotient sets, it is certainly no doubt to draw a parallel with the set of rational numbers. We know that 1/2 or 2/4 or 50/100 represent the same rational number. In other words, the set of rational numbers is a quotient set obtained from the set of couples of integer numbers and the following equivalence relation: two couples (a,b) and (a',b') represent the same rational number if and only if ab' = a'b. Each couple (a,b), or fraction a/b to take the usual notation, is a representative of the relational number. Similarly, combinations are representatives of connections. For instance, { *at, a photo of her room* } and { *Mary looks at, a photo* } are two representatives of the same connection, represented in Figure 3.

Connections in a dependency tree have a minimal representative, which is a combination between two elementary units, that is, two elements of X. The minimal representative of the connection of Figure 3 is { *at, photo* }. By considering only the minimal combinations, we obtain a graph on X that we call the *connection structure* underlying the dependency tree (see Figure 4).

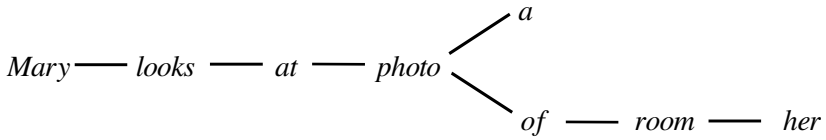

Figure 4. The connection structure associated to D0

The connection structure underlying a dependency tree D and the set of connections on D, Combi(Catena(D)/≈, are equivalent: one can be deduced from the other.

3 First consequences of our interpretation of dependency structures

3.1 Interpretation of phrase structure trees

Let us take a very simple sentence such as *Mary loves Peter*, whose dependency tree D1 contains a subject dependency from *loves* to *Mary* and an object dependency from *loves* to *Peter*. Catena(D1) = {*Mary, loves, Peter, Mary loves, loves Peter, Mary loves Peter*}. With our interpretation of connections we consider that the subject dependency indicates both a combination between *Mary* and *loves* and a combination between *Mary* and *loves Peter*. And symmetrically to this, we also consider that the object dependency indicates both a combination between *loves* and *Peter* and between *Mary loves* and *Peter*. Seen from the point of view of decomposition, this means that we consider that the phrase *Mary loves Peter* can be decomposed into both *Mary* + *loves Peter* and *Mary loves* + *Peter*. This fundamentally distinguishes dependency-based analyses from constituency-based analyses, which require that only one of

[7] Consider for instance a subject combination c0 = {*Mary, is sleeping*} and two incompatible refinements of this combination: c1 = {*Mary, is*} and c2 = {*Mary, sleeping*}. We have c1 ≈ c0 and c0 ≈ c2, but c1 ≉ c2, which shows the non-transitivity.

the two possible decompositions (the combination of the subject and the verb phrase) be retained.[8] Similarly, for a phrase such as *a photo of Mary*, our dependency analysis considers both the decomposition *a* + *photo of Mary* and the decomposition *a photo* + *of Mary*.

A binary-branching constituency tree also defines combinations and connections. Starting from the set U of constituents, we can define Combi(U): {A,B} is a combination if A and B are two constituents that combine to form a constituent. Combinations on a constituency tree are pairwise incompatible (for the relation ≈) and each connection has a unique representative. If we start with the same set X of elementary units, a dependency tree on X and a binary-branching constituency tree on X define the same number of connections. (If X has *n* elements, we need *n*–1 combinations to combine all the elements of X.) But, while a dependency tree propose many ways to combine the elements (with our interpretation of combinations in a dependency tree), a constituency tree only considers one way.

As a consequence, there are many constituency trees that can be derived from a given dependency tree and, conversely, there are many dependency trees which contain the set of combinations of a given constituency tree. More formally, a constituency tree C is said to be compatible with a dependency tree D if each combination defined by C is in Combi(Catenea(D)). As we just said, every constituency tree is compatible with several dependency trees and vice versa. Every constituency tree is obtained by choosing a representative in each connection of a compatible dependency tree.

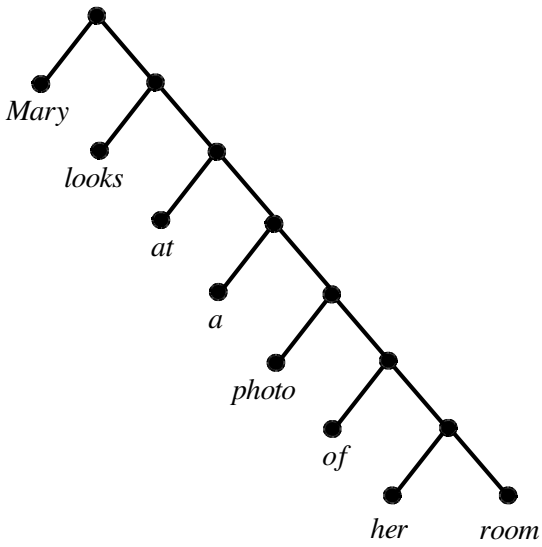

Figure 5. A binary-branching constituency tree C0 for (1)

Let us continue to see a constituency tree from the point of view of dependency. The question that arises is how a particular constituency tree is chosen from a given compatible dependency tree. Let us see how we obtain the constituency tree C0 of Figure 5 from D0. We start from the root *looks* and we choose one of the two dependencies starting from it: the subject dependency between *looks* and *Mary*. Now we choose the maximal representative of this connection {*Mary, looks at a photo of her room*}. We have now two units to analyze and we adopt the same strategy at each step: we consider each unit we have obtained, we take the root, we choose a dependency starting from the root and we choose the maximal representative of this connection inside the unit we are dealing with. In other words, all the constituency trees compatible with a given dependency tree D are obtained by ordering the dependents of every node and applying the same strategy that consists in taking the maximal representative at each step. For each order, we obtain a different binary-branching constituency tree. A dependency tree with such an order on the dependents has been called a stratified dependency tree by Kahane (1997) and is illustrated by Figure 6.

[8] Gerdes & Kahane (2013) give the following citation from Gleason (1955[1961]): "We may, as a first hypothesis, consider that each of [the words of the considered 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. […] We will also lay down the rule that each word can be marked as a member of only one such pair."

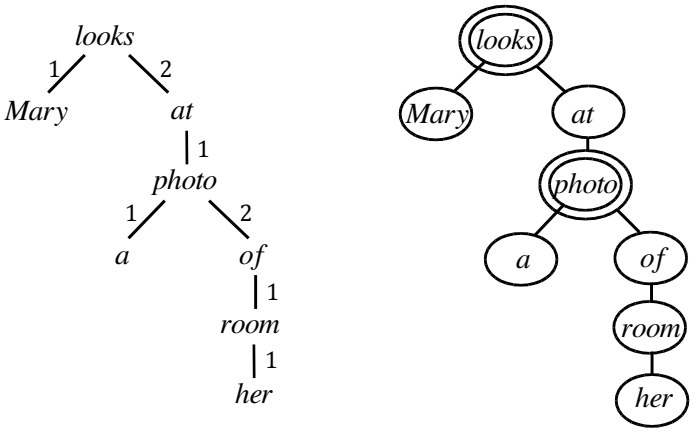

Figure 6. The stratified dependency tree D0+ giving C0
a. With order on dependencies. b. With the conventions of Kahane (1997)

Note that D0 and C0 are not equivalent: C0 induces stratification on D0 which is absent from D0 and conversely D0 induces headedness which is absent from C0. As stated by Kahane & Mazziotta (2015), a dependency tree is a connection structure plus headedness, while a constituency structure is a connection structure plus stratification. But the connection structure induced by a constituency tree is less fine-grained than the connection structure induced by a dependency structure, because it contains only large representatives and finer representatives cannot be deduced without adding additional information (such as headedness for instance).

From the theoretical point of view, the question that arises is whether or not the stratification is useful and need to be encoded in the syntactic structure. From the point of view of dependency syntax, it is an artifact of phrase structure grammars that has no real theoretical foundations.[9]

3.2 Granularity

We have just seen that our interpretation of connections in dependency trees was useful to compare dependency trees and constituency trees. It is also useful to compare dependency representations with one another.

Suppose that we have two structures S1 and S2 respectively defining the sets of combinations Combi1 and Combi 2. We say that S1 is finer than S2 if Combi1 $\supseteq$ Combi2. We have seen that dependency trees are finer than constituency trees in Section 3.1. (We will see in Section 4 that, from some points of view, traditional dependency trees can be considered as too fine.) Now consider a dependency tree D on a set of elementary unit X for a sentence *s* and take another segmentation Y of *s* less fine than X, which means that elements of Y are units on X. The dependency tree on X induces a dependency tree on Y. We can build it geometrically by collapsing connections in order to obtain only units in Y. It can also be defined algebraically by only considering combinations between units on Y, that is, Combi(Catena(D) $\cap$ Unit(Y)). The connections are still the equivalence classes for $\approx$. The new connections we obtain are subsets of the previous connections.

Changing the granularity of the analysis is quite useful. Traditional dependency trees consider words as the basic units, but some authors have considered dependency trees between morphemes (Gross 2011), while others consider that chunks (Abney 1991) are the units that combine together (Vergne 2000). Mel'čuk (1988) considers two levels of syntactic analysis: the Surface-Syntactic Structure (SSS) is a dependency tree between words (even if he considers that words are decomposed in lexemes plus inflectional morphemes), while the Deep-Syntactic Structure (DSS) is a dependency tree between (semantically full) lexical units which can be multi-word expressions. The DSS can be seen as a less granular structure than the SSS, which is interfaced with the semantic structure, while the SSS is interfaced with the phonological structure.

In Machine Translation, various granularities of the structure are involved. Current strategies based on translation memories consist in searching for the largest sub-units of a sentence for which a translation is available and combining their translations. This amounts to instantiating the connections in different ways, the units considered being generally not constituents.

[9] Scope phenomena are often considered as an argument for constituency (that is, stratification from the dependency point of view). But for instance, negation can have various scopes in a sentence such as *Mary did not give the book to Peter* and this is unrelated to the constituency structure.

3.3 Units and connections

The most important consequence of our interpretation of connections concerns the notion of connection itself. The same connection can be apprehended at different levels of granularity: between words, morphemes, chunks, constituents, etc. Whatever the level we consider, it remains more or less the same connection. As seen in Section 1, both dependency grammars and phrase structure grammars consider a subject connection for (1) even if they do not retain the same combinations as representatives. (Even inside dependency grammar, different representatives can be chosen, such as UD that favors combinations between content words, while many others favor functional heads (Osborne & Gerdes 2019).)

Hence, the connection strictly speaking is not subject to a particular level of granularity. The notion of connection is an abstraction on the notion of combination. Whereas the notion of combination is inseparable from the notion of unit, the notion of connection is not attached to a particular type of units.

From the theoretical point of view, it means that the definition of a dependency structure is not subject to a prior definition of the minimal units, and in particular to the controversial notion of word (Haspelmath 2011). As we will see now, we need to consider units to start the definition of the syntactic structure, but the units we consider at the outset are not necessary determining.

4 How to define the syntactic structure

In the first part of this paper, we started from a syntactic structure in order to see how to interpret it. We will now see how our interpretation of the structure can help us to define it.

4.1 From units to connections

We have seen that a syntactic structure such as a dependency tree defines a set of units we called catenae, following Osborne et al. (2012). We will now reverse the process and see how a set of units can define a syntactic structure. This idea was first developed by Gerdes & Kahane (2013). They propose to call fragment any part of an utterance that "is a linguistically acceptable phrase with the same semantic contribution as in the initial utterance". The fragments of a sentence are in some sense the syntactic units that are contained in the sentences, but, contrary to constituency-based analysis, it is not excluded that two syntactic units may overlap.

Gerdes and Kahane introduce the following example:

(3) *Peter wants to read the book*.

For (3), they consider the following set U of fragments: U = {*Peter, wants, to, read, the, book, Peter wants, wants to, to read, the book, Peter wants to, wants to read, read the book, Peter wants to read, to read the book, wants to read the book, Peter wants to read the book*}. We can remark that *read the book, read, the book, the,* and *book* are fragments, but not *read the* or *read book*, which are not acceptable phrases in English. Our purpose here is not to discuss the definition of fragments, but to see how the fragments can be used to define a structure.

Starting from U they build a structure they call the connection structure by first building all the binary-branching constituency trees whose constituents belong to U and then refine any of these trees by a geometric method. What we propose here is to directly build the connection structure.

Our first step is to define Combi(U), the set of combinations on U, as proposed in Section 2.3: A combination on U is any pair {A,B} of disjoint units in U ($A \cap B = \emptyset$) such that $A \cup B$ is still a unit in U.

The second step is to define the connections. Our set U verifies the three properties introduced in Section 2.3: Intersection Property, Acyclicity, and Sticking Property. Consequently, we can define the relation $\approx$ on Combi(U) by any of the three properties [P1], [P2], or [P3], and the relation of compatibility $\approx$ is an equivalence relation on Combi(U). The connections are the five equivalence classes of $\approx$ on Combi(U):

c1 = { {*Peter, wants*}, {*Peter, wants to*}, {*Peter, wants to read*}, {*Peter, wants to read the book*} }
c2 = { {*wants, to*}, {*wants, to read*}, {*wants, to read the book*}, {*Peter wants, to*}, {*Peter wants, to read*}, {*Peter wants, to read the book*} }
c3 = { {*to, read*}, {*to, read the book*}, {*wants to, read*}, {*wants to, read the book*}, {*Peter wants to, read*}, {*Peter wants to, read the book*} }
c4 = { {*read, the book*}, {*to read, the book*}, {*wants to read, the book*}, {*Peter wants to read, the book*} }
c5 = { {*the, book*} }.

We can now associate a connection structure to this set of connections . We will see that it does not correspond to the connection structure of a dependency tree

4.2 From connections to the connection structure

To build a connection structure, we choose in any connection the minimal representative. We obtain the five following combinations: {*Peter, wants*}, {*wants, to*}, {*to, read*}, {*read, the book*}, {*the, book*}. With these combinations we can build a structure we call a bubble graph, because some edges link non-elementary units which are represented by bubbles. (Cf. the notion of bubble tree in Kahane 1997.) It is given in Figure 6.

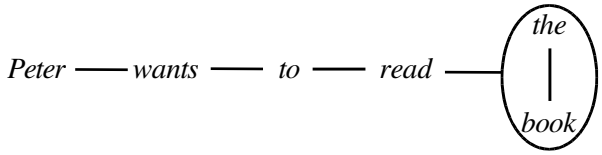

Figure 6. The connection structure associated to the set of fragments of (3)

We can obtain more complex bubble trees. Let us consider the French sentence (4).

(4) *Peter a parlé de Mary*
Peter has talked about Mary

Contrary to English, the subject in French cannot combine with the auxiliary alone (*Mary a* 'Mary has' is not an acceptable sub-phrase of the sentence) and the preposition cannot combine with the verb alone (*parlé de* 'talked about' is not an acceptable phrase). We therefore have the following set of fragments: U = { *Peter, a, parlé, de, Mary, a parlé, de Mary, Peter a parlé, parlé de Mary, a parlé de Mary, Peter a parlé de Mary* }. As before we can calculate Combi(U) and its partition by ≈. The minimal representatives of the connections are: {*Peter, a parlé*}, {*a, parlé*}, {*parlé, de Mary*}, {*de, Mary*}. We obtain the bubble tree of Figure 7.

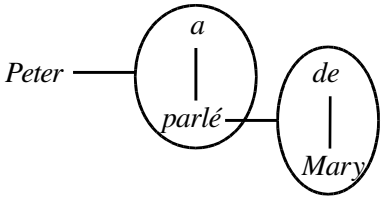

Figure 7. The connection structure associated to the set of fragments of (4)

The same methodology can apply if the elementary units we consider are morphemes. Consider the following sentence:

(5) *Peter reads two books.*

The words *reads* and *books* can be decomposed in *read-s* and *book-s*. The set of possible fragments is: U = { *Peter, read, -s, two, book, -s, read-s, book-s, Peter read-s, two book-s, read two book-s, read book-s, read-s book-s, read-s two book-s, Peter read-s book-s, Peter read-s two book-s* }. We obtain the bubble graph of Figure 8.

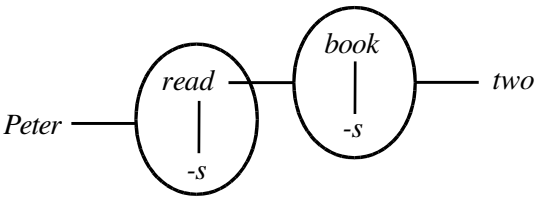

Figure 8. The connection structure associated to the morpheme-level fragments of (8)

4.3 From connection structures to catenae

It is interesting to remark that the set of fragments that gives us the connection structure can be recovered from the connection structure. It can be done by generalizing the notion of catena to bubble graphs. Let G be a bubble graph: Catena(G) contains all the units labeling the vertices of G, that is, all the elementary units as well as all the units corresponding to the content of a bubble. We add to Catena(G) all the units A ∪ B where A and B are the two vertices

of an edge in the bubble graph. And then Catena(G) is satured by sticking all units that intersect: if A and B are in Catena(G) and A ∩ B is non empty, then A ∪ B must be added to Catena(G). The sticking operation is iterated until it produces no new unit.

The catenae can also be built "geometrically" by cutting edges in the bubble graph. For instance, if we take our last example, the bubble graph of Figure 8, we can cut the edge between *read* and its inflection, as well as the edge between *Peter* and the bubble containing *read-s*. We obtain a subgraph corresponding to *read two book-s*, which is therefore a catena. We can cut the edge between *two* and *book-s* and obtain *read book-s*. It is possible to separate *read* and *book-s* by cutting the edge linking them, But if we maintain this edge, it is not possible to separate *book* and *-s*, even if we cut the edge linking them. Consequently, it is not possible to obtain units such as *read book* or *read -s (*where *-s* is the plural morpheme on *book*).

4.4 From connection structures to dependency structures

A dependency is a hierarchized connection linking a governor to a dependent. A dependency structure is then a bubble graph where all (or a part) of the edges are directed. This can be done by adding criteria, essentially distributional criteria to define a head in units (see Bloomfield, 1933; Hudson 1984; Mel'čuk, 1988). We will not develop this point. See Gerdes & Kahane (2013) for a description of this step applied to connection structures. As they remark, it is possible to refine some connections when adding new criteria (refining a connection means adding combinations to it). But it must be noted that if we do that, our set of catenae will no longer correspond to the syntactic units that were identified for building the connection structure, that is, the fragments. For instance, distributional criteria applied to the French sentence (4) will show that the subject is linked to the auxiliary, as in English, because it agrees with it and that the preposition is the head of the complement of the verb. In this way, we can obtain a dependency tree for (4), but this tree produces more catenae than the fragments considered for the building of the connection structure. These new catenae are *Peter a* or *parlé de*, as well as larger units. Contrary to the fragments considered at the beginning of the process, any of these units is very legitimate from the syntactic point of view. In other words, saying that A and B are syntactically linked does not necessarily mean that A ∪ B forms an acceptable syntactic unit (even if it is true in many cases). The same problem will arise with our morpheme-level analysis of (5): For instance, if we consider, following Gross (2011) (as well as most analyses in phrase structure grammars), that phrases have functional heads, we will obtain catenae such as *Peter -s* (where *-s* is the inflection morpheme of *reads*) and *read -s* (where *-s* is the plural morpheme on *books*), which are not acceptable units. This problem can lead us to consider richer syntactic structures where the different criteria used to define the structure are not merged and we keep some traces of the role of each of them. An attempt in this direction is proposed in Kahane & Mazziotta (2015).

5 Conclusion

From the mathematical point of view, we have shown that a connection structure can be built from a set of units verifying some properties: Acyclicity, Intersection Property, and Sticking Property. The structures we obtain are what we called bubble graphs. These structures reflect the properties of the set of units they come from and, in particular, they are acyclic and all their edges are binary. It could be interesting to generalize this approach to cases with ternary connections or with cycles.

From the linguistic point of view, we have shown that the connections considered by the dependency structures can be considered as sets of combinations. Most connections have a minimal representative which is a combination of two words, but some connections are not instantiated by a combination of words. First, combinations between morphemes inside a word are not combinations of words of course. Conversely, it is possible that some combinations have a minimal representative that involves units larger than words. For example, considering that, in a sentence such as *the dog slept,* the unit *the dog* has a well identified head and that we can refine the combination {*the dog, slept*} in a unique way is likely to be a bias of the analysis by dependency trees, which requires instantiating each connection by a combination between words or, which ultimately amounts to the same, choosing a head word for each unit. In fact, by their extreme nature, the two modes of representation, dependency and constituency, are biased. While the bias for dependency trees is systematic headedness, the bias for constituency trees is stratification: At each step of the immediate constituent analysis, it is necessary to decide which two units connect, even when there are several connections available. It is not possible with a constituency tree to simply say that, in *Mary loves Peter*, *loves* combines with *Mary* and *Peter*. We must stratify, i.e. treat the connections in a certain order (considering for example that *Mary* combines with *loves Peter,* which is itself the combination of *loves* and *Peter*). As a result, there are as many constituent trees associated with a given connection structure as there are ways to order the connections.

Lastly, a few concluding remarks concerning the cognitive point of view. Even if we think that dependency structures are the best way to encode the syntactic structure, we do not think that connections are instantiated between words. We postulate that, when a hearer analyzes a sentence, connections are instantiated by particular combinations

of particular units and that these instantiations may differ from one situation to another and do not correspond a priori to either words or constituents. We believe that prosody (including the silent prosody of the reader) plays an important role and that prosodic units are essential candidates for this instantiation (see Steedman (2014) for a similar approach in Categorial Grammars). As said in Section 2, Frazier & Fodor (1978) claim that main connections are instantiated by combinations of chunks of about six words. The fact that connections are not necessarily instantiated by their minimal representative can have interesting consequences from the NLP point of view: It means that parsing algorithms could take into account the fact that we are not trying to build a particular instantiation of the syntactic structure (a phrase structure tree or a dependency tree in the current state of systems), but to build any instantiation of the connections, even if it means refining them later.

We think that speakers manipulate units of different levels both when producing and analyzing statements. We have shown that our formalization of connections makes no assumptions about the nature of the units involved in the combinations instantiating the connections. We thus argue that we can study syntax without a priori asking the question of units, which is a very delicate question, since it is so difficult to define concepts such as words or sentences.[10] From this point of view, a definition that claims that syntax is the study of the organization of words within the sentence seems to us to have to be totally rejected. For us, syntax is above all the study of (free, regular) combinations between linguistic signs, without prejudging the level of granularity of these signs.

Acknowledgments

I would like to acknowledge Guy Perrier, as well as the three reviewers, for valuable comments.

References

Steven P. Abney. 1991. Parsing by chunks. In R. C. Berwick, S. P. Abney, and C. L. Tenny, *Principle-Based Parsing: Computation and Psycholinguistics*. Springer, Dordrecht, 257-278.

Nicolas Beauzée. 1765. Régime, in Denis Diderot and Jean Le Rond D'Alembert (eds.), *Encyclopédie*, vol. 14, 5-11. On line edition at enccre.academie-sciences.fr.

Leonard Bloomfield. 1933. *Language*, The University of Chicago Press.

Lyn Frazier, Janet D. Fodor. 1978. The sausage machine: A new two-stage parsing model. *Cognition*, 6(4), 291-325.

H. A. Gleason. 1955. *An Introduction to Descriptive Linguistics.* New York: Holt, Rinehart & Winston, 503 p. Revised edition 1961.

Kim Gerdes, Sylvain Kahane. 2013. Defining dependency (and constituency). In K. Gerdes, E. Hajičová, L. Wanner (eds.), *Computational Dependency Linguistics*, IOS Press.

Thomas Gross. 2011. Catenae in morphology. *Proceedings of the first in ternational conference on Dependency Linguistics (Depling)*, Barcelona, 47-57.

Martin Haspelmath. 2011. The indeterminacy of word segmentation and the nature of morphology and syntax. *Folia linguistica,* 45(1), 31-80.

Richard A. Hudson. 1984. *Word grammar*, Oxford: Blackwell.

Richard A. Hudson. 1987. Zwicky on heads. *Journal of linguistics*, *23*(1), 109-132.

Sylvain Kahane. 1997. Bubble trees and syntactic representations. *Proceedings of the 5th conference on Mathematics of Language (MoL)*, 70-76.

Sylvain Kahane. Forthcoming. How dependency syntax appear in the French Encyclopedia: from Buffier (1709) to Beauzée (1765). In A. Imrényi and N. Mazziotta, *History of dependency-based approaches to grammatical theory*, Benjamins.

Sylvain Kahane, Nicolas Mazziotta. 2015. Dependency-based analyses for function words – Introducing the polygraphic approach, *Proceedings of the 3rd international conference on Dependency Linguistics (Depling)*, Uppsala.

Yves Lecerf. 1960. Programme des conflits, modèle des conflits. *Bulletin bimestriel de l'ATALA,* 1(4), 11–18.

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

Timothy Osborne, Michael Putnam, and Thomas Groß. 2012. Catenae: Introducing a novel unit of syntactic analysis. *Syntax,* 15(4), 354-396.

[10] Gerdes & Kahane (2013: note 3) defends a similar position when they say: "It may seem paradoxical that we do not think that fragments are syntactic primitives. In some sense we agree with Tesnière when he says that "the mind perceives connections". A fragment is a witness to a possible connection and it allows us to postulate one connection or another."

Timothy Osborne, Kim Gerdes. 2019. The status of function words in dependency grammar: A critique of Universal Dependencies (UD). *Glossa: a journal of general linguistics,* 4(1).

Mark Steedman. 2014. The Surface Compositional Semantics of English Intonation. *Language*, 90, 2-57.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris, [transl. Timothy Osborne and Sylvain Kahane, *Elements of structural syntax*, Benjamins, 2015].

Jacques Vergne. 2000. *Étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur - Analyse syntaxique automatique non combinatoire*, Habilitation thesis, Université de Caen.