Some formal details

Greg Kobele

April 10, 2018

1 Syntactic structures as MDSs

A tree is either a leaf or a root node with some number of daughter subtrees. It is *headed* if and only if (iff) every node has a head, which is either 1. itself, if it is a leaf, or 2. the head of one of its daughters, otherwise . Since the structures we are concerned with are binary branching (each internal node has exactly two daughters), we indicate that the head of a node is the head of its left (right) daughter by giving it the label < (>). This label is intended as a mnemonic which *points* to the head.

A Multiple Dominance Structure (MDS) (aka (rooted) Directed Acyclic Graph (DAG)) is a tree where nodes may have multiple mothers. Interpreting each branch as a *dependency*, this allows an expression to be dependent on multiple others. This is intended to be an uncontroversial theory-independent claim; *clearly* in a sentence like the below, we want to say that *John* is dependent on (i.e. is semantically related to) the verb *laugh*, and that it is dependent on (related to) the matrix verb *seems*, as witnessed by the overt expression of agreement:

John seems to laugh loudly.

1.1 how does this relate to the trees with traces we know and love?

This is more general. We can get the familiar 'trace'-full trees by making (all but one) edges to a multiply dominated node point instead at a trace. However, the same kind of grammatical information is present.

the 'point' of movement is to allow a single expression to be in multiple places 'at once'. Multiple dominance structures encode this information in an arguably simpler way.

Using MDSs over trees with coindexed nodes, or coindexed copies, is **not** a substantive claim about anything [Kracht, 2001]!!! This is just a way of presenting information (about syntactic dependencies) that I think is convenient.

1.2 how do we pronounce these?

The provisional answer:

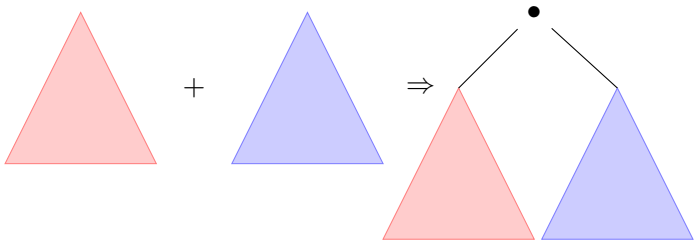
turn it into a tree first, and then pronounce it as usual

We can turn an MDS into a tree in lots of ways. The one we will adopt at the moment is to simply remove all branches into nodes except the 'highest'. Other answers have been proposed in the literature, and we will encounter some of them later on.[1]

2 Describing sets of MDSs

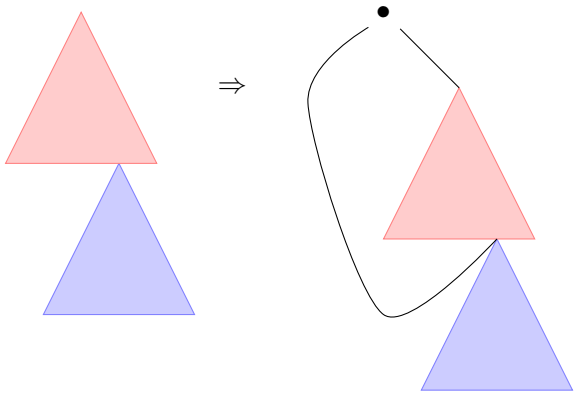
In syntax we are interested in assigning structures to strings. (This structure should represent in some way some interesting property of the string, such as its meaning.) Therefore, to a first approximation, a language consists of a set of MDSs (the structures we want to assign to its expressions). Usually, we want to say that languages have potentially infinitely many well-formed/meaningful expressions (if S is a sentence, so is *I believe that* S). One way to describe an infinite set of MDSs is by specifying how to construct larger MDSs from smaller ones. In the evolution of transformational grammar to minimalism, the vast array of transformations have given way to just two basic operations.

1. Merge / External Merge



2. Move / Internal Merge

[1] A wide variety is given in Kobele [2006].



Observe that, architecturally, **merge** and **move** have something in common:

they both introduce a new 'root' node, and two branches from the root. One of these branches always goes to the root of one of the inputs to the function.

In the case of **merge**, the other branch from the root goes to the root of the other input. In the case of **move**, the other branch from the root goes to the root of some maximal projection contained within the input. Thus, we might think of there as being just a single operation, which takes two arguments, with two cases:

case 1 its second argument is part of its first (i.e. it's *internal* to the first argument)

case 2 its second argument is not part of its first (i.e. it's *external* to the first argument)

Many people seem to think that this is a **BIG DEAL**. I am not convinced.

2.1 Overgeneration

By themselves, **merge** and **move** wildly overgenerate. we want to control them, to rein them in. This is the same problem faced by grammar formalisms of all stripes, and our solution is every-man's solution:

operations are controlled by the categories of the expressions they apply to

In many traditions, categories are viewed as *structured objects*. We will call a category a *feature bundle*, and think of it as consisting of a list of *features*. Features come in two different kinds :

1. those relevant for **merge**

   We write these as '`=x`', '`x=`', and '`x`'

2. those relevant for **move**

   We write these as '`+x`' and '`-x`'

. . . and have two different polarities

1. those which are 'active' (endocentric)

   These are '`=x`', '`x=`', and '`+x`'

2. those which are 'passive' (exocentric)

   These are '`x`' and '`-x`'

We will view features as *resources*, which are are consumed at each derivational step. We indicate this by getting rid of them (as opposed to marking them as 'consumed' or 'checked').

A prominent research question in modern syntax asks whether we explain syntactic features in terms of something else. In other words:

> can we eliminate the need to specify the syntactic feature bundle of an expression because we can *derive* it from its semantic, morphological, etc properties?

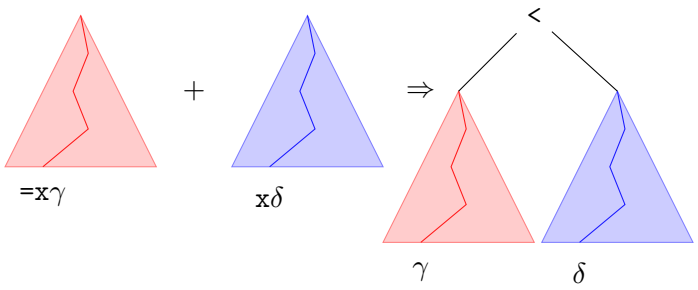
This is a *reductionist* question, which is, while perhaps important, somewhat orthogonal to the goal of specifying a set of MDSs; it doesn't matter whether you specify the set by using a particular bunch of feature bundles, or whether you specify the same set by first deriving these feature bundles from something more principled. At any rate, I don't know what the answer to the above question is, and won't worry about it too much yet.

2.2 merge and move, take two

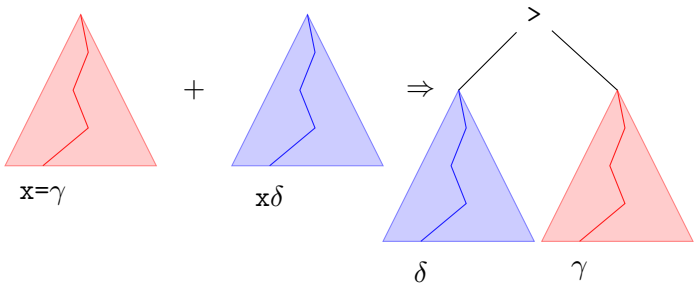
In the below, I indicate the position of the *head* of an expression by pointing to the subtree it is contained in (`>` points to the right, and `<` to the left). Furthermore, a triangle with a squiggily line running down from the top to somewhere on its base, under which is a feature bundle, indicates an MDS whose head (the leaf that is ultimately pointed to by the arrow at the root) has these features.

1. Merge / External Merge

- on the right

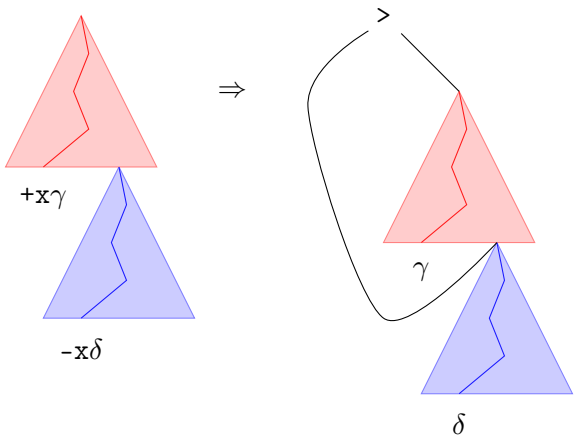


- on the left



One popular idea is that there is only rightward (or leftward) merge, and apparent exceptions are the result of movement. The most famous proponent of this idea is Richard Kayne [Kayne, 1994]. Kayne is actually famous for (among many other things) popularizing an idea about how to avoid directly stipulating this. It turns out that there is no set of sentences you can describe with both right- and leftward merge than you can't already describe with just rightward or with just leftward (which are themselves descriptively equivalent).

2. Move / Internal Merge



We could easily add rightward movement (like we added rightward merger), and then I would here have written that "one popular idea is that there is only leftward move."

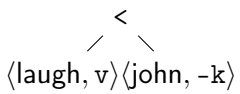
2.3 Lexica

A lexicon is a set of atoms paired with feature bundles.[2] Given **merge** and **move**, we can *completely* specify a language (qua set of MDSs) by giving a lexicon:

$$\langle \mathsf{john}, \texttt{d -k} \rangle \quad \langle \mathsf{laugh}, \texttt{=d v} \rangle$$
$$\langle \mathsf{will}, \texttt{=v +k t} \rangle$$

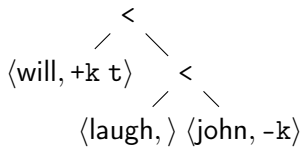
The expressions *generated by* a lexicon are those which can be built up from lexical items by a finite number of applications of **merge** and **move**. Given the lexicon above, we can generate (in addition to the lexical items themselves) all and only the following expressions :

1. **merge**$(\langle \mathsf{laugh}, \texttt{=d v} \rangle, \langle \mathsf{john}, \texttt{d -k} \rangle)$:

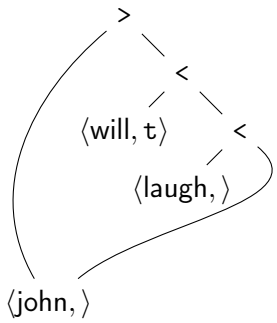


2. **merge**$(\langle \mathsf{will}, \texttt{=v +k t} \rangle, 1)$:

[2] An atom is whatever you want it to be. For our current purposes, you can think of it as a word, or a morpheme.



3. **move**(2) :



2.4 Which MDSs?

Viewing features as outstanding requirements ('I need a DP to be complete', 'I need case to feel good about myself', etc), it makes sense to think that the MDSs which have no outstanding features (other than their category (x) feature) are special ('complete'). The 'XPs' generated by a lexicon are all the MDSs which have just the category x at their heads which can be generated by the lexicon

References

R. Kayne. *The Antisymmetry of Syntax*. MIT Press, Cambridge, Massachusetts, 1994.

G. M. Kobele. *Generating Copies: An investigation into structural identity in language and grammar*. PhD thesis, University of California, Los Angeles, 2006.

M. Kracht. Syntax in chains. *Linguistics and Philosophy*, 24(4):467–529, 2001.