Some formal considerations on the generation of hierarchically structured expressions[1]

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

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

Abstract

In this brief 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 the faculty 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 others domains than language where some computation or evolution is at work, which suggests the unifying force of the mathematical abstraction this note is based on.

Keywords: nest, theory of order, dominance, constituency, chain (of copies), rigorization.

1. Introduction

The intuitive development of an incipient theory is a necessary step in any rational enterprise. However, when the theory grows and increases its degree of complexity, 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.

[1] To appear in Catalan Journal of Linguistics.

The main objective of this short letter is to provide a rigorous definition of the basic syntactic algorithm and derive the basic syntactic relations.

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 and the dominance relationship.

2.1 Single nesting

Given a finite set $A = \{a_1, a_2, \ldots, a_n\}$ of singletons, 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 $a_i \in A$ comes into the computation at the step s_k, it becomes an occurrence a_i^k of a_i. This can be summarized through the recursive operation:

$$M_0 = a_i^0$$

$$M_{n+1} = a_k^{n+1} \cup M_n$$

where n is unboundedly large[2]. The outcome of the machine is the family[3]:

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

$\boldsymbol{N}$ is a family of sets linearly ordered by the inclusion relation[4]:

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

A family $\boldsymbol{F}$ of sets is a nest of a set S iff the elements of $\boldsymbol{F}$ are subsets of S and they are linearly ordered by inclusion. A family $\boldsymbol{F}$ of sets is saturated as to the property of being a nest with respect to S iff $\boldsymbol{F}$ 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 with respect to $\{g_1, g_2,.., g_{n-1}, g_n\}$, 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), a family of sets saturated as to being a nest of S is a linear order of S, and thus $\boldsymbol{F_1}$ (but not $\boldsymbol{F_2}$) can be rewriten as $<g_n, < g_{n-1}, <\ldots, <g_2, g_1>>>>$.

Observe that S 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 the outcome $\boldsymbol{N}$ of the nesting machine induces a linear ordering of $\cup\boldsymbol{N}$, the set of occurrences of the elements of the relevant alphabet.

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 $M_0 \subset M_1 \subset M_2 \subset \ldots \subset M_k$, and that (b) $\boldsymbol{C_k} \subseteq \boldsymbol{N}$. When $k = 0$, the constituent C_0 is a trivial nest, with only one element.

Given a nest $\boldsymbol{N}$ and an alphabet A, $M_k \in \boldsymbol{N}$ dominates B, noted as DOM(M_k, B), iff:

(a) $B \in \boldsymbol{N}$ and $B \subset M_k$, or

(b) $B \in A$ and $B \in M_k$.

---

[2] That n is unbounded does not mean that n is infinite. We refer the interested reader to Partee et alii. (1990: 69-70) for for a clear argumentation for the distinction between unbounded and infinite sets.

[3] 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 $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ … shall designate families, upper case non-bold letters A, B, C … sets whose elements are not all sets, and lower case letters a, b, c … primitive elements. 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.

[4] A relation R is a linear order iff it is a transitive, asymmetric and total.

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

The nesting machine itself ensures the existence of an M_k that is both the maximal and most element in the dominance relation. M_k is a maximal 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, and M_k is a most element iff there is no M_j that dominates M_k.[5]

Let the nesting machine to perform at a stage s_i the operation $X \cup Y$ in the case that X is an occurrence dominated by Y. We shall say, along with Chomsky (2001, 2005), that X is externally merged when it is selected from the alphabet and internally merged when it is selected from the domain of Y. X becomes an occurrence X_j when it is externally merged in 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, $\boldsymbol{CH}(X_j)$= $\{\{X_{j/k}, X_{j/i}, X_j\}, \{X_{j/i}, X_j\}\ \{X_j\}\}$. We shall call X_j the copy tail of $\boldsymbol{CH}(X_j)$, $X_{j/k}$ the copy head and any $X_{j/i}$ 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 ('$/_x$'). 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. Note that different concepts like chains or constituents can be naturally represented on the basis the same mathematical structure. This reinforces the concept of nest as a fundamental, unifying entity of structural relations.

2.2 Complex nesting

We want to allow the nesting machine to perform $X \cup Y$, when X and Y are two non-trivial nests, i.e., two non-singletons.

To enable our machine to perform such operations, we need to define several derivational spaces, which we will label as D_1, D_2, D_3… 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^j_k. Every derivational space involves a machine like the one defined in the above section, except that at a

[5] This requirement is equivalent to the Single Root Condition defined on tree structures.
The Single Root Condition (Partee et *al.* 1990: 441)
'In every well-formed constituent structure tree there is exactly one node that dominates every node'

given step s_i, the nesting machine can accept as input, not only $M_{i\text{-}1}{}^k$ and an element $a_t \in A$ but also the outcome of a derivation performed in a parallel space. We express that an element a_t has been inserted at the step s_k in the derivational space D_j as $a_t{}^{kj}$. Once D_{i} generates its outcome $\boldsymbol{N_i}$, it 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 from D_i to D_k where Di 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:

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

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). Since the object generated in D_j is a family of sets linearly ordered by inclusion, then there exists an M_t that is both maximal and most.

[6] If the number of derivational spaces were 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.

Figure 1

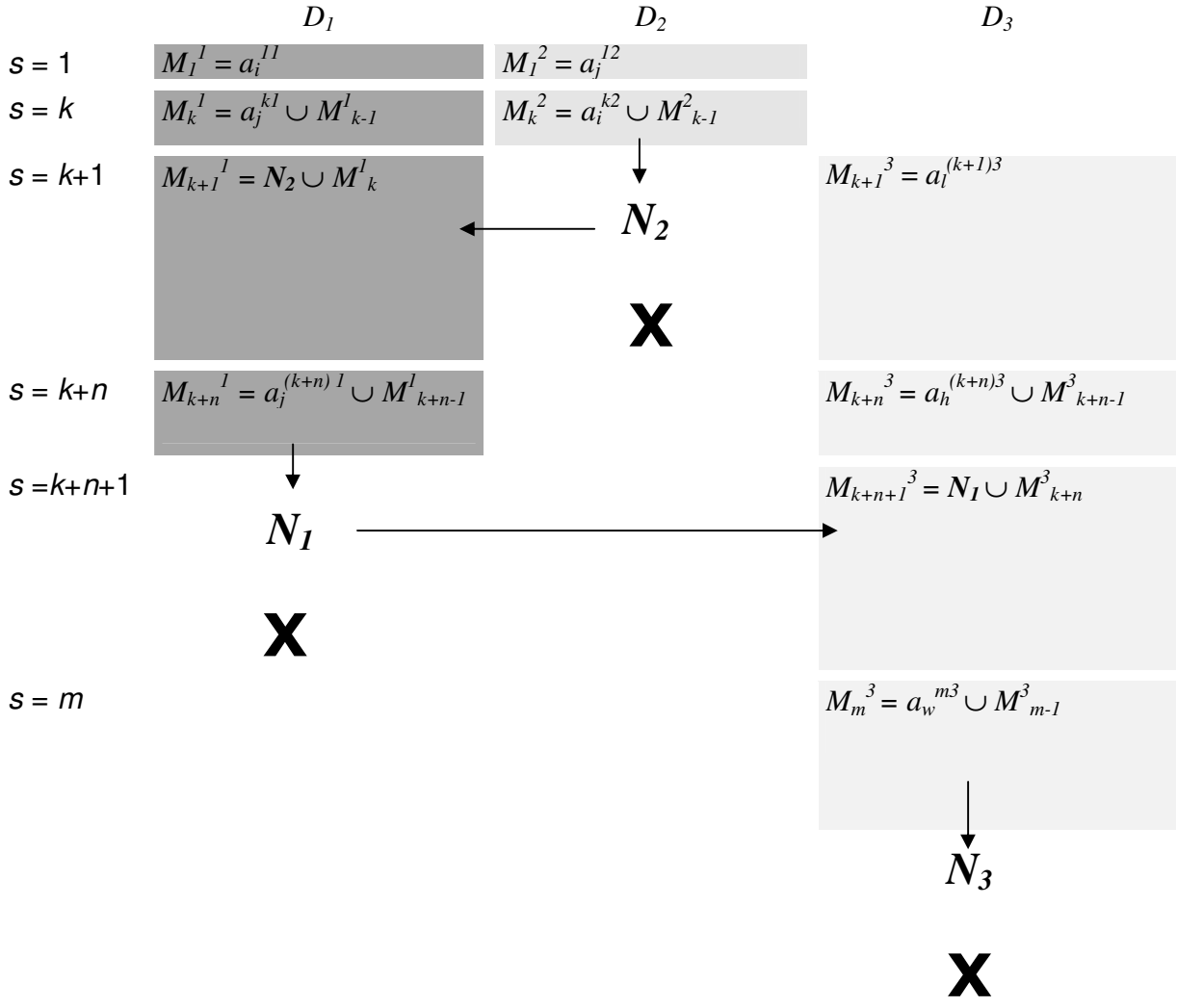

Figure 1. A picture of the application of the nesting algorithm over an arbitrary alphabet enabling complex nesting. In this example, three derivational spaces have been used and we assume that $\mu < 3$, i.e., only two derivational spaces can be opened at the same time. Note that, in the last step,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

It is necessary to define the dominance relations not only with respect to a sole derivational space (see 2.1), but also with respect to the multiple derivational spaces of a computation. Let $D_1,\ldots, D_n$ be all the derivational spaces used in the computation and $\boldsymbol{N}_1,.., \boldsymbol{N}_m$ their respective outcomes (one of them, the final outcome). Let the set M contain the members of all $\boldsymbol{N}_1,.., \boldsymbol{N}_m$:

$$M = \bigcup_{i \leq m} N_i$$

Thus, DOM(M_j^k, B) iff

(a) $[(B \subset M_j^k) \vee (B \in M_j^k)] \vee$

(b) $[\exists(n < \infty)\exists\{X_1,\ldots, X_n\}:((\forall i \leq n)(X_i \subset M) \wedge (B \subset X_1)) \wedge (X_1 \subset X_2 \subset \ldots \subset X_n \subset M_j^k)] \vee$

(c) $[\exists(n < \infty)\exists\{X_1,\ldots, X_n\}:((\forall i \leq n)(X_i \subset M) \wedge (B \in X_1)) \wedge (X_1 \subset X_2 \subset \ldots \subset X_n \subset M_j^k)]$

Condition (*a*) accounts for immediate dominance between two nodes ('$B \subset M_j^k$') and a node and a terminal ('$B \in M_j^k$') in arborean representations. Condition (*b*) accounts for non-immediate dominance between two nodes and condition (*c*) for non-immediate dominance between a node and a terminal.

3. Remarks and reflections

3.1 The nesting machine as an abstract entity

So far 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 any suitable syntactic theory. 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 $|F_0\rangle$[7] be the vacuum state, the state of a system where there is no any 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 $|F_1(\boldsymbol{k})\rangle$ as

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

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

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

$$|\boldsymbol{F}_{n+1}(\boldsymbol{k})\rangle = a^\dagger(\boldsymbol{k})\,|\boldsymbol{F}_n(\boldsymbol{k})\rangle$$

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. Obviously, this is just an informal 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 A = {{John}, {Mary}, {kisses}}, the nesting machine could proceed as follows:

s_1 = {John}

s_2 = {John, kisses}

s_3 = {John, kisses, Mary},

thereby generating the outcome $\boldsymbol{N}$, a nest saturated on {John, kisses, Mary}:

$\boldsymbol{N}$ = {{John}, {John, kisses}, {John, kisses, Mary}}.

We wonder whether $\boldsymbol{N}$ is the representation lying under the expression (*i*) 'John kisses Mary' or under the expression (*ii*) 'Mary kisses John'. If $\boldsymbol{N}$ underlies (*i*), then the nesting machine is a top-down procedure and $\boldsymbol{N}$ is read as a precedence relation by the A-P system. If $\boldsymbol{N}$ underlies (*ii*), then the nesting machine is a bottom-up procedure and $\boldsymbol{N}$ is read as a successor relation by the A-P system. We observe that if $\boldsymbol{N}$ lies under (*i*), we predict that the subject and the verb form a

[7] A quantum state is defined completely by a vector of a Hilbert space, and the standard notation is $|\psi\rangle$. This vector encodes all the physically relevant information of such state. In our case, we noted $|\boldsymbol{F}_n(\boldsymbol{k})\rangle$ as the state with n particles with momenta equal to $\boldsymbol{k}$.

constituent to the exclusion of the object, whereas if ***N*** lies under (*ii*), we predict that the object form a constituent with the verb to the exclusion of the subject. Since the later 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 right 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 a singleton subset of 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:

$$M_0 = A$$
$$M_1 = M_0 \setminus B_0$$
$$M_{n+1} = M_n \setminus B_n$$

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 the same situation than in section 2.1, *single nesting*:

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

To generate saturated nests (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 $|Y| > 1$. Again, we need to define a set of derivational spaces D, with the properties defined in section 2.2. If s_k in D_j forms the complementary subset of a non-singleton subset B of $M^j_{k-1,}$ then B becomes the initial symbol of a new derivational space D_i. As in section 2.2, all D is a nesting machine that generates saturated 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 a 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.

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 their constituency.

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

If both $\boldsymbol{N_1}$ and $\boldsymbol{N_2}$ were useful representations for the two sentences under discussion, the nesting machine would be no more than a semantically vacuous algorithm responsible for linearly ordering occurrences, but not for representing grammatical relations into constituents. Crucially, whereas the noun 'Peter' can be argued to be characteristically identified as the object of the verb 'visited' in $\boldsymbol{C_2}$, there is no way to warrant on configurational terms that 'who' is identified as the

object 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 an object of a verb.

One could raise the question of whether the algorithm responsible for generating hierarchically structured expression is precisely no more than a vacuous algorithm of linearizing occurrences, the grammatical relations among those occurrences being set at a different level, say at the C-I system. Though there is no reason to think that this possibility is not tenable, it seems to us that it would not lead to a better understanding of grammar, but simply to a reformulation of what we usually consider as part of syntax in terms of the syntax of the syntax-semantic interface. In this note we shall keep to the intuition that the combinatorial syntactic procedure is responsible for yielding not only order among terminals but also constituency and dominance relations, three relations that can be defined on nested families.

With the objective of providing the formal means of representing grammatical relations in configurational terms, we explicitly defined the nesting machine in such a way that it generated chains of multiple copies of a occurrence (section 2.1), thereby adopting the standard view in minimalistic syntax that an occurrence acquires multiple semantic features because it has been merged and 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 $\boldsymbol{CH_1}$ that comprises the multiple copies of the occurrence 'who'. We also indicate the constituent $\boldsymbol{C_2}$ where 'who' is identified as an object and the constituent $\boldsymbol{C_5}$ where it is identified as a *wh*-phrase.

D_2'

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

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

$\boldsymbol{CH_1} = \{\{\text{who}_1\}, \{\text{who}_1\text{, who}_{1/5}\}\}$

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

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. If, for instance, the complement *X* turns out to precede *H* in an utterance, this is because a further constituent $\boldsymbol{C_l} = \{\{X_i\}, \{X_i, H\}\}, \{X_i, H, \ldots\}, \{X_i, H, \ldots, X_{i/l}\}\}$ has been generated by internal merge, and the most deeply nested copy X_i of a chain $\boldsymbol{CH_i} = \{\{X_i\}, \{X_i, X_{i/l}\}\}$ has been deleted; similarly, if the specifier *Y* turns out to follow *H* this is attributed to *H* being internally merged in a position less deeply embedded than *Y* .[8]

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 canc 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 the faculty 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,

[8] 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. We also note that we do not postulate a checking (sub)theory in our theory of order; the procedure of the nesting machine is not claimed to be driven by the need to delete uninterpretable features, to put in standard minimalistic terms, or by the requirement of matching type features attributed to abstract heads and token features attributed to current categories, to put it in Fortuny's (2008) terms. The standard minimalist suicidal greed is argued to be problematic in Fortuny (2008), and Fortuny's alternative proposal lacks any explanatory power whatsoever, since the insertion of token features is triggered but not the insertion of type features, whose ordering is claimed to derive from the Full Interpretation condition; this is as rather vacuous way of saying that the ordering of current categories defined by the nesting algorithm must face the Full Interpretation condition.

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 constituency, dominance and chains 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 some kind of computation or evolutionary process is at work.

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 (2005). "On phases". Ms., MIT.

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

Corominas-Murtra, Bernat, Sergi Valverde, Carlos Rodríguez-Caso & Ricard V. Solé (2007b). "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.

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

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

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.