Some Formal Considerations on the Generation of Hierarchically Structured Expressions*

Jordi Fortuny Andreu
University of Groningen. Center for Language and Cognition Groningen
J.Fortuny.Andreu@rug.nl

Universitat Pompeu Fabra
ICREA-Complex Systems Lab. Parc de Recerca Biomèdica de Barcelona
Bernat.Corominas@upf.edu

Abstract

In this note we define a machine that generates nests. The basic relations commonly attributed to linguistic expressions in configurational syntactic models as well as the device of chains postulated in current transformational grammar to represent distance relations can be naturally derived from the assumption that the combinatorial syntactic procedure is a nesting machine. Accordingly, the core of the transformational generative syntactic theory of language can be solidly constructed on the basis of nests, in the same terms as the general theory of order, an important methodological step that provides a rigorization of Chomsky's minimalist intuition that the simplest way to generate hierarchically organized linguistic expressions is by postulating a combinatorial operation called Merge, which can be internal or external. Importantly, there is reason to think that nests are a useful representative tool in other domains besides language where either some recursive algorithm or evolutionary process is at work, which suggests the unifying force of the mathematical abstraction this note is based on.

Key words: nest, theory of order, dominance, domain, constituency, chain (of copies), rigorization.

Table of Contents

* We would like to express our gratitude to Txuss Martín, Jeroni Tutusaus and Jan-Wouter Zwart for helpful comments. This work has been founded by the NWO-sponsored research project *Dependency in Universal Grammar* (JF), the James McDonnell Foundation and the grant FIS2004-0542, IST-FET ECAGENTS project of the E.U. under R&D contract 011940 (BCM).

1. Introduction

The intuitive development of an incipient theory is a necessary step in any rational enterprise. However, when the theory grows and its degree of complexity increases, the lack of internal rigor can render the theory impracticable and lead it to undesirable situations such as the introduction of redundant or unnecessary principles that jeopardize the credibility of the intellectual construct. Therefore, the rigorization of the core notions and the derivation of the main conclusions has the virtue of ensuring internal consistency and providing a healthy backbone that not only sustains the theory, but also leads it towards further developments.

The main objective of this note is to provide a rigorous definition of the basic syntactic algorithm and derive the basic syntactic relations from a minimal collection of general assumptions.

Section 2 proposes a rigorized model for the core syntactic theory. Section 3 exposes our view about the abstract character of the algorithm of structure creation defined in section 2 as well as the place it deserves in the study of language and justifies our technical choices concerning the direction of derivations and the formulation of distance relations. Section 4 briefly summarizes the main conclusion of this work by stressing the unifying force of nests, the mathematical concept on which our proposal is based.

2. The nesting machine

In this section we define an abstract machine that generates hierarchically structured expressions. The basic units that this machine manipulates are given by an alphabet, which we define as a finite set of singletons. The outcome of this machine is a nest, i.e., a set whose elements are sets linearly ordered by inclusion. A computation by the nesting machine may comprise a sole derivational space (2.1) or multiple derivational spaces (2.2), in which case the outcome of a particular derivational space D_i feeds a different derivational space D_j. If the syntactic algorithm is assumed to be a nesting machine, the common syntactic relations can be readily defined with no need to introduce any idiosyncratic grammatical element. More precisely, not only the linear order among the terminals of an expression can be properly defined on the basis of Kuratowski's (1921) general set-theoretical definition of order, as advanced in Fortuny (2008), but also the constituency relationship, the dominance relationship and the concepts of domain and chain.

2.1. Simple nesting

Given an alphabet, viewed as a finite set of singletons, $A = \{\{a\}, \{b\}, \ldots, \{z\}\}$, we define a machine that works as follows. At the first step, s_0, this machine generates the set M_0, which is an element of A. At the following step s_1, it generates a new set, M_1, by forming the union of M_0 and a member of A. At step s_n, we generate the set M_n, which is the union of M_{n-1} and an element of A. When an arbitrary element of A, namely $\{k\}$, comes into the computation at the step s_i, its element, k, becomes an occurrence k_i. This can be summarized through the recursive operation:

$$M_0 = \{a_0\}$$
$$M_{n+1} = \{k_{n+1}\} \cup M_n$$

where n is unboundedly large[1]. The outcome of the machine is the family[2]:

$$\boldsymbol{N} = \{M_0, \ldots, M_{n+1}\}.$$

N is a nest, i.e., a family of sets linearly ordered by the inclusion relation[3]:

$$M_0 \subset M_1 \subset M_2 \subset \ldots \subset M_{n+1}$$

A family ***F*** of sets is a nest of a set *S* iff the elements of ***F*** are subsets of *S* and they are linearly ordered by inclusion. A family ***F*** of sets is saturated as to the property of being a nest of *S* iff ***F*** is a nest and it is not a proper subset of a nest of *S*. For instance, $\boldsymbol{F_1} = \{\{g_1\}, \{g_1, g_2\},\ldots,\{g_1, g_2,\ldots, g_{n-1}\}, \{g_1, g_2, \ldots, g_{n-1}, g_n\}\}$ is saturated as to the property of being a nest of $P = \{g_1, g_2,.., g_{n-1}, g_n\}$, but not $\boldsymbol{F_2} = \{\{g_1\}, \{g_1, g_2\},\ldots, \{g_1, g_2,\ldots, g_{n-1}\}\}$, which is a subset of $\boldsymbol{F_1}$. As proved by Kuratowski (1921), ***F*** linearly orders *S* iff ***F*** is saturated as to being a nest of *S*, and thus $\boldsymbol{F_1}$ (but not $\boldsymbol{F_2}$) is a linear order of *P*.[4]

Observe that *P* is the union of $\boldsymbol{F_1}$, noted as $\cup\boldsymbol{F_1}$, the set whose elements are all the elements of the elements of $\boldsymbol{F_1}$. Thus, we shall simply say that the linear ordering by inclusion we find among the elements of an outcome ***N*** of the nesting machine induces a linear ordering of $\cup\boldsymbol{N}$, the set of occurrences of the elements of the relevant alphabet.

1. That *n* lacks an upper bound does not mean that *n* must be infinite. That there is no fixed length for a nesting computation is consistent with the claim that a given nesting computation must always be finite (or that the cardinality of a nest generated by the nesting machine must always be finite). We refer the interested reader to Partee et alii (1990: 69-70) for the distinction between the terms 'unbounded' and 'infinite'.
2. We shall adopt the term 'family' to refer to sets whose elements are all sets. For the sake of representational clarity, upper case bold letters ***A***, ***B***, ***C***, … shall designate families, upper case non-bold letters *A*, *B*, *C*,… sets (families or not), and lower case letters *a*, *b*, *c*, … objects (without specifying whether they are primitive elements or sets). The membership relation ('$\in$') holds between a set and its elements or members. For instance, it is said that *a* and $\{b\}$ are elements or members of or belong to the set $\{a, \{b\}\}$, but $\{a\}$ or *b* are not. The inclusion relation ('$\subseteq$') holds between a set and its subsets. *P* is said to be included in *Q* iff all the elements of *P* are elements of *Q*. An asymmetric inclusion relation is called a proper inclusion relation ('$\subset$'). Thus, $\{a\}$ and $\{\{b\}\}$ are both (properly) included in $\{a, \{b\}\}$, and in fact they do not belong to this set. Note that $\{a, b\}$ both belongs to and is included in $\{a, b, \{a, b\}\}$. Whereas inclusion is a transitive relation, membership is not.
3. A relation *R* is a linear order iff it is antisymmetric, transitive and total.
4. Kuratowski used the expression "classe de sous-ensembles décroissants (ou croissants)" to refer to a set whose elements are sets linearly ordered by inclusion. Here and elsewhere we use the term 'nest' to refer to this type of set. Other terms found in the literature are 'tower' and 'chain' (Kelley 1955: 32).

We define the constituent $\boldsymbol{C}_k$ $(0 \leq k \leq n+1)$ in a nest $\boldsymbol{N}$ as the outcome of the computation at a particular step s_k, $\boldsymbol{C}_k = \{M_0, M_1, \ldots, M_k\}$. Note that (*a*) $\boldsymbol{C}_k$ is a family of sets linearly ordered by inclusion, $M_0 \subset M_1 \subset M_2 \subset \ldots \subset M_k$, that (*b*) $\boldsymbol{C}_k \subseteq \boldsymbol{N}$ and that (*c*) the $\boldsymbol{C}$s in $\boldsymbol{N}$ are linearly ordered by inclusion. When $k = 0$, the constituent $\boldsymbol{C}_0$ is a trivial nest, with only one element.

Given a nest $\boldsymbol{N}$, $M_k \in \boldsymbol{N}$ dominates b, noted as DOM(M_k, b), iff:

$$(b \in \boldsymbol{N} \wedge b \subset M_k) \vee (b \in M_k).$$

M_k immediately dominates b in $\boldsymbol{N}$ iff DOM(M_k, b) and $\neg\exists(F \in \boldsymbol{N})$: [DOM($M_k$, F) $\wedge$ DOM(F, b)]. We observe that DOM is a linear order in any nest $\boldsymbol{N}$.

The nesting machine itself ensures the existence of an M_k that is the largest element in DOM. Given a nest $\boldsymbol{N}$, M_k ($M_k \in \boldsymbol{N}$) is the largest element in the dominance relation iff it dominates any b, b being either a member of $\boldsymbol{N}$ or a member of any M_j.[5] Given a nest $\boldsymbol{X}$, we shall refer to the largest element of $\boldsymbol{X}$ as larg($\boldsymbol{X}$). Consistently, the largest element of a constituent $\boldsymbol{C}_k$ is larg($\boldsymbol{C}_k$) $= M_k = \cup \boldsymbol{C}_k$.

We shall say, along with Chomsky (2001, 2008), that X is externally merged to Y when X is selected from the alphabet and internally merged to Y when it is selected from "the domain of" Y. Therefore, we allow the nesting machine to perform at a stage s_i the operation $X \cup Y$ when X is an element of the alphabet but also when X belongs to "the domain of" Y. Observe that the set Y to which X is merged at s_i is M_{i-1}. Generally, given the set Γ of all constituents in a given derivation, we define the domain of M_j as the set $\Delta(M_j)$:

$$\Delta(M_j) = \{x\text{: } [(x \in \Gamma) \wedge \text{DOM}(M_j, \text{larg}(x))] \vee (x \in M_j)\}.$$

x becomes an occurrence x_j when it is externally merged at s_j and a copy $x_{j/i}$ when it is internally merged at $s_{i>j}$. A chain $\boldsymbol{CH}(x_j)$ can be defined as a linear order of the copies of an occurrence x_j, $\boldsymbol{CH}(x_j) = \{\{x_{j/k}, \ldots, x_j\}, \{\ldots, x_j\}, \{x_j\}\}$. We shall call x_j the copy tail of $\boldsymbol{CH}(x_j)$, $x_{j/k}$ the copy head and any $x_{j/n(j<n<k)}$ an intermediate copy. Note that multiple copies are identified by virtue of the subindex referring to the derivational stage where the occurrence X is introduced in the derivation ('j') and distinguished with respect to each other by virtue of the subindexical suffix referring to the derivational stage where they are subsequently merged ('$_{/i}$'). If the generation of a nest $\boldsymbol{N}$ involves internal merging, $\cup\boldsymbol{N}$ is a set of occurrences and copies of the selected elements of the alphabet.

Therefore, the notions of order, constituent, dominance, domain and chain can be naturally defined on the basis of the same mathematical structure, which reinforces the concept of nest as a fundamental unifying entity of structural relations (see sections 3.2 and 3.3 for illustration).

5. This requirement resembles the Single Root Condition defined on tree structures.
The Single Root Condition (Partee et *alii* 1990: 441)
'In every well-formed constituent structure tree there is exactly one node that dominates every node'
Note, though, that this statement does not consider dominance from nodes to terminals.

2.2. Complex nesting

We want to allow the nesting machine to perform $X \cup Y$, when X is a set M_j generated by the nesting machine at s_j and Y a singleton whose element is not a primitive element but a nest.

To enable our machine to perform such operations, we need to define several derivational spaces, which we shall label as D_1, D_2, ..., D_n. For the sake of clarity, the union operation between two sets performed in the derivational space D_j at the step s_k will be labeled as M_k^j. Every derivational space involves a machine like the one defined in the above section, except that at a given step s_i, the nesting machine can accept as input M_{i-1}^k and $\{a\}$ not only when $\{a\}$ belongs to the alphabet but also when a is an outcome of a derivation performed in a parallel space. When $\{a\}$ has been merged at the step s_k in the derivational space D_j it becomes a_k^j; $\boldsymbol{N}_{[i]k}^{\ j}$ is the outcome of D_i merged at s_k in D_j. Once D_i generates $\boldsymbol{N}_i$, D_i is automatically removed, which means that $\boldsymbol{N}_i$ cannot grow anymore although this does not imply that the nesting machine lacks the power of internally merging Y from $\boldsymbol{N}_i$ to M_k^j when D_i feeds D_k.

The following two conditions are required:

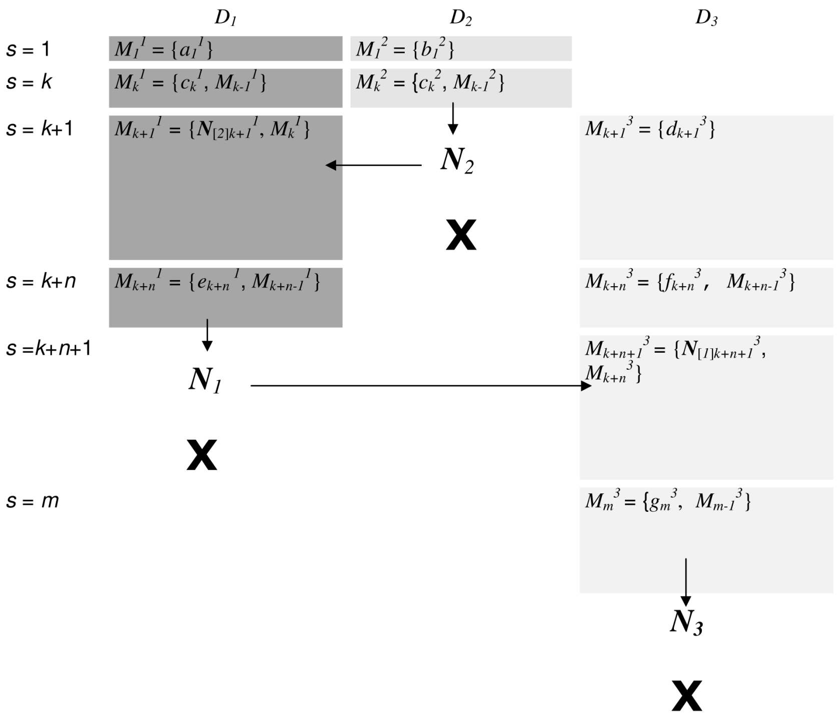
a) Let $|D|_{Sj}$ be the number of opened derivational spaces at the step s_j. Then, there exists a finite constant μ such that:

$$\forall(S_j)\ (|D|_{Sj} < \mu).$$

There are strong reasons to take this condition for granted. Firstly, it implies that the memory of the machine is finite, which is a reasonable assumption. Secondly, if we want to decide whether or not a given object is the outcome of a computation performed by the machine, the condition avoids the halting problem.[6]

b) At the last step of a given computation, only one derivational space D_j can remain opened.
This implies that all derivational spaces used in the computation generated their outcomes as inputs for other derivational spaces (see Figure 1).

6. If the number of derivational spaces was unbounded, there would exist the possibility that the algorithm responsible for deciding whether a given object belongs to the set of the outcomes of a machine would never halt. We refer the reader to Chaitin (1977), Davis (1958) and Hermes (1965) for a compact and broad presentation of the problems associated to computability theory.


D_1
D_2
D_3
$s = 1$
$s = k$
$s = k+1$
$s = k+n$
$s = k+n+1$
$s = m$
$M_1^{\,1} = \{a_1^{\,1}\}$
$M_1^{\,2} = \{b_1^{\,2}\}$
$M_k^{\,1} = \{c_k^{\,1}, M_{k-1}^{\,1}\}$
$M_k^{\,2} = \{c_k^{\,2}, M_{k-1}^{\,2}\}$
$M_{k+1}^{\,1} = \{\boldsymbol{N}_{[2]k+1}^{\,1}, M_k^{\,1}\}$
$M_{k+1}^{\,3} = \{d_{k+1}^{\,3}\}$
$\boldsymbol{N}_2$
X
$M_{k+n}^{\,1} = \{e_{k+n}^{\,1}, M_{k+n-1}^{\,1}\}$
$M_{k+n}^{\,3} = \{f_{k+n}^{\,3}, M_{k+n-1}^{\,3}\}$
$M_{k+n+1}^{\,3} = \{\boldsymbol{N}_{[1]k+n+1}^{\,3}, M_{k+n}^{\,3}\}$
$\boldsymbol{N}_1$
X
$M_m^{\,3} = \{g_m^{\,3}, M_{m-1}^{\,3}\}$
$\boldsymbol{N}_3$
X

$\boldsymbol{N}_1 = \{\{a_1^{\,1}\}, \{a_1^{\,1}, ...\}, \{a_1^{\,1}, ..., c_k^{\,1}\}, \{a_1^{\,1}, ..., c_k^{\,1}, \boldsymbol{N}_2\}, \{a_1^{\,1}, ..., c_k^{\,1}, \boldsymbol{N}_2, ...\}, \{a_1^{\,1}, ..., c_k^{\,1}, \boldsymbol{N}_2, ..., e_{k+n}^{\,1}\}\}$

$\boldsymbol{N}_2 = \{\{b_1^{\,2}\}, \{b_1^{\,2}, ...\}, \{b_1^{\,2}, ..., c_k^{\,2}\}\}$

$\boldsymbol{N}_3 = \{\{d_{k+1}^{\,3}\}, \{d_{k+1}^{\,3}, ...\}, \{d_{k+1}^{\,3}, ..., f_{k+n}^{\,3}\}, \{d_{k+1}^{\,3}, ..., f_{k+n}^{\,3}, \boldsymbol{N}_1\}, \{d_{k+1}^{\,3}, ..., f_{k+n}^{\,3}, \boldsymbol{N}_1, ...\}, \{d_{k+1}^{\,3}, ..., f_{k+n}^{\,3}, \boldsymbol{N}_1, ..., g_m^{\,3}\}\}$

Figure 1. A picture of the application of the nesting algorithm over an arbitrary alphabet enabling complex nesting. In this example, three derivational spaces have been used and we assume that $\mu < 3$, i.e., at most two derivational spaces can be opened at the same time. Observe that, at the last step, only one derivational space is opened.

We observe that the outcome that we obtain if we stop the computation at a certain step s_k in a given derivational space D_i is a constituent, the constituent $\boldsymbol{C}_k^{\,i}$. Some constituents of the complex derivation given in Figure 1 are $\boldsymbol{C}_1^{\,1} = \{a_1^{\,1}\}$, $\boldsymbol{C}_{k+1}^{\,1} = \{\{a_1^{\,1}\}, \{a_1^{\,1}, ...\}, \{a_1^{\,1}, ..., c_k^{\,1}\}, \{a_1^{\,1}, ..., c_k^{\,1}, \boldsymbol{N}_2\}\}$ and $\boldsymbol{C}_m^{\,3} = \boldsymbol{N}_3$.

It is necessary to define the notions of dominance and domain not only with respect to a sole derivational space (see 2.1), but also with respect to a complex nesting computation.

Let $D_1, \ldots, D_n$ be all the derivational spaces used in a computation. We now form: (i) the set Σ of all the outcomes $\boldsymbol{N}_1,..., \boldsymbol{N}_n$ generated respectively in the derivational spaces $D_1,\ldots, D_n$, (ii) the set $\boldsymbol{M}$ of all the elements of all $\boldsymbol{N}_1,.., \boldsymbol{N}_n$, namely, the union of Σ, and (iii) the set Φ of all the elements of all the elements of all $\boldsymbol{N}_1,..,$ $\boldsymbol{N}_n$, or more simply, the union of all Max($\boldsymbol{N}_i$).

$$\begin{aligned}\Sigma &= \{\boldsymbol{N}_1, ..., \boldsymbol{N}_n\}\\ \boldsymbol{M} &= \cup_{i\leq n} \boldsymbol{N}_i\\ \Phi &= \cup_{i\leq n} \mathrm{Max}(\boldsymbol{N}_i)\end{aligned}$$

Thus, $(\forall b)$ $(b \in \boldsymbol{M} \cup \Phi)$, DOM($M_j^k$, b) iff:

a) $[(b \subset M_j^k) \vee (b \in M_j^k)] \vee$
b) $[\exists(n < \infty)\ \exists(\{X_1, \ldots, X_n\})$: $((\forall i\leq n)\ (X_i \in \boldsymbol{M} \cup \Sigma) \wedge (b \in X_1 \neq M_j^k)) \wedge (X_1 \in X_2 \in \ldots \in X_n \in M_j^k)]$[7]

Consider for clarity how our definition applies to the computation represented in Figure 1. Condition (a) is no more than the definition of dominance given for simple nesting in section 2.1 though relativized to a particular derivational space D_k, as the superindex '^{k}' of M_j^k indicates. By virtue of condition (a), $M_{k+n}{}^1 = \{a_1{}^1, ..., c_k{}^1, \boldsymbol{N}_2, ..., e_{k+n}{}^1\}$ dominates, for instance, $e_{k+n}{}^1$, and $M_k{}^1 = \{a_1{}^1, ..., c_k{}^1\}$, since $(e_{k+n}{}^1 \in \boldsymbol{M} \cup \Phi) \wedge (e_{k+n}{}^1 \in M_{k+n}{}^1)$ and $(M_k{}^1 \in \boldsymbol{M} \cup \Phi) \wedge (M_k{}^1 \subset M_{k+n}{}^1)$.

Let us consider an example in order to understand condition (b). We observe that n is the number of different derivational spaces involved in the derivation from b to M_j^k. Does $M_m{}^3$ dominate $M_k{}^2$? Note that $M_k{}^2 = \{b_1{}^2, ..., c_k{}^2\}$ belongs to $\boldsymbol{N}_2 = \{\{b_1{}^2\}, \{b_1{}^2, ...\}, \{b_1{}^2, ..., c_k{}^2\}\}$ and $\boldsymbol{N}_2$ belongs, for instance, to $M_{k+1}{}^1$, which in turn belongs to $\boldsymbol{N}_1$; finally we observe that $\boldsymbol{N}_1$ belongs to $M_m{}^3 = \{d_{k+1}{}^3, ..., f_{k+n}{}^3, \boldsymbol{N}_1, ..., g_m{}^3\}$. Thus there exists a sequence X_1, X_2, X_3 (being $X_1 = \boldsymbol{N}_2$, $X_2 = M_{k+1}{}^1$, $X_3 = \boldsymbol{N}_1$ and $X_4 = M_m{}^3$). We can conclude that $\exists(\{X_1 X_2, X_3\})$: $(X_i \in \boldsymbol{M} \cup \Sigma) \wedge (\mathrm{M}_k{}^2 \in X_1 \neq M_m{}^3) \wedge (X_1 \in X_2 \in X_3 \in M_m{}^3)$. The same reasoning applies to the elements of Φ if we find an appropiate sequence of Xs.

Note that DOM in a complex nesting computation is not a linear order, since it does not satisfy the totality condition; for instance, in the derivation depicted in Figure 1, neither DOM($M_k{}^2$, $M_k{}^1$) nor DOM($M_k{}^1$, $M_k{}^2$).

Finally, we define the domain of an M_k^j, i.e., of the set created by the union-formation application k of the derivational space D_j. We only need to elaborate the definition of domain given in section 2.1 for simple nesting by relativizing the set Γ of constituents to D_i (as indicated by subindexation -Γ_i) and by appealing to the subset of elements of Φ dominated by M_k^j, instead of simply M_k.

$$\Delta(M_k^j) = \{x\text{: } [(\exists i\leq n)\text{: } (x \in \Gamma_i) \wedge \mathrm{DOM}(M_k^j, \mathrm{Max}(x))] \vee [(x \in \Phi) \wedge \mathrm{DOM}(M_k^j, x)]\}.$$

7. This definition of DOM in a complex nesting computation is precisely the transitive closure (Gross & Yellen 1998: 371) of the relation DOM defined in the previous section.

3. Remarks and reflections

3.1. The nesting machine as an abstract entity

We have defined a mathematical entity and we have studied some of its properties. The above developed formalism is intended to be a consistent mathematical apparatus supporting the current generative transformational framework. As any abstract theoretical background, it is not reasonable to ask about the 'reality' of the operations and objects defined. For example, although the algorithm runs through a time step indicator, such time step is only given for operational purposes and does not imply -in our field of study- any temporal evolution. What is reasonable to ask is whether from the defined mathematical framework we can derive the core properties that we observe in the studied object. This is a common procedure in other scientific domains, like theoretical physics. As a paradigmatic example, we can take Quantum Mechanics, constructed on the basis of solely six axioms. Specifically, the theory is based on Hilbert spaces and the theory of abstract operators associated to it (Peskin et *al.* 1995). One of the most fundamental operators is the *creation operator* which *creates* a particle in a given state, which could be compared with the algorithm we have defined to create nests. Indeed, let $|F_0>$[8] be the vacuum state, the state of a system where there is no particle. The application of the creation operator, $a^\dagger(\boldsymbol{k})$, *creates* a particle of momentum $\boldsymbol{k}$ in this system. Thus, we can construct $|F_1(\boldsymbol{k})>$ as

$$|\boldsymbol{F}_1(\boldsymbol{k})> = a^\dagger(\boldsymbol{k})\,|\boldsymbol{F}_0>$$

Similarly, we can construct a state with $n+1$ particles by applying recursively $a^\dagger(\boldsymbol{k})$ $n+1$ times:[9]

$$\begin{aligned}|\boldsymbol{F}_1(\boldsymbol{k})> &= a^\dagger(\boldsymbol{k})\,|\boldsymbol{F}_0(\boldsymbol{k})>\\ |\boldsymbol{F}_{n+1}(\boldsymbol{k})> &= a^\dagger(\boldsymbol{k})\,|\boldsymbol{F}_n(\boldsymbol{k})>\end{aligned}$$

Obviously, we cannot ask whether or not such operator exists in the real world, but its abstract existence is necessary to construct the whole theory of Quantum Mechanics. This is just an analogy to emphasize that the nesting machine is a mathematical entity that is not intended to describe any neuronal or material process. A different question is how the underlying computational power can emerge from the growing of computing assemblies, like the brain.

3.2. On the direction of derivations

Given the alphabet $\boldsymbol{A} = \{\{\text{John}\}, \{\text{Mary}\}, \{\text{kisses}\}\}$, the nesting machine could proceed as follows:

8. A quantum state is completely defined by a vector of a Hilbert space, and the standard notation is $|\psi>$. This vector encodes all the physically relevant information of such state. In our case, we noted $|\boldsymbol{F}_n(\boldsymbol{k})>$ as the state with n particles with momenta equal to $\boldsymbol{k}$.
9. In this illustration we neglect the normalization form in order to avoid certain mathematical technicalities that are not relevant to our concerns).

$s_1 = \{\text{John}\}$
$s_2 = \{\text{John, kisses}\}$
$s_3 = \{\text{John, kisses, Mary}\}$,

thereby generating the outcome ***N***, a set saturated as to the property of being a nest of {John, kisses, Mary}:

N = {{John}, {John, kisses}, {John, kisses, Mary}}.

We ask whether ***N*** is the representation lying under the expression (*i*) 'John kisses Mary' or under the expression (*ii*) 'Mary kisses John'. If ***N*** underlies (*i*), then the nesting machine is a top-down procedure and ***N*** is read as a precedence relation by the Articulary-Perceptual system. If ***N*** underlies (*ii*), then the nesting machine is a bottom-up procedure and ***N*** is read as a successor relation by the Articulary-Perceptual system. We observe that, if ***N*** lies under (*i*), we predict that the subject and the verb form a constituent to the exclusion of the object, whereas if ***N*** lies under (*ii*), we predict that the object forms a constituent with the verb to the exclusion of the subject. Since the latter prediction, and not the former, is in accordance with standard constituency considerations, we conclude that the nesting machine is a bottom-up procedure and that its outcome is read as a successor relation.

We want to observe that it is possible to generate nests by means of a top-down procedure where smaller sets are generated from larger sets that yields the expected constituency. Such a procedure resorts not to recursive union formation but to recursive complementary subset formation.

Given a finite set A of terminal elements, the machine takes A as an initial symbol at s_0 and yields M_0, the complementary subset of Ø; thus $M_0 = A$. At s_1 the machine generates M_1, a complementary subset of a singleton subset of M_0. At S_n, we generate M_n, the complementary subset of a singleton in M_{n-1}. Recall that, in section 2.1, multiple occurrences of the same type are distinguished with regard to the derivational step where they are merged, but they are not distinguished as elements of the alphabet. In order to ensure that a syntactic derivation based on the splitting mechanism under consideration can make use of multiple occurrences of a lexical item, it is mandatory to assume that A can have multiple occurrences a_1, a_2, …, a_k of a lexical item a as elements, as well as multiple occurrences a, b, …, n of different lexical items.

Given an initial symbol $A = \{a_1, a_2, \ldots, a_k, b, \ldots, n\}$, with $|A|$ unbounded, the splitting algorithm can be expressed through the recursive operation:

$$\begin{aligned} M_0 &= A \\ M_1 &= M_0 - B_0 \\ M_{n+1} &= M_n - B_n \end{aligned}$$

where $B_0 \subset M_0, \ldots, B_n \subset M_n$ and $n \leq |A|$. If $B_0, \ldots, B_n$ are singletons, then we are in a situation equivalent to that described in section 2.1, *simple nesting*:

$$\boldsymbol{N} = \{M_0, \ldots, M_{n+1}\}.$$

To generate nests saturated with respect to an initial symbol A (i.e, to consider we reach the end of the derivation), we need to split A until we obtain a singleton, which implies that we performed $|A|$ splittings. Thus the saturated nest will be:

$$\boldsymbol{N} = \{M_0, \ldots, M_{|A|}\}.$$

However, there is no reason to forbid the nesting machine to perform $X - Y$ when $|Y| > 1$. Accordingly, a complex nesting computation involving multiple spaces equivalent to that defined in section 2.2 is required. If s_k in D_j forms the complementary subset of a non-singleton subset B of M_{k-1}^{j}, then B becomes the initial symbol of a new derivational space D_i. As in section 2.2, all D is a machine that generates nests, the number of opened derivational spaces at a stage must be finite in order to avoid the halting problem, and only one D remains opened at the last step of a computation.

Whereas union formation can generate chains of multiple copies of an occurrence without further complications (section 2.1), it is unclear how subset complementary formation could generate a chain without postulating a particular syntactic operation for it. For this reason, we formulate the nesting machine as a (bottom-up) union formation algorithm.[10]

3.3. Chains

Consider the following two English sentences, the indicated syntactic derivations D_1, D_2, their outcomes $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, and the indicated constituency.

'John visited Peter'	'Who did you visit?'
$A_1 = \{\{\text{John}\}, \{\text{visited}\}, \{\text{Peter}\}\}$	$A_2 = \{\{\text{who}\}, \{\text{did}\}, \{\text{you}\}, \{\text{visit}\}\}$
D_1	D_2
$s_1 = \{\text{Peter}\}$	$s_1 = \{\text{visit}\}$
$s_2 = \{\text{visited, Peter}\}$	$s_2 = \{\text{visit, you}\}$
$s_3 = \{\text{John, visited, Peter}\}$	$s_3 = \{\text{visit, you, did}\}$
	$s_4 = \{\text{visit, you, did, who}\}$
$\boldsymbol{N_1} = \{\{\text{John, visited, Peter}\}, \{\text{visited, Peter}\},\{\text{Peter}\}\}$	$\boldsymbol{N_2} = \{\{\text{who, did, you, visit}\}, \{\text{did, you, visit}\}, \{\text{you, visit}\}, \{\text{visit}\}\}.$
$\boldsymbol{C_1} = \{\{\text{Peter}\}\}$	$\boldsymbol{C_1} = \{\{\text{visit}\}\}$
$\boldsymbol{C_2} = \{\{\text{visited, Peter}\}, \{\text{Peter}\}\}$	$\boldsymbol{C_2} = \{\{\text{you, visit}\}, \{\text{visit}\}\}$
$\boldsymbol{C_3} = \{\{\text{John, visited, Peter}\}, \{\text{visited, Peter}\}, \{\text{Peter}\}\}$	$\boldsymbol{C_3} = \{\{\text{did, you, visit}\}, \{\text{you, visit}, \{\text{visit}\}\}$
	$\boldsymbol{C_4} = \{\{\text{who, did, you, visit}\}, \{\text{did, you, visit}\}, \{\text{you, visit}\}, \{\{\text{visit}\}\}$

10. Cfr. Zwart (this volume) for a tentative exploration of top-down and layered derivations that argues against internal merge applications.

If both $\boldsymbol{N_1}$ and $\boldsymbol{N_2}$ were useful representations for the two sentences under discussion, the nesting machine would be an algorithm responsible for linearly ordering occurrences, but would not suffice to provide an accurate configurational basis for semantic relations. Crucially, whereas the noun 'Peter' can be argued to be characteristically identified as the patient of the verb 'visited' in $\boldsymbol{C_2}$, there is no way to warrant in configurational terms that 'who' is identified as the patient of 'visit' in $\boldsymbol{N_2}$: there is no constituent in $\boldsymbol{N_2}$ that can be characteristically identified as the minimal context where a nominal category x becomes a patient of a verb.

With the objective of providing the syntactic means of representing multiple semantic features of an occurrence in configurational terms, we have defined the nesting machine in such a way that it generates chains of multiple copies of an occurrence (section 2.1), thereby adopting the standard view in minimalist syntax that an occurrence acquires multiple semantic features because it has been merged and subsequently remerged at multiple syntactic positions. For the sake of explicitness, we indicate the derivation D_2' for the expression 'who did you visit?', its outcome $\boldsymbol{N_2}$**'** and the generated chain $\boldsymbol{CH}(who_1)$ that comprises the multiple copies of the occurrence 'who$_1$'. We also indicate the constituent $\boldsymbol{C_2}$ where 'who$_i$' is characteristically identified as an object and the constituent $\boldsymbol{C_5}$ where it is characteristically identified as a *wh*-phrase.

D_2'
$s_1 = \{\text{who}_1\}$
$s_2 = \{\text{visit, who}_1\}$
$s_3 = \{\text{you, visit, who}_1\}$
$s_4 = \{\text{did, you, visit, who}_1\}$
$s_5 = \{\text{who}_1\text{, did, you, visit, who}_{1/5}\}$

$\boldsymbol{N_2}$' = $\{\{\text{who}_{1/5}\text{, did, you, visit, who}_1\}, \{\text{did, you, visit, who}_1\}, \{\text{you, visit, who}_1\}, \{\text{visit, who}_1\}, \{\text{who}_1\}\}$

$\mathbf{CH}(\text{who}_1) = \{\{\text{who}_1\}, \{\text{who}_1, \text{who}_{1/5}\}\}$

$\boldsymbol{C_2} = \{\{\text{visit, who}_1\}, \{\text{who}_1\}\}$
$\boldsymbol{C_5} = \boldsymbol{N_2}$'

Observe that 'who$_1$' belongs to $\Delta(M_4)$, since 'who$_1$' $\in M_4$. Therefore, 'who$_1$' can be selected from $\Delta(M_4)$ and be internally merged to M_4, thereby forming M_5.

Chain formation also plays a crucial role in constructing a theory of word order variation along the course initiated by Kayne (1994). If we express Kayne's *X'*-theoretic framework in terms of our set-theoretical approach, we shall say that x is interpreted as the complement of a head h iff there exists a constituent $\boldsymbol{C_j} = \{\{x_i\}, \{x_i, h\}\}$ and that y is interpreted as the specifier of h iff there is a constituent $\boldsymbol{C_k} = \{\{x_i\}, \{x_i, h\}, \{x_i, h, y\}\}$ and y is not a head.[11] If, for instance, the complement x turns out to precede h in an utterance, this is because a further constituent $\boldsymbol{C_l} = \{\{x_i\},$

11. The definition and review of the notion of head (and projection) are beyond the scope of this note.

$\{x_i, h\}\}, \{x_i, h, \ldots\}, \{x_i, h, \ldots, x_{i/l}\}\}$ has been generated by internal merge, and the most deeply nested copy x_i of the chain $\mathbf{CH}(x_i) = \{\{x_i\}, \{x_i, x_{i/l}\}\}$ is not pronounced; similarly, if the specifier y turns out to follow h this is attributed to h being internally merged in a position less deeply nested than y.[12]

We emphasize, adapting Chomsky's (2001: 9, note 29) view, that the virtue of allowing the nesting machine to perform internal merge is a matter of applicability and of conceptual necessity: it is a matter of applicability because it is a simple analytical tool that is constructively used to capture the property of 'displacement', which seems ubiquitous in natural languages, and a matter of conceptual necessity because only by stipulation could the nesting machine be banned to perform $X \cup Y$ at s_i when X is M_{i-1} and Y belongs to $\Delta(M_{i-1})$, a stipulation that would seem unmotivated, given our current understanding of the syntactic patterns of natural languages.

4. Conclusion

In this note we have been concerned with the theory of syntax. More particularly, we have been concerned with a set-theoretical definition of an algorithm that generates hierarchically structured expressions that can potentially support grammatical systems of a formidable generative power. However, the object we observe, built up on the basis of the nesting machine, has to satisfy constraints belonging to multiple domains such as (a) learnability, (b) material embedding and evolution of the computing assembly, (c) interpretability and (d) mathematical information theory. We thus emphasize the truism that the theory of order we have rigorously defined is no more than a component of a general theory of language.

We have constructed our theory of hierarchical expressions on the basis of the concept of nest, thereby applying Kuratowski's general theory of order. We have shown that, by properly exploring the features of nested structures, several core syntactic properties can be rigorously derived. The concept of nest is indeed a powerful abstract entity postulated in different domains, like Theoretical Biology (Bascompte et *al.* 2003), Statistical Physics (Dorogovtsev et *al.* 2006, Corominas Murtra et *al.* 2008) or Genetics (Rodriguez Caso et *al.* 2005, Corominas Murtra et *al.* 2007). Therefore, nestedness does not only play a crucial role in our specific domain (where we have defined the notions of constituent, dominance, domain and chain on the basis of the same mathematical object), but it is also a useful abstraction that seems to shed light in understanding several natural phenomena where either a recursive algorithm or an evolutionary process is at work.

12. Observe that the nesting machine allows multiple specifiers of a head and does not require the stipulation of abstract functional heads to host a category in a derived position, thereby differing from Kayne's (1994) Linear Correspondence Axiom.

References

Bascompte, Jordi, Pedro Jordano, Carlos J. Melián & Jens M. Olesen (2003). "The nested assembly of plant-animal mutualistic networks". *Proceedings of the National Academy of Science USA* 100: 9.383-9.387.

Chaitin, Gregory J. (1977). "Algorithmic Information Theory". *IBM Journal of Research and Development* 21: 350-359.

Chomsky, Noam (2001). "Beyond explanatory adequacy". *MIT Occasional Working Papers in Linguistics* 20: 1-18. Cambridge, Mass.: The MIT Press.

Chomsky, Noam (2008). "On phases". In Freidin, Robert, Carlos P. Otero and Maria Luisa Zubizarreta (eds.), *Foundational Issues in Linguistic Theory*, 133-166. Cambridge: MIT Press.

Corominas-Murtra, Bernat, José F. F. Mendes & Ricard V. Solé (2007). "Nested Subgraphs of Complex Networks". *Santa Fe Institute Working Papers*. 07-12-050.

Corominas-Murtra, Bernat, Sergi Valverde, Carlos Rodríguez-Caso & Ricard V. Solé (2008). "K-scaffold subgraphs in complex networks". *Europhysics Letters* 77, 18004.

Davis, Martin (1958). *Computability and Unsolvability*. New York: McGraw-Hill.

Dorogovtsev, S. N., A. V. Goltsev, J. F. F. Mendes (2006). "K-core organization of complex networks". *Physical Review Letters* 96, 040601.

Fortuny, Jordi (2008). *The emergence of order in syntax*. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Gross, Jay & Yellen, Jonathan L. (1999). *Graph Theory and its applications*. Boca Raton, Fl.: CRC Press.

Hermes, Hans (1965). *Enumerability, Decidability, Computability*. New York: Springer.

Kayne, Richard S. (1994). *The antisymmetry of syntax*. Cambridge, Mass.: The MIT Press.

Kelley, John L. (1955). *General Topology*. Springer-Verlag.

Kuratowski, Casimir (1921). "Sur la notion de l'ordre dans la théorie des ensembles". *Fundamenta Mathematicae* 2: 161-171.

Partee, Barbara, Alice ter Meulen & Robert E. Wall (1990). *Mathematical Methods in Linguistics*. Dordrecht/Boston/London: Kluwer Academic Publishers.

Peskin, Michael E. & Daniel V. Shroeder (1995). *An Introduction to Quantum Field Theory (Frontiers in Physics)*. New York: HarperCollins Publishers.

Rodriguez-Caso, Carlos, Miguel A. Medina & Ricard V. Solé (2005). "Topology, tinkering and evolution of the human transcription factor network". *FEBS Journal* 272 (23): 6423-6434.

Zwart, Jan-Wouter (this volume). "Prospects for Top-Down Derivation".