Beyond Trees

François Lamarche

Projet Calligramme
LORIA UMR n^o 7503 – INRIA
Campus Scientifique, B.P. 239
54506 Vandœuvre lès Nancy Cedex – France
lamarche@loria.fr

1 Introduction

For a long time the objects described by linguistic formalisms have been one-dimensional. Whether the grammar was about strings or trees, if one took an object that it generated, one could always extract from that object a total order, which would usually relate its minimal (or "terminal") constituents, saying that those constituents were ordered along along a left-right (or speech time) axis. But linguists have always been aware of another important axis, which we may call the surface-depth axis, and which may roughly be described by a sequence like phonetics — phonology — morphology —syntax—predicate semantics— deep semantics—discourse.

So the first linguistic formalisms were geared first and for all for describing a "horizontal slice" of a two-dimensional object, the great benefactor being the syntactic slice.

Later there was a move towards integrating two or more "parallel" slices. A lot of the research (but by no means all of it) has been concentrated on the pair syntax—predicate semantics. The literature is already enormous, and we can only give a very broad outline of the different philosophies. In many approaches, whose common ancestor is Montague semantics, it is posited that predicate semantics is a "homomorphic image" of syntax (in other words, that it does not include more *structural*, as opposed to typing, or categorial, information than syntax). We would put the "glue language" approach to LFG semantics in this class, as well as numerous refinements of Montague semantics.

Another class of formalisms generate a *single* large tree-like object, from which different "slices" (and here more cases have been covered than just the syntax—predicate semantics pair) can be extracted as substructures. In these formalisms the extraction process is not necessarily immediate, the embeddings not being trivial at all some times; also, different slices may be embedded in quite different ways. The formalisms that we would put in this class include HPSG, the approaches to TAG semantics considers derivation tree (as opposed to the tree generated) as embodying the semantics, and the the various formalizations of Chomsky's work, in particular the ones associated to the minimalist program.

This should be constrasted with a third important class of approaches that aim instead at "superposing", or "synchronizing" the different tree structures belonging to different slices, using various technical devices. This class definitely comprises the multimodal categorial approaches that deal with more than one representational slice, as well as the "synchronous TAG" approach to semantics.[1] The difference here is that the various slices have some amount of independent existence, that includes in particular their own sets of structural constraints, and that the result of an analysis is the integration of all these constraints. Also, the process of extracting a slice is uniform, having to to simply with the choice of a slice index. It is certainly the approach that best fits the slogan "analysis is the assembly of a jigsaw puzzle".

This paper addresse the question: what is the best way to describe the structure which is generated by a grammar of this third class? This being, naturally, a prior question to defining the grammar itself, and just as much a matter of mathematical formalism than of linguistic/coginitive realism.

We claim that a superposition of different tree slices is best described as a single structure, that has tree-like properties, but

[1] The final version of this paper will contain a decent bibliography. Given the little time we had to prepare this abstract, we would rather not quote at all than quote badly.

that "branches in more than one dimensions at once", so the jigsaw puzzle is a multi-dimensional one. Consider the case of a transitive verb. The active-passive duality can be summarized by the following distribution of roles:

act	synt	sem
	subject	agent
	object	patient

pas	synt	sem
	subject	patient
	fac adj	agent

We think that the lexical entry of such a verb should have all these four[2] roles present, but as independent branches of a *single* node (that has potentially more branches). Fitting this entry inside an analysis would entail relating it with other atoms in ways more general than the ordinary way nodes of a tree are related (immediate dominance, etc). This atom would "straddle" both the syntactical and the semantical slice, in such a way that its roles would be distributed correctly in the end analysis.

Let us extend this example: suppose now that the verb is used in the active voice, but that one of its syntactical roles (say the object) is filled by a pronoun, and that the analysis is complete enough so that this pronoun has found its referent. Obviously, one of the pronoun's functions is to transfer the verb's `agent` role to that referent. So we have an interaction between three components, which can be seen for simplicity as atomic nodes: the verb, the pronoun, and the referent. To formalize that analysis, we need the pronoun's reach to extend in the semantic slice[3], in order for it to catch its referent, and to interact in the semantical slice with the verb's `patient` role, in a way which is geometrically quite different from what happens at the syntactical level.

In this paper we present the basic outlines of a formalism that can construct such generalized trees, and other even less familiar structures. The presentation will be closer to mathematics than to computer science or linguistics. One reason is that the mathematical formalism has to be developed, and the correct definitions be given, before efficient implementation strategies and complex linguistic examples are given. So this preliminary paper is best seen as the announcement of a research program.

So in our formalism a full linguistic analysis of say, a paragraph, should end up as is a single structure, but on which can be "sliced" in many different ways, each slice providing a tree, of a structure very close to one. In general, as one goes towards the deep end the structures get larger and larger, so the deeper tree slices look roughly like quotients of more surface ones[4]. But this is only partially true. Sometimes the branching of different slices truly *disagree*, as the pronoun example above shows. Also, consider the age-old problem "does the verb bind closer to the subject or to the object?". From a practical point of view, a good answer is: at the syntactical level it is the object which is closer to the verb (as is generally agreed), but at the morphological level it is the subject, since it has to agree with the verb. The agreement cues being useful for analysis, we contend that a good formal linguistic representation (from the point of view of computational linguistics at least) of a sentence has an advantage if it can slice differently at the morphological level and at the syntactical one.

So far this looks like a two-dimensional structure, but nothing limits the number of dimensions. And once it is available, multi-dimensionality can be put to good linguistic uses. Consider for example the semantics of time: a generalized tree can include a time dimension, where semantical events are related with respect to their time of occurence, and the time of their enunciation. This time dimenson is not related in any predictable way with the speech-time one, or the surface-depth one. So it is better seen as a new dimension.

So we may be seen as advocating a completely integrated approach to linguistic representation: a single, very general way of combining atomic objects that allows every piece of information that can be extracted from a linguistic analysis to be assembled in a single unified formal representation. One immediate advantage of such an approach is the inherent

[2] A standard linguistic analysis would say that in the passive voice, the object role has been killed, and that the facultative adjoint role is not given out by the verb, but actually comes from outside of it; so in the table above it is better to replace that entry by a blank.

[3] exactly at which representation level issues of reference is not important here

[4] and as is well-known, the speech-time dimension has very little importance for slices deeper than syntax

robustness due to the redundancies present in a unified representation. The basic ingredients are atomic objects of various arities, which can be thought of as basic nodes for trees, or function or predicate (e.g., category) symbols. But we show that these atomic symbols can be assembled in very many more ways than expected, by the means of a large set of basic combination rules.

An interesting aspect of this approach is that it is naturally compatible with issues of underspecification. A recent and important area of research in computational linguistic is the development of grammars that manipulate not trees, but tree specifications — that is, trees that are not entirely defined as to their structure and contents. And combination rules can be more or less strict, and this introduces a natural order structure, which has to do with the constraints the objects are subject to. Objects that are maximal with this order as those that are fully defined.

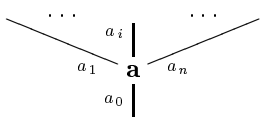
So the formalism has an inherent "information ordering", of the kind used in domain theory [1]. Part of the work that lies ahead is making the exact connection with traditional domain theory, which some authors feel should have growing importance in the field of natural language processing.

2 Combining Atoms Intuitively

Let us start with a slogan to summarize things: *Symbols simultaneously define and occupy a certain kind of space.* Naturally we have to give a useable formalization of this.

Definition 1. An *atom type* $\mathbf{a}$ of arity n is defined by a set $\widetilde{\mathbf{a}} = \{a_0, a_1, \ldots, a_n\}$, whose elements are called its *roles*.

This can be pictured in an obvious way:



So at first glance each role a_i can be thought of as a name for a *port*, a way of connecting a symbol with others... but we will see that this is only part of the picture. Notice that because of the zeroth port a_0 a symbol $\mathbf{a}$ of arity $n = \mathrm{Ar}(\mathbf{a})$ actually has $n+1$ ports. We say that it has *valence* $n+1$, so we always have $\mathrm{Val}(\mathbf{a}) = \mathrm{Ar}(\mathbf{a}) + 1$.

We use the term "role", but naturally other people would call them "functions". We emphasize that it is a very general notion, that covers the more standard cases, like semantical and syntactical role, as well as things like before/after at the speech-time slice, or, before/after at the semantical-time slice.

Another way of thinking of roles is as fields for record data structures, but structures that have an additional field for the querying process[5].

An atom should be thought of as a unit of structured information. Indeed, the existing work we know about which we find closest to our philosophy is Barwise and Perry's situation theory [2], developed for natural language semantics. There, a unit of information is called an *infon*, and corresponds roughly to our atoms. Infons are combined in a rather different way than we do here.

It will be useful to consider atom types, and their occurences (i.e., atoms) as linear (degree one) polynomials:

$$\mathbf{a} = a_0 + a_1 + \cdots + a_n \tag{1}$$

This is why we make a notational difference between $\widetilde{\mathbf{a}}$ (considered as a set) and $\mathbf{a}$ (considered as a polynomial).

So we assume that we are given a set $\mathcal{T}$ of atomic types; in this paper we will always assume that different types have disjoint roles, in other words that $\mathbf{a}, \mathbf{b} \in \mathcal{T}, \mathbf{a} \neq \mathbf{b}$ implies $\widetilde{\mathbf{a}} \cap \widetilde{\mathbf{b}} = \varnothing$.

In practice symbols and symbol types will be denoted in a way that tries to avoid orgies of indices. For illustration let us first give ourselves a small stock of atom types:

$$\begin{aligned} \alpha &= \mathsf{s}_\alpha + l + r \\ \beta &= \mathsf{s}_\beta + u + d \\ \mathbf{m} &= \mathsf{s}_{\mathbf{m}} + m \\ \mathsf{B} &= \mathsf{s}_{\mathsf{B}} + b \end{aligned}$$

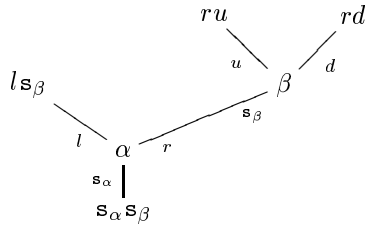
[5] Although none of the techniques of linear logic is used in this paper, its spirit it present everywhere, because of J.-Y. Girard's slogan "a formula is a space", and the idea of reciprocity, embodied by linear negation, which shows that the difference between input and output is relative, not absolute. This idea of reciprocity is represented here by the that extra (stem) record field a_0.

One important convention is that given an atom type $\mathbf{a}$ we often denote its zeroth role a_0 by $\mathtt{s}_{\mathbf{a}}$. We call it the *stem* role. It represents the "business end" of the symbol, the "channel through which the information given by the symbol comes out". For example if $\mathbf{a}$ is meant to be a function/operation symbol, its stem represents the result of the operation. This special notation for stems allows them to be more easily identified than the other roles. The index denoting the symbol type can be dropped when confusion is impossible: $\alpha = l + r + \mathtt{s}$.

The point of our approach is that symbols can be combined in many more ways than is usual, but let us start by showing how we represent the most standard way there is, namely plugging two symbols by wiring one port to another:

Example 2. if we take one occurence of α and one of β, and decide to connect r to $\mathtt{s}_\beta$, we find that a good informal way of representing it is as a *homogeneous*[6] *quadratic polynomial*

$$\alpha{-}r{:}\mathtt{s}_\beta{-}\beta \quad = \quad l\mathtt{s}_\beta + ru + rd + \mathtt{s}_\alpha\mathtt{s}_\beta \ . \quad (2)$$



So we imagine that *roles are broadcast on their port wires away from the symbol that they originate from. When they "enter" a symbol via a port they come out through all the other ports of that symbol.*

So a combination of k symbols creates a new set of roles, which can be thought of as monomials, all of power k; the role that decorates a free wire/port of a combination is the product of all the atomic roles that projected away from their respective symbols. In particular this means that an atomic role (like r here) projected by a symbol can be *split* by another symbol, resulting in its representation in more than one role of the final combination. For example the unique occurence of r here results in $ru + rd$; it has been split in two.

[6] i.e., all monomials have the same degree.

It is not really correct after all to consider roles as names for "ports", or "wires". The "stuff" these wires is made of is a kind of space, which can be split ad libidum. There is a "symbolic space", where the atoms live, and there is a "role space", which is split in different ways by the atoms that inhabit the atom space

As a matter of fact, the role space should and will be formalized as a Boolean algebra; in the example above this algebra is the set of subsets of the set $\{l\mathtt{s}_\beta, ru, rd, \mathtt{s}_\alpha\mathtt{s}_\beta\}$. The stem of an atomic symbol symbol occurence should be seen as the Boolean complement of the role space that the symbol controls, or owns (it may have to share this piece of role space with other symbols). For example here symbol β controls $ru + rd$ and for it the outside world is $l\mathtt{s}_\beta + \mathtt{s}_\alpha\mathtt{s}_\beta\}$.

What we mean by "polynomial" has to be defined carefully, though, because problems arise when two symbols of the same type are combined. A useful technique is to use positional notation in the monomials. If we construct

$$\alpha{-}r{:}\mathtt{s}_\alpha{-}\alpha = l\mathtt{s}_\alpha + rl + rr + \mathtt{s}_\alpha\mathtt{s}_\alpha \quad (3)$$

this is to be read with the understanding that in every monomial uv at the right of (3), the atomic role u has been sent by (belongs to) the first occurence of α to the left of the equation, and the atomic role v belongs to the right occurence of α. But, for this paper at least, this will only be a notational trick, and we have to be careful. The idea of using a non-commutative semiring of polynomial to represent combinations of symbols is very appealing, but gives rise to technical problems due to the need to express what happens when the order of symbols is permuted. In reality, what is meant is that (in this paper at least) both atoms should be distinguished by different indices, e.g.,

$$\alpha_1{-}r_1{:}\mathtt{s}_{\alpha_2}{-}\alpha_2 \quad = \quad l_1\mathtt{s}_{\alpha_2} + r_1l_2 + r_1r_2 + \mathtt{s}_{\alpha_1}\mathtt{s}_{\alpha_2} \ ,$$

which is heavier, but will allow a completely airtight formalization, as we will see. Naturally, if the set of indices is of the form $\{1, 2, \ldots, k\}$, they can be dropped at the right of an equation if we assume that everything is written in order, and positional notation can be used, which simplifies life.

So in general, given two of our polynomials $\mathbf{A} = A_0 + A_1 + \cdots + A_n$ and $\mathbf{B} = B_0 +$

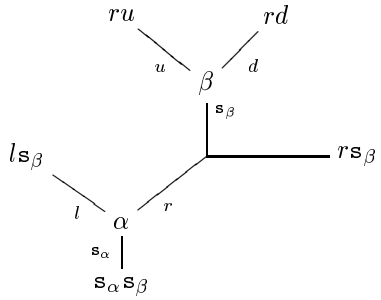
$B_1 + \cdots B_m$ we can define (positional notation, and subtraction used only as notational aid): $\mathbf{A} -_{A_i:B_j} - \mathbf{B} = (\mathbf{A}B_j - A_iB_j) + (A_i\mathbf{B} - A_iB_j)$.

Given the particular status given to the stem role, in any combination of atoms we will always want the monomial which is the product of all the atomic stems to appear; it will be the combination's stem. So for example the combination $\mathbf{A} -_{A_0:B_0} - \mathbf{B}$ should be considered illegal.

Now that we have got rid of these bureaucratic details, let us give examples that show the power of our approach.

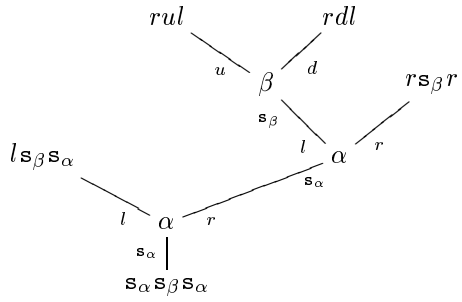
Example 3. Another way of combining α, β is the following, which adds one monomial:

$$\alpha -_{r|\mathtt{s}_\beta} - \beta \quad = \quad l\mathtt{s}_\beta + ru + rd + r\mathtt{s}_\beta + \mathtt{s}_\alpha\mathtt{s}_\beta \ .$$



The atoms have been combined in such a way that a new "port" has appeared, which is completely determined by the combination. It is perfectly legal to think of $\alpha -_{r|\mathtt{s}_\beta} - \beta$ as a legitimate combination by itself; but this depends on the grammar to which the combination rule belongs. It may also be that the grammar has been defined in such a way that $\alpha -_{r|\mathtt{s}_\beta} - \beta$ cannot appear "free" and that it has to be a subcombination of a larger one, for example

$$\alpha_1 -_{r:\mathtt{s}_{\beta_2}\mathtt{s}_{\alpha_3}} - (\beta_2 -_{\mathtt{s}_\beta:l} - \alpha_3) \quad = \\ l\mathtt{ss} + rul + rdl + r\mathtt{s}r + \mathtt{sss}$$



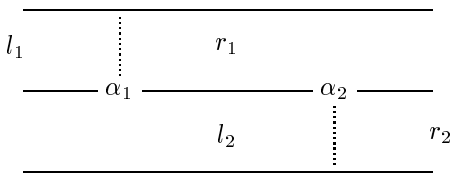
Notice that at the right of the equation we have dropped the indices that associate stems with their atoms, as well as the indices that distinguish occurences of atoms; all this is allowed by positional notation. Naturally there is another way to describe the same combination, by associating to the left: $(\alpha_1 -_{r:\mathtt{s}_\alpha} - \alpha_3) -_{rl:\mathtt{s}_\beta} - \beta_3$, and there is a third, more informal way (to be formalized) that drops the parentheses: $\alpha_1 -_{r:\mathtt{s}_\alpha} - \alpha_3 -_{l:\mathtt{s}_\beta} - \beta_2$.

So in general, given two polynomials $\mathbf{A} = A_0 + A_1 + \cdots + A_n$ and $\mathbf{B} = B_0 + B_1 + \cdots B_m$ we can define $\mathbf{A} -_{A_i|B_j} - \mathbf{B} = \mathbf{A}B_j + A_i\mathbf{B} - A_iB_j$.

Example 4. Another way of combining atoms of valence three is as follows

$$\alpha_1 \lhd \alpha_2 = ll + rl + rr + \mathtt{ss} \ . \qquad (4)$$

Something new happens here: simultaneously the first occurence gets one role (its r) split by the second occurence: $rl + rr$, while the second occurence gets its l-role split by the first one: $ll + rl$. The symbols are combined in a tighter way than is the case in ordinary syntax. But this combination makes perfect intuitive sense: think of α as a symbol which is used to separate a line in two. This line could be time, and α would be an punctual event, like a state transition, for example, or it could be a word in a free monoid, in which case α would be the concatenation operator. An observer looking up the stem of α sees the whole of that linear space, and sees that it is split in two: a left l-part and a right r-part. So if our observer looks up the stem of $\alpha \lhd \alpha$, what is seen is a line which is split in three, by two occurences of the separator; the defining equation means that separation α_1 is to the left of separation α_2 in that space (in the following picture, one should think that the stem role is behind the observer's head.



So this formalism allows us to deal with strings as well as trees, if we want to.

Example 5. Also, two symbols can be combined in an orthogonal fashion:

$$\alpha \diamond \beta = lu + ld + ru + rd + \mathtt{ss} \ . \qquad (5)$$

If we think of β as an up/down separator, then this combination creates a "square", two-dimensional object. So this is the basic way to create a structure that includes both left-right, and surface-depth axes as mentioned in the introduction. Another natural use for this way of combining symbols is to create information databases: given a set of symbols that each represents some (let us say binary) means of discrimination, like cold/hot, married/single, extrovert/introvert... then their orthogonal combination is a way of creating a classifying space, with locations for putting objects according to the features [7] (seen as atomic roles) they possess. Here roles are simply a means of using record notation.

In general, we can always combine two atoms (or polynomials for that matter) orthogonally (subtraction poses no problem): $\mathbf{A} \diamond \mathbf{B} = \mathbf{AB} - A_0\mathbf{B} - \mathbf{A}B_0 + A_0B_0$.

Example 6. Another way of combinining symbols is by superposition:

$$l\mathtt{u} + r\mathtt{d} + \mathtt{ss} \ .$$

Here the combination of symbols does not add any structural information to that given by the individual components: the resulting space's structure is not a refinement of its constituents, but only "role isomorphic". But it is a pretty common situation, which means we are not adding structure, only information. For example, if $\mathbf{m}$ is an atom that represents a marble, and B the color blue, then a blue marble can be written $m\mathsf{b} + \mathtt{s_m s_B}$. Complex objects can be constructed by mixing combination rules. If B is now thought of as a bag, then "two marbles in a bag" can be written $m\mathtt{s_m}\mathsf{B} + \mathtt{s_m}m\mathsf{B} + \mathtt{s_m s_m s_B}$ (they fit snugly), or $m\mathtt{s_m}\mathsf{B} + \mathtt{s_m}m\mathsf{B} + \mathtt{s_m s_m}\mathsf{B} + \mathtt{s_m s_m s_B}$ (there's room left in the bag).

Example 7. Let α be thought of as a binary operator $(-)\cdot(-)$, with the l role representing its left argument. So $\alpha - r{:}\mathtt{s}_\alpha - \alpha = l\mathtt{s}_\alpha + rl + rr + \mathtt{s}_\alpha\mathtt{s}_\alpha$ represents the term $x_1 \cdot (x_2 \cdot x_3)$, *without the need to choose variables*, since the roles determine their positions. Let β play the part of a binding operator. The role u will apply to the bound locations, and the role d to the ones that the binder leaves free. So for instance a term like $(\lambda x)(x \cdot (x_2 \cdot x))$ is represented by $l\mathtt{s}_\alpha u + rld + rru + \mathtt{s}_\alpha\mathtt{s}_\alpha\mathtt{s}_\beta$.

[7] Here we are using the word feature in a very broad sense. But naturally it raises an unavoidable question: how conventional feature structures and feature logic are best implemented in our formalism, which we think is powerful enough to incorporate them naturally.

3 Formalizing Things

We do not have the space here to detail the mathematical formalism that makes the polynomial approach absolutely rigorous. We will just say that the basic approaches uses tensors of Boolean algebras and sets of quotients of these.

References

1. S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Clarendon Press, 1994.
2. J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, 1983.