Musical syntax: space-to-tree algorithms

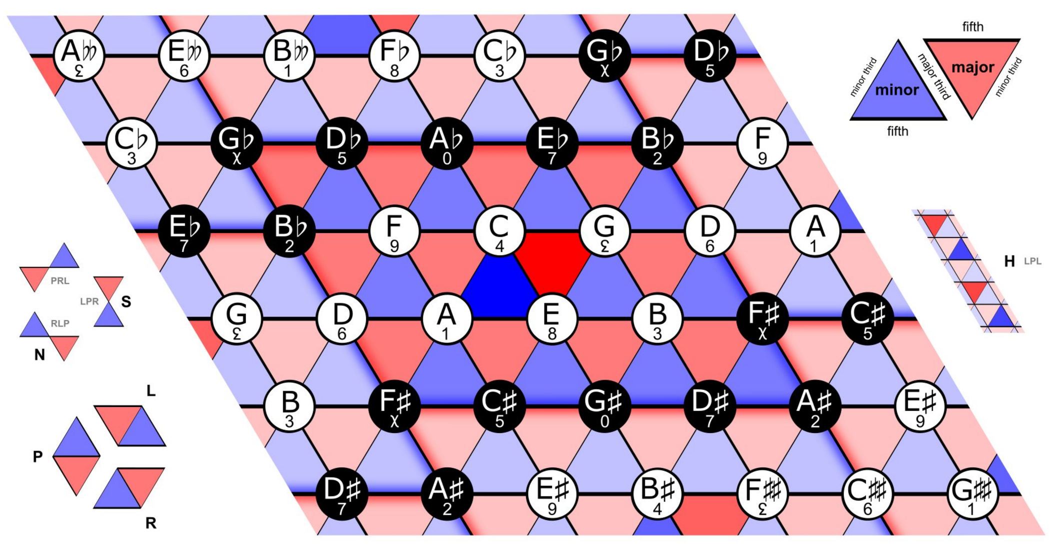
Assaf Brown[1], Andreas Haida[2]

[1]The Department of Cognitive Science, [2]ELSC

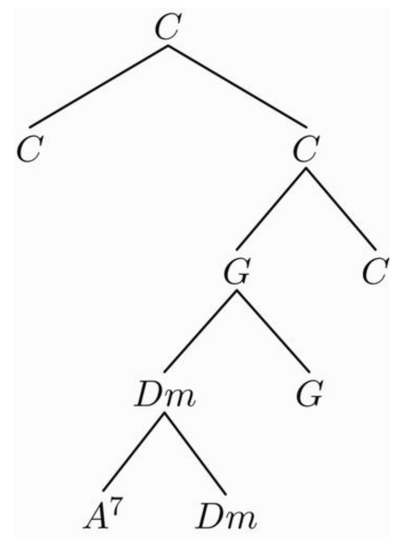
Abstract

How can we describe the behaviour of **harmony**? There are two main paths:

Musical spaces, in which musical objects are located in an abstract space and the distance between them can be measured



Syntactic trees, in which the hierarchy of a musical sequence is determined according to a system of rules



Following earlier suggestions (Grosjean et al, 1979; Lerdahl, 2004), we propose a family of **algorithms that associate these two representations**. Our results suggest that tree representations and spatial ones are different perspectives on the same cognitive reality.

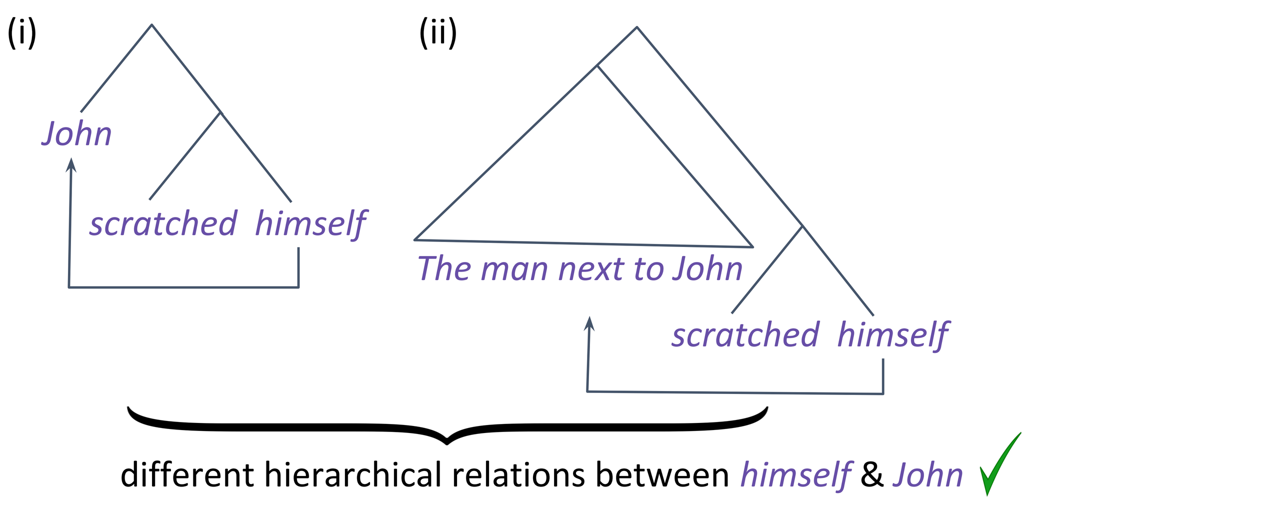
Hierarchical structure in language and harmony

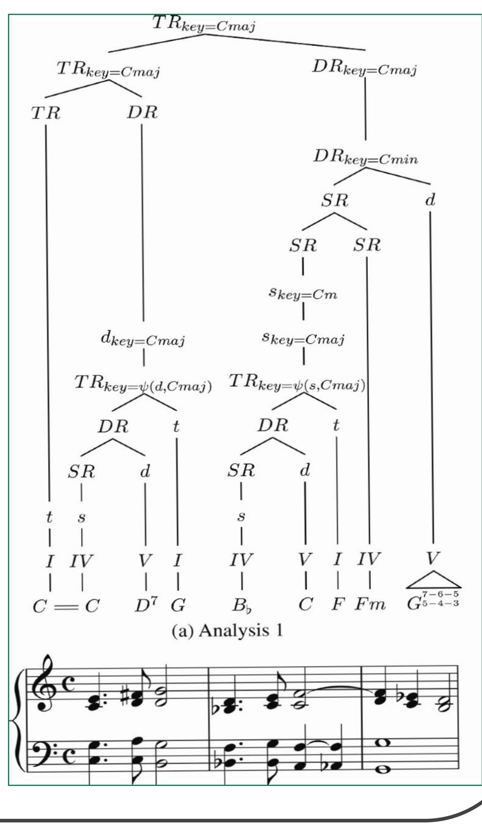
Complex expressions = strings of words/chords or hierarchies of phrases built from words/chords?

(i) *The man next to John scratched himself*

(ii) *John scratched himself*

same linear relation in (i) & (ii) ⇒ both express *himself* = John





Building hierarchical structures from strings

Basic idea: **similar neighbors merge** before dissimilar neighbors.

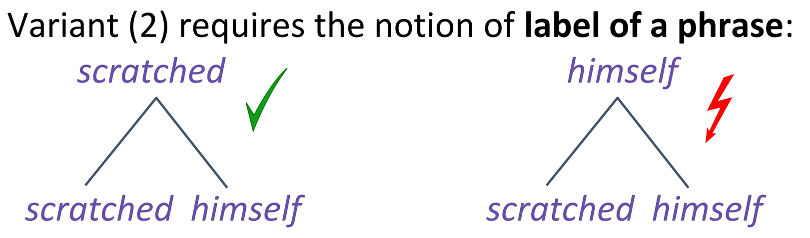
Two crucial notions: (D) **distance** between objects (similarity metric) & (N) **neighborhood** of an object

(N): An object's predecessor and successor in either

1. the string (atomic objects) or
2. the resulting hierarchical structure (atoms or phrases).

We explore variant (2).

Variant (2) requires the notion of **label of a phrase**:



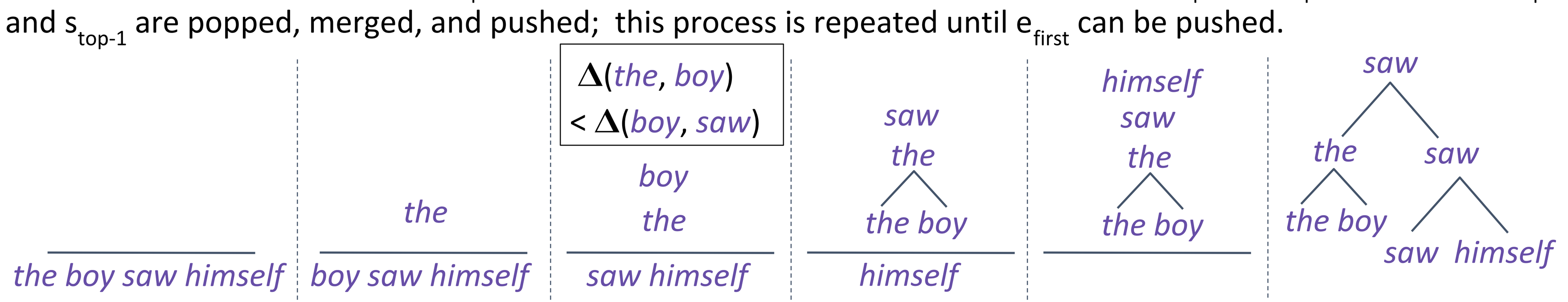
Our **online parser** (with a running time of O(n · log n)) operates on two data structures:

I. the input string: $e_1 e_2 e_3 \ldots$

II. a (memory) stack

The parser makes us of a **distance function** $\Delta(e_i, e_j)$ that determines if objects are merged or pushed:

e_{first} is pushed iff its distance to s_{top} is smaller than or equal to the distance between s_{top} and s_{top-1}, otherwise s_{top} and s_{top-1} are popped, merged, and pushed; this process is repeated until e_{first} can be pushed.

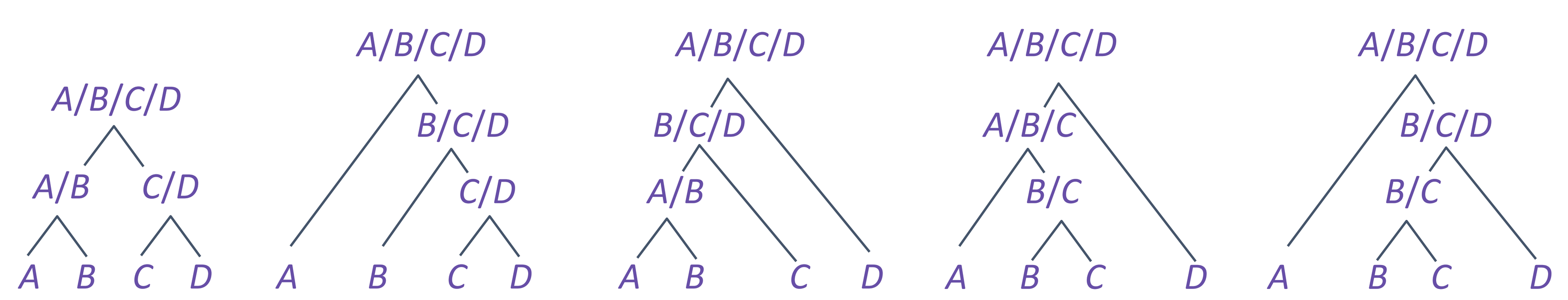


Our online parser with **dynamic labeling**:

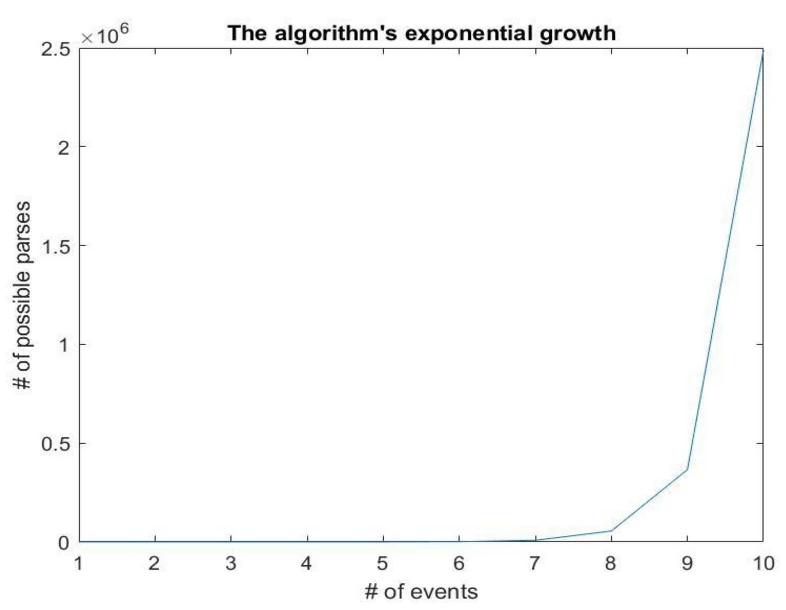
The label of $[s_{top-1}\ s_{top}]$ is chosen as to **minimize the distance** between the label and e_{first} (Lerdahl labeling).

Global optimization parser:

Labels and hierarchies are chosen as to **minimize the sum of all distances** in the resulting tree. To find the optimal solution is probably a NP hard problem. Here's the search space for a string of 4 events:



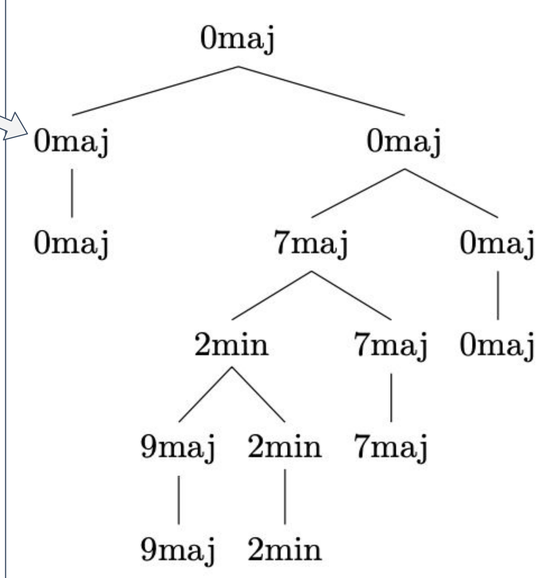
For the time being, we use a brute-force algorithm. The algorithm builds all possible trees and labelings and picks among those the optimal one(s). The graph on the right shows the exponential growth of the algorithm.

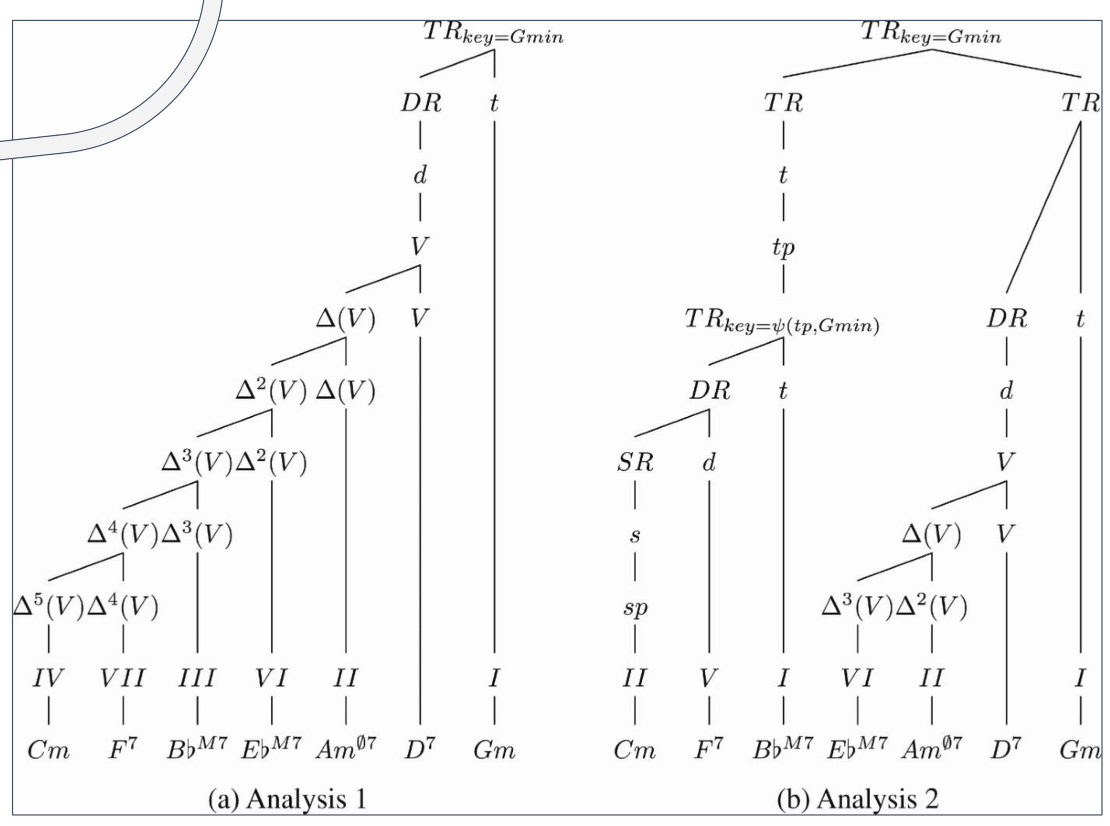


Empirical support and discussion

The output of our family of algorithms agrees to various degrees with annotated trees proposed in the literature.

The tree to the right is generated both by the online parser with dynamic labeling and the global optimization parser as the optimal parse. It corresponds exactly to an example by Rohrmeier shown on this poster.

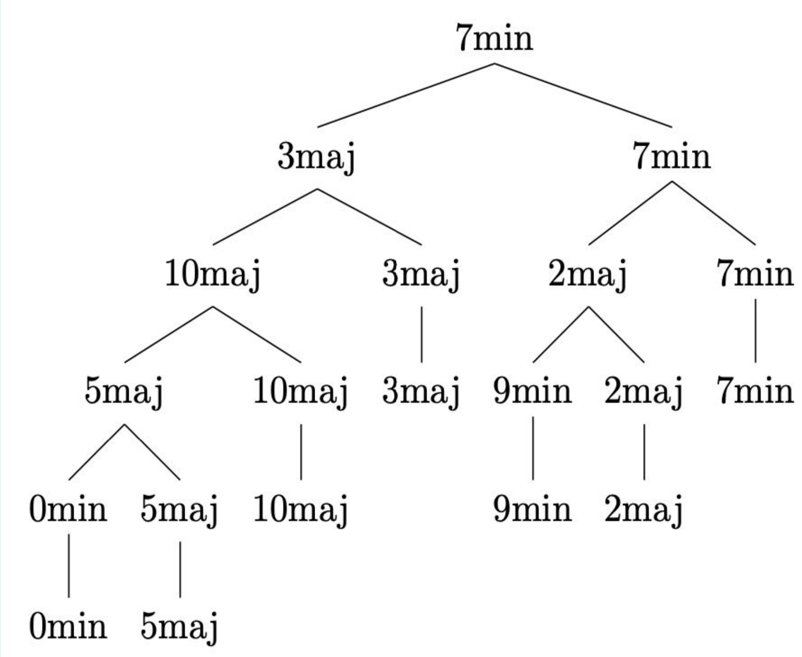


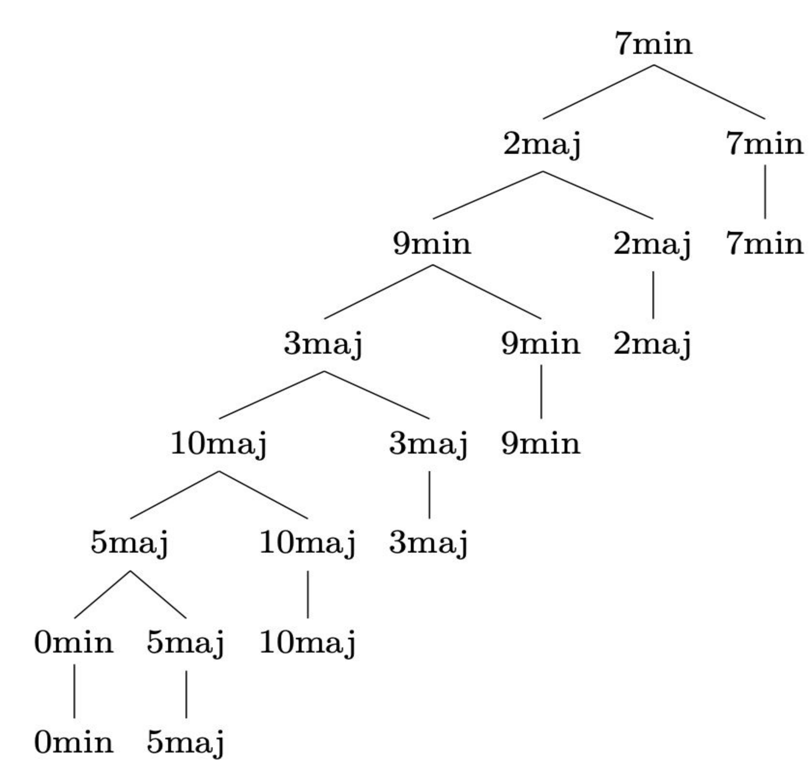


The trees to the left are two alternative parses, suggested by Rohrmeier (2011).

The output of our online parser with dynamic labeling (see below) is reminiscent of analysis 2. The deviation from Rohrmeier's tree might be due to imprecision in the musical description we apply.

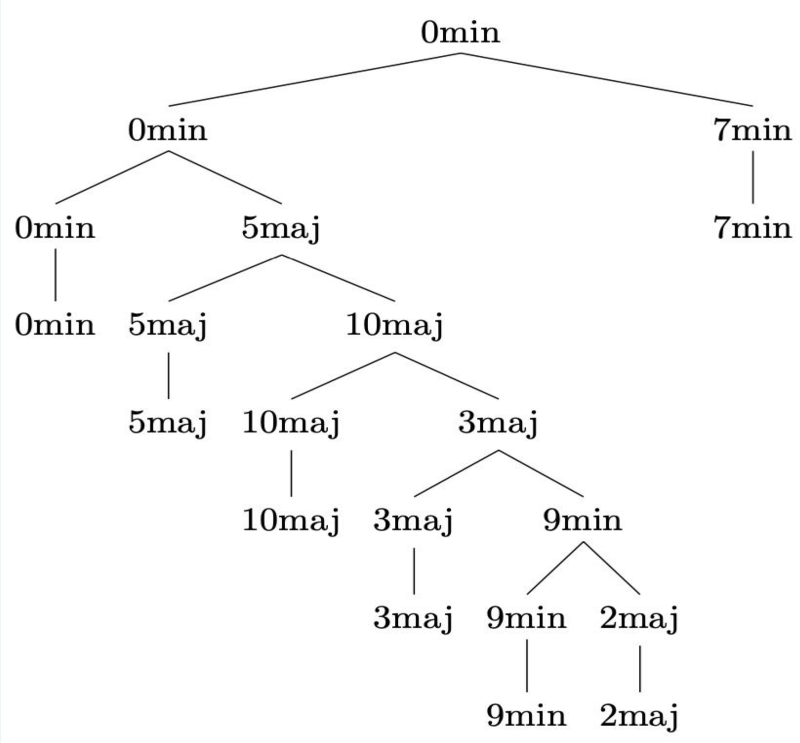
Analysis 1 is derived by the global optimization parser (see below). This tree is the 3rd best parse amongst ~4500 competitors.



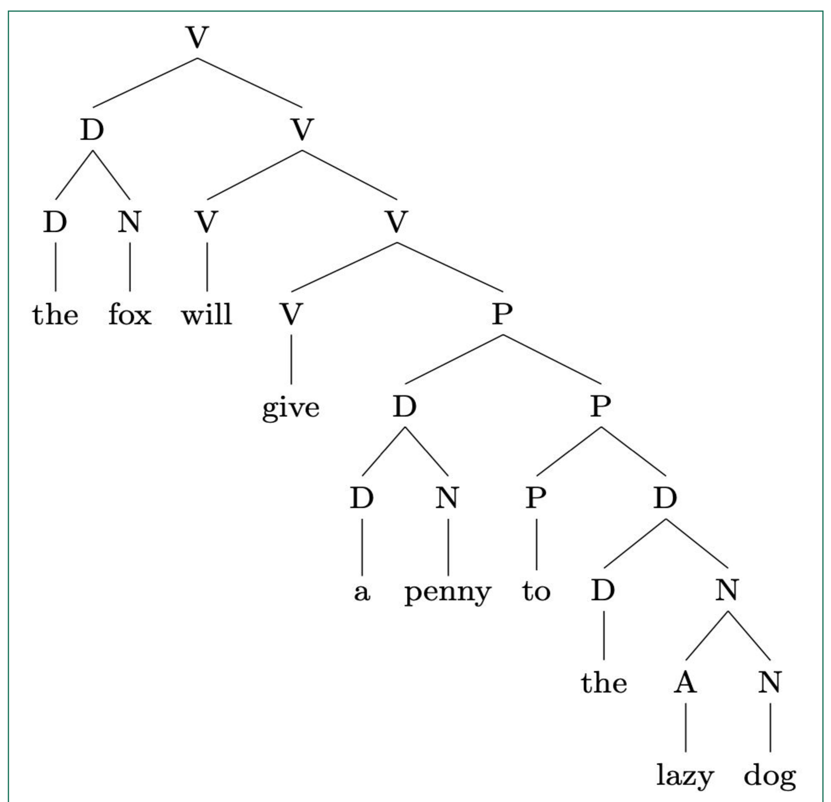


The tree to the right is the first optimal parse produced by the global optimization parser. As can be seen, this is not a very plausible parse.

To derive the results of the global optimization parser, we use two parameters to reward either late merge (l, r > 1), or early merge (l, r < 1), as well as right branching (l > r) as opposed to left branching (r < l). In agreement with musical intuitions, the appropriate parameters for music parsing seem to be those that encourage early merge and right branching (1 > l > r). We don't know yet if the parameters can be derived from the input, and if so, how. This topic needs further research.



By defining a 'linguistic space' on the basis of independently established properties of linguistic categories, the algorithm also displays promise in the language domain (see the parse to the right).



Another version of the algorithm that should be considered is a non-global optimization parser. In this version, we will use dynamic programming to compute the optimal tree for the following *k* events. Thus, in the *i*th iteration, the algorithm computes the optimal tree for event *i* to *i*+*k* at level L_i.

Conclusions

- We present a family of algorithms that use distance functions to create hierarchical trees. Despite using very rough distance functions, all the versions show plausible outcomes.
- Our research is very preliminary in several respects. To arrive at a reasonable approximation for the real cognitive distance between musical events, we would have to consider rhythm, voice-leading specific realization, asymmetrical distances, and the effect of performance.
- Still, we think that the early results of our framework for parsing structured sequences show promise.
- To the extent that our approach is successful, we show that two widespread representations of music behavior are different manifestations of the same cognitive reality.
- After having improved the approximated cognitive distance function, we hope to do broader quantitative research to validate our results.

References:

Grosjean, F., Grosjean, L., & Lane, H. (1979). The patterns of silence: Performance structures in sentence production. *Cognitive psychology*, *11*(1), 58-81.

Lerdahl, F. (2004). *Tonal pitch space*. Oxford University Press.

Rohrmeier, M. (2011). Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, *5*(1), 35-53.