Syntactic Polygraphs
A Formalism Extending Both Constituency and Dependency

Sylvain Kahane
Université de Paris Ouest
Laboratoire Modyco (CNRS UMR 7114)
sylvain@kahane.fr

Nicolas Mazziotta
Universität Stuttgart
Institut für Linguistik/Romanistik
nicolas.mazziotta@ulg.ac.be

Abstract

Syntactic analyses describe *grouping operations* that explain how words are combined to form utterances. The nature of these operations depends on the approach. In a constituency-based approach, grouping operations are ordered, or *stratified*, part-whole relations. In a dependency-based approach, grouping operations identify a *governor* (or *head*), i.e. they are directed hierarchical relations between words. It is possible to convert a constituency tree into a dependency tree by dereifying the nodes, by identifying the governor and by removing the stratification of the part-whole relations. Polygraphs combine the two types of information into a single structure and are therefore a more powerful formalism. By relaxing constraints, polygraphs also allow to *underspecify* both kinds of information.

Keywords: syntactic structure, immediate constituents analysis, dependency tree, headedness, phrase structure tree, polygraph, reification, stratification, underspecification.

1 Introduction

Despite their differences, syntacticians agree on the fact that they study the way words combine to form utterances, either from a hierarchical point of view (abstract model) or focussing on word order.[1] Word order and distributional properties are beyond the scope of this study. The focus is on the means of modelling the hierarchy of words. Moreover, the aim of this paper is to explore the formal aspect of syntactic description rather than to propose grammatical rules or make predictions: the examples in this paper do not necessarily reflect the point of view of the authors as linguists, but are provided to illustrate that various formal points of view on syntactic combination can be encoded by the mathematical structures proposed here.

Dependency and constituency are often seen as alternative frameworks that describe how words combine to form utterances. For many linguists, choosing between these two alternatives is an important point that needs to be resolved early on[2], since dependency-based descriptions cannot be totally translated into a constituency-based counterpart and vice versa. By investigating the operations that allow this kind of conversion, an understanding of the differences between them as well as their similarities is promoted. The exploration of dependency and constituency provided here defines a new kind of linguistic structure: the *syntactic polygraph*. These structures encompass the expressive power of the other two formalisms.

In short, the aim of this paper is to compare immediate constituent analysis with dependency analysis and to provide a joint formalism that takes advantage of both approaches, making it possible to express more than either of the two formalisms would

[1] In this paper, *words* are considered to be minimal linguistic units. However, the argument is easily extendable to morphemes as in X-bar syntax since the introduction of IP instead of S; see also Groß (2011) a.o. for a syntactic analysis of morphemic combinations in dependency.

[2] However, phrase structure grammars have been the dominant paradigm in syntax since (Chomsky, 1957) and people working in PSGs generally do not discuss whether dependency could be an option. Most of the discussions on that subject have been conducted by syntacticians working in dependency syntax (Hudson, 1980; Mel'čuk, 1988; Osborne et al., 2011; Kahane, 2012).

allow. While a large part of the paper is devoted to presenting a new process of conversion between constituency and dependency, the conversion itself is not the main point. An extensive literature already exists on the conversion of syntactic treebanks from constituency-based formats to dependency-based formats and vice versa. The question in such studies is to see how to add missing information (for instance headedness) during conversion. Our concern here is to see how to encode information when you have it, how to make it explicit, structurally visible, and also how to avoid adding irrelevant information with formalisms that force you to express things you do not want to express.

Sec. 2 introduces the specific grouping operations of constituency and dependency from a linguistic perspective, with respect to the way they acknowledge combinations of words and gives a formal definition of syntactic trees. Sec. 3 evaluates the transformations that models undergo when converted from constituency to dependency and highlights the better expressiveness of polygraphs. Sec. 4 shows how to extend both formalisms, taking dependency trees as the starting point. The conclusion summarizes what is achieved in this paper (sec. 5).

2 How to group words: constituency and dependency

The minimal consensus between syntacticians is that words combine to form sets of words that follow some organizational rules allowing them to function together. Operations that describe these associations can be called *grouping operations*, for the sake of neutrality. Unlike words, which are observable units (see note 1), the *groups* that result from grouping operations are constructs that can also combine with other words or groups to build a hierarchy. Both constituency and dependency models acknowledge this, but the nature of the grouping operation, i.e. the constructed relation uniting the grouped elements, is somewhat different. This section gives a general linguistic definition of grouping operations in a constituency-based approach as well as in a dependency-based approach (sec. 2.1) and a general formal definition of the tree-like objects they use to represent groups (sec. 2.2).

2.1 Syntactic tree archetypes: linguistic definitions

This section describes how grouping operations differ in a constituency-based approach and in a dependency-based approach. To do this, it focuses on *strict* constituency and *strict* dependency, although many linguists actually blend these two ways of describing grouping operations.

Strict constituency: stratification of part-whole relations. This approach is grounded in Bloomfield's description of groups (Bloomfield uses the term *constructions*) in terms of *part-whole* relations (Bloomfield, 1933, § 10.2). Grouping operations result in complex units that *contain* lesser units called *immediate constituents*. The analysis thus consists of strata of constructions, each stratum being recursively divisible into smaller constructions until the lowest stratum is reached. This *stratification*[3] can be represented in various ways: bracketing, embedded boxes or trees. The latter representation (henceforth *constituency tree*) has become the most popular.[4]

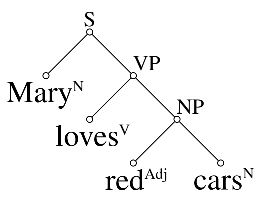
(1) Mary loves red cars

The constituency tree of (1) appears as fig. 1(a): internal nodes represent various groups that can be labelled and terminal nodes represent words (POS are marked as superscripts to words and will not be discussed here), but edges express only one kind of link between linguistic elements: the part-whole relation. This approach generates nodes in the tree, each of them corresponding to a stratum.

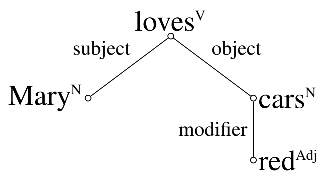
Strict dependency: directed relations between words. In a dependency-based approach, grouping operations are essentially directed connections between words (Tesnière, 1959; Tesnière, 2015, ch. 1). Such an approach implies that some words are more important than others with respect to their

[3]The term *stratification* has the same meaning as in our previous work (Kahane, 1997), where stratification is applied in a dependency-based representation.

[4]There were no tree in Bloomfield (1933), nor in the famous paper by Wells (1947). According to Coseriu (1980, 48), the first tree-like representation of constituency appears in Nida (1949, 87) (the "tree" may rather be a polygraph, cf. sec. 3.1) and current trees with labels on nodes became popular after Gleason (1955) and Chomsky (1957); embedded boxes are used in Hockett (1958).



(a) Constituency tree



(b) Dependency tree

Figure 1: Constituency and dependency trees

syntactic position. Therefore, the proposed hierarchy is not based on constructed part-whole relations, but on constructed *dependency* relations that associates pairs of *governor* and *dependent* words – see Mel'čuk (1988) for this terminology. Unlike part-whole relations, dependencies are associated with specific labels indicating the kind of grouping operation. The most common way to represent a set of dependencies is a tree-like structure (henceforth *dependency tree*).[5] The dependency tree of (1) appears as fig. 1(b), with traditional labels on top-down lines between governors (top) and dependents (bottom).

Hybrid models. Linguists often combine part-whole relations and governor-dependent dependencies. An extended review of all hybrid models is not needed: one example will suffice. X-bar trees analyse all phrases as a combination of a governor X (the *head*) and two dependents that aggregate with the head at their distinct strata (the *complement* at the $\bar{X}$ level and the *specifier* at the SX level). Although many scholars refer to this kind of modelling by using the word *constituency*, it should be clear that does not consist of *strict constituency*: it combines the two kinds of grouping operations that have just been introduced. This hybrid approach is based on Bloomfield's description of endocentric constructions (Bloomfield, 1933, § 12.10). The hybrid model introduced in this paper is compared with such headed constituency trees in sec. 3.2. Note that even if the headedness can be added in a constituency tree, it can only be introduced in the labelling and it will not be part of the topology of the structure as it would be in a classical dependency tree.

[5]Our dependency tree corresponds to the so-called surface-syntax structure in the Meaning-Text Theory (Mel'čuk, 1988). Other levels of representation in the same theory and other theories – e.g.: Word Grammar (Hudson, 2010) – may use graphs where a node can have several governors.

2.2 Syntactic trees as graphs: formal definitions

Intuitively it is obvious that dependency and constituency trees can be described as tree objects, i.e. constrained forms of graphs. The following analysis is grounded on the basic hypergraph structure (Bergé, 1973). A *hypergraph* on a set X is a set E of subparts of X. The elements of X are called the *nodes* and the elements of E are the *edges* of the hypergraph. An edge e is a subset of X whose elements are called its *vertices* (e.g. $e = \{x, y, z\}$, where $x, y, z \in X$). A *graph* is a hypergraph whose edges are pairs, i.e. edges have two vertices (e.g. $e = \{x, y\}$).

Since the distinction becomes important further on, it should be noted that nodes and vertices are different concepts, the latter being slots that the former occupy. In other words, a node can be the vertex of several edges. Vertices of a hypergraph can also be assigned a *type* that is bound to their association with an edge. An edge of a *typed hypergraph* is a subset of $X \times T$, where T is the set of types. A *directed graph* is a particular case of typed graph where each vertex is associated to a type carrying appropriate semantics: $T = \{source, target\}$; e.g. $e = \{(x, source), (y, target)\}$ with x and y in X, usually abbreviated in (x, y) and represented by an arrow from x to y ($x \rightarrow y$) or, as in a tree, by a vertical line with the source on top.

A *tree* is a connected directed graph such that every node but one, called the *root*, is the target of one and only one edge. In the graphical representation of such trees, the directions of the edges are expressed by a convention: the source is higher on the figure than its targets. Moreover, constituency and dependency trees are labelled trees, i.e. nodes and edges can be labelled.

3 Constituency and dependency dialectics

This section describes what formal operations must be applied to strict constituency trees (fig. 1(a)) in order to obtain a strict dependency tree (fig. 1(b)) and vice versa. Although it is possible to apply these operations to any strict constituency tree, the focus is on binary-branching trees and abstracts away from other constituency structures.[6] It is possible to convert a binary-branching strict constituency tree into a strict dependency tree in five steps:[7] converting the branching nodes into edges (sec. 3.1), specifying the direction of the edges (sec. 3.2), delinearizing the graph (sec. 3.3), selecting more specific vertices (sec. 3.4) and relabelling the edges (sec. 3.5). Some of these formal transformations alter the amount of information, either by addition or by substraction. Additionally, the graphical means used to express the syntactic structure achieves a variable level of semiotic coherence and readability.

There exists another way to convert a constituency tree into a dependency tree presented by Lecerf (1961), consisting of collapsing all the constituents with their lexical head, i.e. , in an X-bar approach, turning any specifier, complement or adjunct into a dependent of this head. This classical conversion gives the same result as the one presented here for headed binary-branching constituents, but it can only be applied to headed constituency trees. The conversion presented here can be applied to headed constituents as well as non-headed ones, preserving the non-headedness (sec. 4.2). Additionally, it shows that constituents and dependencies encode the same groups of words and that the dependency itself can be viewed as a grouping operation. The two conversion processes also differ for non-binary branching trees, giving interesting insight into the different uses of ternary branchings in constituency trees (sec. 4.3).

3.1 Node/edge conversions

From a purely formal point of view, graphs can undergo transformations that convert edges into nodes or nodes into edges, namely *reification* and *dereification*. As pointed out in sec. 2.1, constituency trees contain nodes that result from the description process. These nodes can be translated into edges by *dereification* without changing the structure of the analysis. To understand dereification, one has to be familiar with reification.

Reification. For the sake of clarity, reification will be illustrated by conversions on the semantic representation of (2). The focus will come back to syntactic trees when dereification is applied to them.

(2) Mary loves Peter

Modelling verbs as predicates is very common. In the semantic structure of (2), 'love' (the meaning of the lexeme LOVE) is a binary predicate taking 'Mary' and 'Peter' as arguments: 'love'('Mary', 'Peter'). The first argument, 'Mary', can be called the agent, and the second argument, 'Peter', the patient. This semantic representation can be encoded by different graphs according to different conventions that are provided here for illustration only (fig. 2): in (a), 'loves' is associated to an edge directed from the agent to the patient (represented as nodes); in (b), the meaning 'love' is a node, bound to two other nodes by edges that make semantic roles explicit through their label (*agent* and *patient*); in (c), *agent* and *patient* are represented as nodes and their relations to their sources and targets have been made explicit.[8]

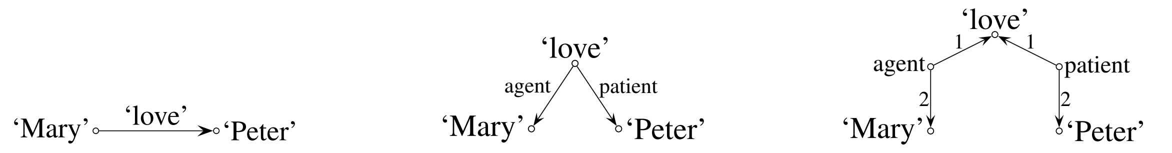
The *reification* of an edge e is the conversion e into a node and creating a directed edge (e,x) for each vertex x of e. If the graph is typed, the edge representing a vertex relation typed a will be labeled by a. For instance the reification of a directed edge e gives a node e and a *source* and a *target* edge. The edge of graph (a) is reified in graph (b). The edges of graph (b) are reified in graph (c), where the numbers are simple indices that distinguish the relations.

Dereification. Dereification is the reverse operation. It can be applied to any oriented graph. Formally, a node n_0 that is the source of two edges

[6]A tree is *binary-branching* if each node is the source of 0 or 2 edges.

[7]Provided that dereification is the first step, the conversion can perform the other steps in any order.

[8]Representations like (b) are structurally similar to the semantic graphs of Meaning-Text Theory – however, Mel'čuk (1988) merely numbers the edges; representation (c) is similar to Sowa's conceptual graphs (Sowa, 2000) – the labels with numbers conventionally identify arguments of the predicate structure.



(a) *'love'* labels an edge *(b)* *'love'* labels a node *(c)* *agent* and *patient* label nodes

Figure 2: Various structures corresponding to 'love'('Mary', 'Peter')

$e_1 = (n_0, n_1)$ and $e_2 = (n_0, n_2)$ will become an edge $e_0 = \{n_1, n_2\}$. If e_1 and e_2 have the labels a and b, e_0 becomes a typed edge $\{(n_1, a), (n_2, b)\}$.

One can dereify all non-terminal nodes of the constituency tree. The terminal nodes (expressing words) remain nodes, whereas the nodes corresponding to groups become edges. The constituency tree in fig. 1(a) can be dereified since a tree is a directed graph (see fig. 3(a), where the orientation of the part-whole relation is explicit). By dereifying the NP representation (*red cars*), the NP node and the two part-whole edges of which it is the source become a single edge (fig. 3(b)) that represent the grouping operation rather than its result. This NP edge remains undirected because the immediate constituents have the same hierarchical status in a strict constituency approach (both are the target of part-whole directed edges sharing the same source). Applying dereification to all nodes expressing constructions results in the structure in fig. 3(c). The starting constituency tree is binary-branching, hence the edges obtained are binary. However, the result is not exactly a tree: vertices can be edges as well as nodes. Henceforth, this type of structure will be called a *polygraph*.[9] In a polygraph, an edge e_1 can have another edge e_2 as vertex. In other words, if a polygraph has a set X of nodes and a set E of edges, each edge e in E is associated with a set of vertices in $X \cup E$; e.g.: $e_1 = \{x, e_2\}$, where $e_2 = \{y, z\}$ with $x, y, z \in X$.

From a formal point of view, the resulting polygraph expresses exactly the same stratification as the constituency tree, and will be called a *constituency polygraph*. However, it achieves two straightforward semiotic correspondences:

- Nodes always correspond to words (i.e. "observable" units, see note 1).
- Edges always correspond to grouping operations/constructions (i.e. linguistic analysis).

Since it shares these characteristics with dependency trees, the constituency polygraph can be considered as a better starting point for the conversion that will be performed.

3.2 Specification of the governors: directing the edges

The constituency polygraph does not represent the governor-dependent relations of the dependency approach. To be able to do so, the polygraph must be directed, by typing the vertices of each edge with either the type *source* or the type *target* (see sec. 2.2).[10] Graphically, lines are turned into arrows to express the direction of the edges according to the target dependency tree (fig. 1(a)). This operation results in fig. 4(a) – which will be compared to fig. 4(b) below. In fig. 4(a), the nodes of the polygraph are also linearly ordered.

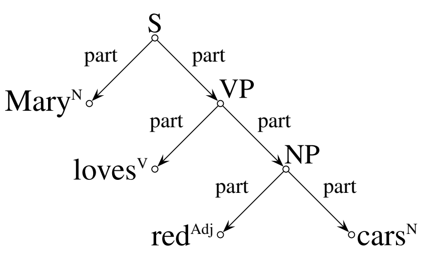
Contrary to the dereification process, direction specification actually adds information to the structure:

- Edges still express grouping operations and the stratification of the constituency tree.
- In addition, edges now express governor-dependent relations.

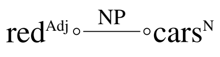
The directed polygraph expresses the same contents as a *headed constituency tree* (sec. 2.1). Such a tree

[9]The term *polygraph* is used in category theory for a family of objects that are a particular case of the structures called polygraphs here (Burroni, 1993; Bonfante and Guiraud, 2008).

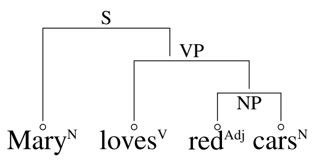
[10]Or *governor/dependent*, *head/non-head*, *H/NH*, etc. The types need to be interpreted to express the direction of the edges. It should be noted that even in a directed graph defined only by tuples (and not unordered sets), associating the direction of the edges with linguistic dependency is also a reading convention.



(a) Direction in the constituency tree

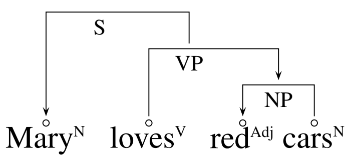


(b) Dereified NP

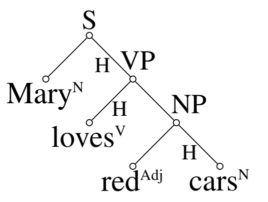


(c) Constituency polygraph

Figure 3: Dereification of constituency trees



(a) Ordered directed polygraph



(b) Headed constituency tree

Figure 4: Directed polygraph and headed constituency tree

combines constituency with dependency by specifying the governor (or *head*) at each level of aggregation. In fig. 4(b), each governor is identified by the label *H* on the part-whole relation linking it to the source construction.[11] When the dereification is applied to a constituent C with two immediate constituents A and B, A being the head, C becomes the edge $\{(A,H),(B,NH)\}$. If H and NH are interpreted as, namely, *source* and *target* typing, the structure becomes a directed polygraph (fig. 4(a)).

Again, the comparison of the two structures with respect to the information they carry must be supplemented with the evaluation of their relative semiotic efficiency. The directed polygraph is more coherent (because of the aforementioned correspondences *nodes/observed units* and *edges/constructed analyses*). It is also more readable:

- All directed edges have exactly the same interpretation (direction and label).
- All directed edges express relations of the same kind, whereas the edges in the headed constituent tree express both part-whole relations and headedness.

3.3 Graphical delinearization

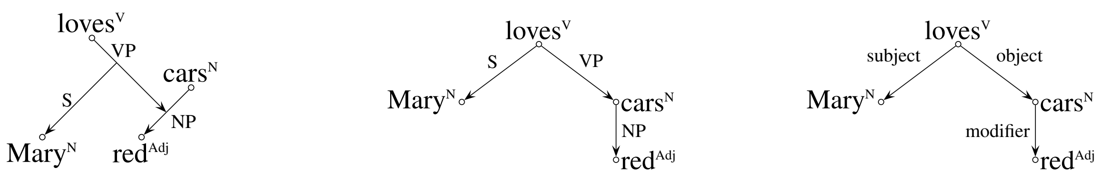
The structures in fig. 4(a) actually combine two structures: the proper polygraph, which expresses the hierarchical order – i.e. the structural order of Tesnière (1959) – and the linear order, which is represented by the position of the nodes on the same horizontal line. Since the focus is on the comparison between strict constituency and dependency, the disposition of the nodes is not relevant: changing it will not modify the hierarchical structure. The directed polygraph in fig. 4 can be drawn as in fig. 5(a) by abstracting away from word order and placing the source of directed edge above the target.[12]

3.4 Underspecification of the grouping orders: node selection

The vertices of an edge of a polygraph can be nodes or edges. To convert this structure into a tree that has the same shape as the target dependency tree in fig. 1(b), one simply has to select nodes as vertices for each edge. Nodes will be selected using the direction. A node, called the *top node*, is associated

[11]The conceptual means to define the governor and its graphical expression may vary, e.g., $\bar{X}$ and X labels in phrase structure trees (Jackendoff, 1977) or labelled edges such as H and NH in HPSG (Pollard and Sag, 1994).

[12]Such representations were first proposed by Tesnière (1959, ch. 65 and ch. 108) for encoding scope relations in its dependency-based representation.



(a) Directed constituency polygraph *(b)* Destratified constituency polygraph *(c)* Dependency tree

Figure 5: From directed polygraph to dependency tree

to each edge recursively: if $e = (v_1, v_2)$, $\mathrm{top}(e) = \mathrm{top}(e_1)$ and if n is a node, $\mathrm{top}(n) = n$.[13] To each directed polygraph, a graph is associated by replacing any edge (v_1, v_2) by the edge $(\mathrm{top}(v_1), \mathrm{top}(v_2))$.

For instance, in the polygraph in fig. 5(a), $\mathrm{top}(S) = \mathrm{top}(VP) = loves$ and $\mathrm{top}(NP) = cars$. The corresponding graph after selection is then the tree in fig. 5(b). From the geometrical point of view, the selection can be viewed as a down-shifting of edges along other edges.

Selection corresponds to deleting the order of grouping operations in the constituency tree. The operation substracts information from the original constituency tree: namely, its stratification. The polygraph expresses the stratification of groups and each instance of selection underspecifies this order for a couple of grouping operations.

Geometrically, the edge *VP* can be slid along the edge *S* (and *loves* becomes the source of *S*) and the edge *NP* can be slid along *VP* (and *cars* becomes the target of *VP*), as seen in the tree in fig. 5(b).

3.5 Relabelling

By relabelling the edges of the polygraph obtained after selection with names of grammatical functions, one generates fig. 5(c), which is the dependency tree of fig. 1(b) plus arrows that express the directions of the governor/dependent relations. We can see that the grouping operation named *S* in phrase structure grammar (Chomsky, 1957) is the grouping operation that dependency grammar names the *subject* relation (Mel'čuk, 1988). In works preceding Chomsky's formalization (Chomsky, 1957), groupings are more clearly interpreted as constructions. Bloomfield (1933, §12, 2) calls this grouping *actor-action construction* – Bloomfield's *action* does not correspond exactly to a VP constituent, since the *direct object* relation (like *watch me*) is called the *action-goal construction* in §12.8. Even Chomsky (1957, ch. 4) introduces the constituency tree as a derivation tree. Following Vijay-Shanker (1992), one can extend the notion of derivation tree to TAG, where each node must be interpreted as a rule. In other words, the S node of the derivation tree can be interpreted as $\alpha_S = [S \to NP + VP]$ and the relation between S and VP as the substitution of a rule α_{VP} into a rule α_S.

Constituency and dependency are two ways to encode grouping operations on the same words. Constituency focuses on the stratification, that is the respective order of the grouping operations (a verb groups with its object before grouping with its subject), while dependency focuses on headedness, that is the governor-dependent relation inside each group (a verb governs its subject and its object). Consequently, when a word is defined as a governor in the latter approach, it is aggregated with each of its dependents in several groups at the same level. Constituency generally acknowledges the same associations, but it has the means to stratify it through the definition of smaller nested groups.

The conversion is complete and reverting the process would allow one to go from the dependency tree to the constituency tree.

3.6 Intermediate conclusion

The five steps can be summarized with respect to the information carried by each strict formalism (the most important steps[14] appear in fig. 6). Polygraphs, which have been used as a temporary step in the process can be regarded as an independent modelling

[13]The top node of an edge is nothing else than the lexical head of the corresponding constituent. This step, the replacement of a constituent by its lexical head, is equivalent to the step in Lecerf's procedure collapsing all the nodes with their common lexical head.

[14]Delinearization and relabelling are not represented.

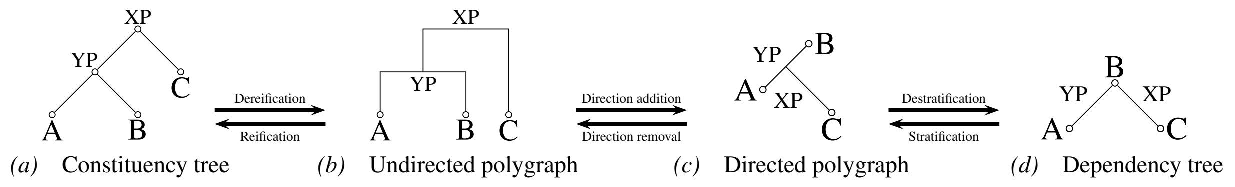


Figure 6: Synthesis of constituency tree/dependency tree conversion steps

tool that achieves better expressiveness.

Descriptive powers compared *Governor specification* adds dependency information (sec. 3.2), that is *headedness*, to the polygraph and *selection* deletes the order of grouping operations (sec. 3.4), that is *stratification*, from the polygraph. What this means is that neither strict constituency trees nor strict dependency trees can really be considered to be a more powerful formalism, because neither of them can express everything that can be expressed by the other. A formalism A is *more powerful* than B if there exists a one-to-one map from A to B preserving the topology of the structures in A. In other words, every structure *a* written with the formalism A can be converted into a structure *b* written in the formalism B without losing any information, that is, *a* can be recovered from *b*.

Constituency is able to acknowledge stratification; e.g.: in fig. 1(a), *red cars* aggregates with *loves* before *Mary* aggregates with the rest of the sentence. Choosing between one order or the other is an important methodological issue that should not be overlooked (Gleason, 1955; Wells, 1947). The drawback of constituency is that part-whole relations do not express any hierarchy between immediate constituents of a single construction (*Mary* and *loves red cars* share the same level). On the other hand, dependency provides an explicit way to describe the relative importance of individual words in comparison with the other words that surround them. Therefore, identifying the head of a group is a major concern (Mel'čuk, 2009, 27-34) and is even *compulsory* in a strict dependency approach.

Polygraph as a synthesis Polygraphs can be considered a gateway for the conversion of one type of structure into the other. They can contain both the stratificational information of constituency trees and the dependency structure. They can potentially express everything that both strict approaches can. Directed polygraphs can be drawn as in fig. 5(a), a representation close to a dependency tree, in order to assert the hierarchical order. But an ordered polygraph can also be represented with the nodes linearly ordered (fig. 4(a)) and looks more like a constituency tree.

Seen as an intermediate stage in a conversion process, polygraphs offer no free option: from strict constituency to strict dependency, the direction of all edges *must* be specified and information about the order of grouping operations *is necessarily* lost. Nevertheless, polygraphs can be considered as an *independent modelling tool* that allows one to choose whether or not specific governors and the order of grouping operations must be specified. They therefore allow *underspecification* in addition to combining the descriptive power of concurrent formalisms.

- A dependency tree is a directed polygraph that is completely destratified.
- A constituency tree is a stratified polygraph that specifies no governor.

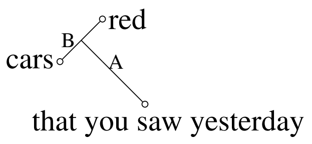
As an independent modelling tool, polygraphs achieve better expressiveness than strict constituency and strict dependency alone, for they satisfy the following requirements:

Descriptive power Polygraphs can structurally express everything that constituency and dependency can express (grouping, headedness, stratification).

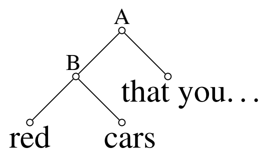
Underspecification Polygraphs can underspecify what is considered as non necessary (headedness, stratification)

The latter property is as important as the former: a formalism that forces one to specify irrelevant information can be as problematic as a formalism that does not allow one to specify relevant information.[15]

[15]Derivation trees provide an example of underspecification



(a) Original polygraph



(b) Reified version

Figure 7: Stratification of dependency trees

4 Extending dependency trees

As demonstrated in sec. 3, it is possible with polygraphs to encode both dependency and constituency, i.e. *grouping*, *headedness* and *stratification*. The aim of this section is to show what can be achieved by using polygraphs as an independent formalism rather than as a gateway for converting constituency trees into dependency trees. In a dependency-oriented perspective, it shows in which ways and for what purpose polygraphs can extend the formalism of dependency trees.

4.1 Stratification of dependency trees

Stratification can be acknowledged by polygraphs. It is not clear what advantage can be obtained by systematically forcing the verb to combine with its subject after its object as is done in PSGs.[16] DGs are attached to the verb centrality, defended at least since Tesnière (1959, ch. 49), who fought strongly against the bi-partition of the sentence into subject and predicate, i.e. NP and VP. Tesnière's point of view could be reformulated by saying that stratification of the verbal subcategorization in current constituency-based approaches is an artefact of the encoding of the syntactic structure by a binary-branching constituency tree (Kahane and Osborne, 2015, xxxix-xlii).

In other words, the fact that dependency trees make it possible to formalize binary groupings without needing stratification is viewed as a strong advantage by DGs.

This preliminary remark does not mean that stratification cannot be useful sometimes. Consider the following example:

(3) red cars that you saw yesterday

As already noticed by Coseriu (1980, 55), Tesnière (1959, ch. 65) proposed to represent the syntactic structure of (3) with a polygraph-like stemma (redrawn here in fig. 7(a)) (Tesnière, 1959, stemma 149) and provided the following justification: "By this process, the phrase *red cars that you saw yesterday* can be analyzed structurally in such a way that the connecting line extending upward from the subordinate clause reaches the connection line connecting *red* to *cars*. This means that *that you saw yesterday* is connected not to *cars* but to *red cars*, since what you saw yesterday was not cars, but red cars." Indeed, this representation says that *cars* combines first with *red* and after with the relative clause, as shown also by the constituency tree in fig. 7(b) which is obtained by reification of the relations in polygraph in fig. 7(a).

4.2 Headless grouping

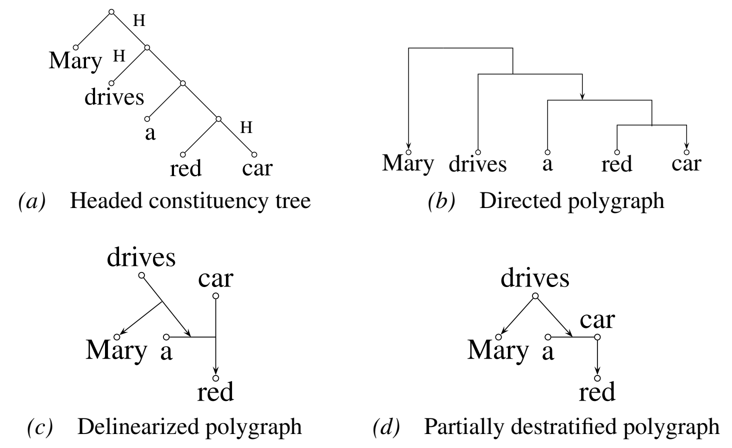
The lively debate about the DP vs. NP analysis of the determiner-noun construction – the still open discussion started 30 years ago (Hudson, 1984; Abney, 1987) – could be resolved by admitting that both the determiner and the noun have head features. Neither of them should be favored. This leads to propose a syntactic structure such as fig. 8(d) for sentence (4):

(4) Mary drives a red car.

This polygraph can be obtained starting from the constituency tree in fig. 8(a). Dereification outputs the ordered polygraph of fig. 8(b) where only a

that every formal linguist knows about. Context-free grammars were defined by Chomsky (1957) as rewriting systems. A *derivation* in a CFG is a string of rewriting steps. The *derivation tree* is a better formalism of representation of the derivation process, because it masks the order in which irrelevant steps occur in a derivation and "keeps only what is essential to understand the phrase structure" (ibid.: ch. 4). That is why the derivation tree is much more adequate than the proper derivation for a syntactic representation of the structure of a sentence.

[16]As far as incremental parsing is concerned, which is cognitively very motivated, it seems clear that the verb will combine with its subject before its object in SVO languages.



(a) Headed constituency tree *(b)* Directed polygraph *(c)* Delinearized polygraph *(d)* Partially destratified polygraph

Figure 8: Underspecified determiner-noun headedness

headed grouping outputs a directed dependency. Delinearization outputs the representation of fig. 8(c), where directed edges are represented by vertical lines with the head on top, while the non-directed edges are represented by horizontal lines. Selection results in the structure depicted in fig. 8(d). Non-directed edges cannot undergo selection and remain governed as a whole.

From the geometrical point of view, it means that non-directed edges cannot undergo any selection, which justifies representing them horizontally. Reed and Kellogg (1877) already use horizontal convention in their famous diagram, where the subject-verb and the object-verb combinations are non hierarchically organized. A linguist wishing to express a "headless" (or exocentric) structure, such as the former Chomskyan scheme $S \rightarrow NP + VP$, simply cannot do so with a dependency tree. The transformation exposed in sec. 3 applied to such a constituency tree (fig. 9(a)) produces a polygraph very similar to the Reed-Kellogg diagram (fig. 9(b)).[17]

(5) I think Mary loves cars.

The horizontal convention was also used by Tesnière (1959, part II) for coordination (see sec. 4.3).

4.3 Ternary grouping

In a dependency tree, every grouping is a binary grouping between a head word and a non-head word, encoded by a dependency. Some linguists consider that function words are just markers of constructions. For instance, *to* in (6) could be analyzed not as the head of a PP but rather as the marker of the combination between *talked* and its indirect object.[18]

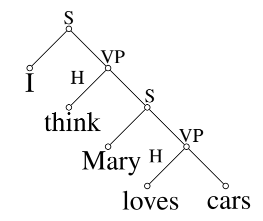
(6) Mary talked to Peter.

This results in the constituency tree in fig. 10(a), with a ternary branching where *talked* is the head (*H*), *to* the marker (*M*), and *Peter* a non-head word. Dereification and selection result in the structure in fig. 10(b), i.e. a *hyperpolygraph*. A *hyperpolygraph* is to a polygraph what a hypergraph is to a graph. In this case, hyperpolygraph contains a ternary "dependency" with three vertices: a governor, a dependent, and a third vertex put on the edge, occupied by the marker *to*, which must not be interpreted as a label. Such a convention already appears in dependency-based representations, even in some very early attempts (Kern, 1883, 17) – see also Débili (1982) for a more recent example. The representation in fig. 10(c) is also valid: the marker *to* depends on the relation it marks (Kahane and Mazziotta, submitted Depling 2015), indicating that the marker cannot occur without the relation. Such a polygraph cannot be reified into a (constituency) tree.

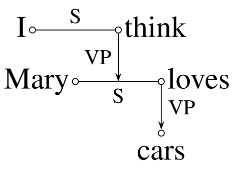
Another example combines ternary grouping with non-directed grouping. In symmetrical analyses of

[17]Note that Nida (1966, 17-47) consists of dozens of similar diagrams, that contrast a.o. endocentric structures and exocentric ones.

[18]A ternary grouping can be justified because the three elements can be grouped pairwise following different criteria: the verb and the object are linked by a semantic relation, the marker can be associated with the verb (*the guy Mary talked to*) as well as the object (*To Peter, Mary should not talk*).

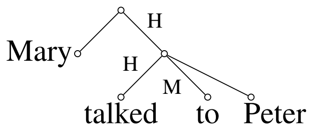


(a) Partially headed constituency tree

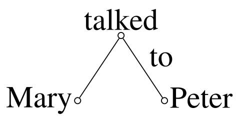


(b) Partially directed polygraph

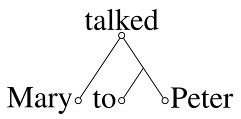
Figure 9: Underspecified subject-verb headedness



(a) Constituency tree



(b) Hyperpolygraph with marker node



(c) Polygraph with marking relation

Figure 10: Ternary grouping

coordination, conjuncts are considered as co-heads, while the coordinating conjunction is a marker, as proposed by Jackendoff (1977).

(7) Mary and Peter got married.

Fig. 11 shows a constituency tree for (7) and the corresponding polygraph after dereification and sliding. A horizontal line is again used to encode a grouping between two co-heads. The resulting representation is similar to what Tesnière proposes from his early works (Tesnière, 1934) to his main book (Tesnière, 1959, ch. 38).

5 Conclusion

The goal of this paper was to present a new formalism that subsumes both constituency and dependency, and which can be then used to extend these two formalisms in order to encode syntactic analyses.

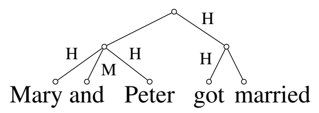
Two goals have been achieved. First, a new way to associate a dependency tree to a headed binary-branching constituency tree has been proposed. Contrary to the previous one from Lecerf (1961), this one proves that every dependency corresponds to a grouping.

Second, the formalism of polygraphs extends both formalisms, constituency trees as well as dependency trees, making stratification as well as headedness explicit. Interestingly these structures are similar to representations that have been previously proposed by other linguists (Reed and Kellogg, 1877; Kern, 1883; Nida, 1949; Nida, 1966; Tesnière, 1959; Débili, 1982) and polygraphs give them a mathematical foundation – see Mazziotta (2014) for a more comprehensive application of polygraphs to Tesnière's model. Beyond that, the formalism of polygraphs can be the basis for the development of new formal grammars, especially lexicalized formal grammars such as TAG, that derive the structure of a sentence by combination of elementary structures. In particular, it seems more accurate for some problematic constructions (such as unbounded dependencies or complex determiners) to be formalized with a derivation structure which is a polygraph rather than a tree or even a graph.[19]

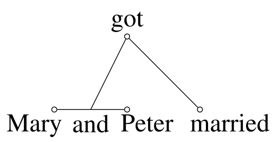
Acknowledgement

The authors would like to thank the following people for their suggestions and corrections (in no particular order): Kim Gerdes, François Lareau, Timothy Osborne and Pierre Swiggers.

[19]As stated earlier (sec. 3.5), CFG derivations have been extended to TAGs (Vijay-Shanker, 1992). However, even for TAGS, it appears that derivations are too complex to be fully formalized by a tree; e.g. predicative adjunction demonstrates the limits of the tree structure in this context (Schabes and Shieber, 1994). This question cannot be developed here; Kahane (2013) suggests polygraphic derivation structures in a dependency-based formalism.



(a) Headed constituency tree



(b) Polygraph with marker node

Figure 11: Ternary grouping with co-heads

References

Steven P. Abney. 1987. *The English Noun Phrase in its Sentential Aspect*. Ph.D. thesis, Massachusetts Institute of Technology.

Claude Bergé. 1973. *Graphs and hypergraphs*. North-Holland, Amsterdam.

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

Guillaume Bonfante and Yves Guiraud. 2008. Intensional properties of polygraphs. *Electronic Notes in Theoretical Computer Science*, 203(1):65–77.

Albert Burroni. 1993. Higher-dimensional word problems with applications to equational logic. *Theoretical computer science*, 115(1):43–62.

Noam Chomsky. 1957. *Syntactic structures*. Mouton, The Hague.

Eugenio Coseriu. 1980. Un précurseur méconnu de la syntaxe structurale: H. Tiktin. In *Recherches de linguistique: hommage à Maurice Leroy*, pages 48–62. Éditions de l'Université de Bruxelles, Bruxelles.

Fathi Débili. 1982. *Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques*. Ph.D. thesis, Université Paris Sud.

Henry A. Gleason. 1955. *An Introduction to Descriptive linguistics*. Holt, Rinehart and Winston.

Thomas Groß. 2011. Catenae in morphology. In Kim Gerdes, Elena Hajičová, and Leo Wanner, editors, *Proceedings of Depling 2011, International Conference on Dependency Linguistics, Barcelona*, pages 47–57. Barcelona.

Charles F. Hockett. 1958. *A course in modern linguistics*. The MacMillan Company.

Richard A. Hudson. 1980. Constituency and dependency. *Linguistics*, 18(3-4):179–198.

Richard Hudson. 1984. *Word Grammar*. Blackwell, Oxford.

Richard Hudson. 2010. *An introduction to word grammar*. Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge.

Ray Jackendoff. 1977. *X-bar syntax: A study of phrase structure*. MIT Press, Cambridge, MA.

Sylvain Kahane and Nicolas Mazziotta. submitted (Depling 2015). Dependency-based analyses for function words - introducing the polygraphic approach.

Sylvain Kahane and Timothy Osborne. 2015. Translators' introduction. In *Elements of structural syntax* (Tesnière, 2015), pages xxix–lxxiv.

Sylvain Kahane. 1997. Bubble trees and syntactic representations. In *Proceedings of Mathematics of Language (MOL5) Meeting*, pages 70–76. Citeseer.

Sylvain Kahane. 2012. Why to choose dependency rather than constituency for syntax: a formal point of view. In J. Apresjan, M.-C. L'Homme, M.-C. Iomdin, J. Milićević, A. Polguère, and L. Wanner, editors, *Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel'čuk*, pages 257–272. Languages of Slavic Culture, Moscow.

Sylvain Kahane. 2013. Predicative adjunction in a modular dependency grammar. In *2nd international conference on Dependency Linguistics (DepLing)*, pages 137–146.

Franz Kern. 1883. *Zur Methodik des deutschen Unterrichts*. Nicolai, Berlin.

Yves Lecerf. 1961. Une représentation algébrique de la structure des phrases dans diverses langues naturelles. *Comptes rendus de l'Académie des Sciences*, 252(2):232–235.

Nicolas Mazziotta. 2014. Nature et structure des relations syntaxiques dans le modèle de Lucien Tesnière. *Modèles linguistiques*, 69:123–152.

Igor Mel'čuk. 1988. *Dependency syntax: theory and practice*. State University of New York, Albany.

Igor Mel'čuk. 2009. Dependency in natural language. In Alain Polguère and Igor Mel'čuk, editors, *Dependency in linguistic description*, pages 1–110. John Benjamins, Amsterdam and Philadelphia.

Eugene Nida. 1949. *Morphology: the descriptive analysis of words*. University of Michigan press, Ann Arbor, 2 edition.

Eugene Nida. 1966. *A synopsys of English Syntax*. Mouton and Co., London, The Hague, Paris, 2 edition.

Timothy Osborne, Michael Putnam, and Thomas M. Gross. 2011. Bare phrase structure, label-less trees, and specifier-less syntax. is minimalism becoming a dependency grammar? *The Linguistic Review*, 28(3):315–364.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Alonso Reed and Brainerd Kellogg. 1877. *Higher Lessons in English: A Work on English Grammar and Composition*. Clark and Maynard, New-York.

Yves Schabes and Stuart M. Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

John F. Sowa. 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA.

Lucien Tesnière. 1934. Comment construire une syntaxe. *Bulletin de la Faculté des Lettres de Strasbourg*, 7:219–229.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Lucien Tesnière. 2015. *Elements of structural syntax, translated by Timothy Osborne and Sylvain Kahane*. Benjamins, Amsterdam/Philadelphia.

K. Vijay-Shanker. 1992. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517.

Rulon S. Wells. 1947. Immediate constituents. *Language*, 23(2):81–117.