An Algebra for Tree-based Music Generation

Frank Drewes and Johanna Högberg

Umeå University

12 juni 2007

Outline

1. The tree-based approach
   - In general
   - In specific
2. The music algebra
   - A bit of music theory
   - The domain
   - The operations
3. An example: variations on a theme
4. Conclusion

Ranked alphabets, trees, and Σ-algebras

- A ranked alphabet is a finite set of symbols $\Sigma = \cup_{k \in \mathbb{N}} \Sigma_{(k)}$, which is partitioned into pairwise disjoint subsets $\Sigma_{(k)}$, where each symbol in $\Sigma_{(k)}$ is said to have rank k.
- A Σ-algebra $\mathcal{A}$ consists of a non-empty set A (the carrier set of $\mathcal{A}$) and a family of operations $(f_a)_{a \in \Sigma}$. For each $a \in \Sigma_{(k)}$, f_a is a function $f_a \colon A^k \to A$.
- The set T_Σ of trees over Σ is the smallest set of strings such that $a[t_1, \ldots, t_k] \in \mathrm{T}_\Sigma$ if $a \in \Sigma_{(k)}$ and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$.
- A tree language over Σ is a subset of T_Σ.

The tree-based approach

- A set A is finitely generated if there is a finite set of operations such that every element in s can be written as an expression over these operations.
- In this case, there is a ranked alphabet Σ such that $\mathcal{A} = (A, (f_a)_{a \in \Sigma})$ is a Σ-algebra, and each element in A can be written as a formal expression (or term) over $(f_a)_{a \in \Sigma}$.
- There is thus a tree language L over Σ, such that we can identify each tree t in L with a term in $\mathcal{A}$, and evaluate t obtain an element in A.
- The tree-based approach has been applied to many different the domains, including those of pictures, graphs, trees, booleans, probabilities, architecture, and now also music.

Treebag

- When working with tree representations, we would like to generate, rewrite, and evaluate terms, but also to visualise (in a wide sense) the resulting objects.
- In formal language theory there are
  - grammars that generate trees,
  - transducers that rewrite trees, and
  - algebras that evaluate a tree to an object in some domain.
- Different types of grammars, transducers, and algebras have been realised in the software system Treebag.
- Treebag also provides displays to render the generated objects.

Generating a musical piece in Treebag

A piece may be generated using the assembly line principle

- A grammar generates a (tree representation) of a rhythmical skeleton: a branching structure of voices, phrases, measures, and notes.
- This tree is the processed by a sequence of tree transducers, each adding a separate attribute, e.g. tempo, melody, or decoration.
- The process ends with an algebra that interprets the tree as a musical piece, which is then
- visualised by an audio display.

Rather than trying to imitate human composers, our goal is to capture different musical constructions in a simple and natural way.

Notes

- We identify a musical piece with a sequence of notes.
- A note is characterised by its tone, length, accent, and timbre.
  - The tone of a note is the ratio between its frequency and that of a fixed reference note.
  - The length of a note is also measured relative to the length of the reference note: In music, a note whose duration is equal to that of the reference note, is a whole note, a note whose duration is but half of that of the reference note is a half note, and so on.
  - Also accent is a relative property; a note is accented if it is played, for example, louder than any surrounding note.
  - The timbre of a note is the subjective quality which lets us distinguish between instruments.
- For the sake of simplicity, we shall disregard accent and timbre.

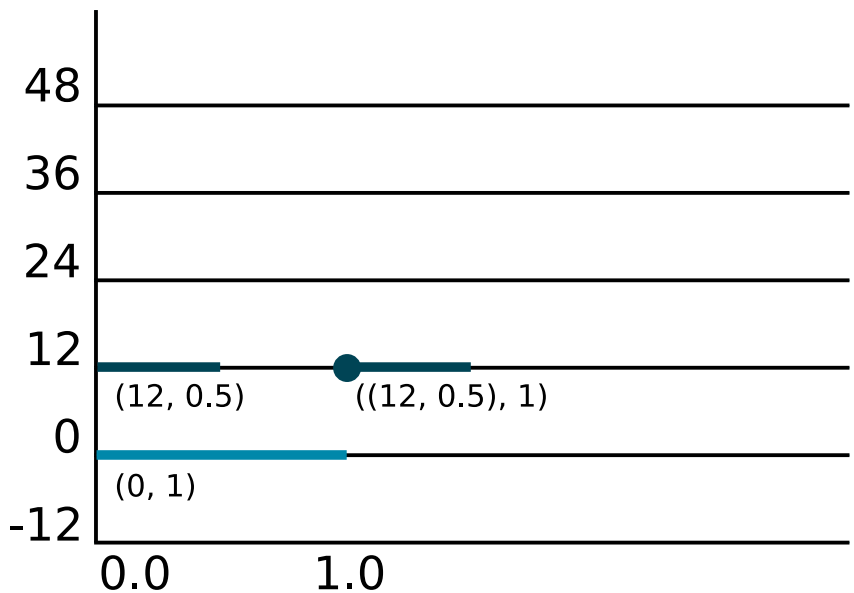
Scales

- A scale is a set of tones.
- The chromatic scale is constructed as follows:
  - First a reference tone is ordained, which would normally be the so-called *a* above middle *c*, an alias for 440 Hz.
  - By doubling this pitch, the tone one octave higher is found, and by dividing it by two, the tone one octave lower.
  - The whole of the audible interval is split into octaves, and every octave is divided into 12 tones, in such a way that the ratio between two consecutive tones amounts to $2^{1/12}$.
  - The tones can thus be referred to by integers: 0 refers to the reference tone, and for every tone t, $t-1$ and $t+1$ refer to the next lower and higher tone, resp., in the chromatic scale.
  - In other words, in our setting, tone t has a frequency of $2^{t/12} \cdot 440$ Hz.

(Played) notes

A note is a pair $n = (t, l) \in \mathbb{Z} \times \mathbb{R}^+$ consisting of the tone t of length l. The set of all notes is denoted by $\mathcal{N}$.

A played note is a tuple (n, s), where n is a note and $s \in \mathbb{R}_0^+$ is the point in time where n is starts to be played.

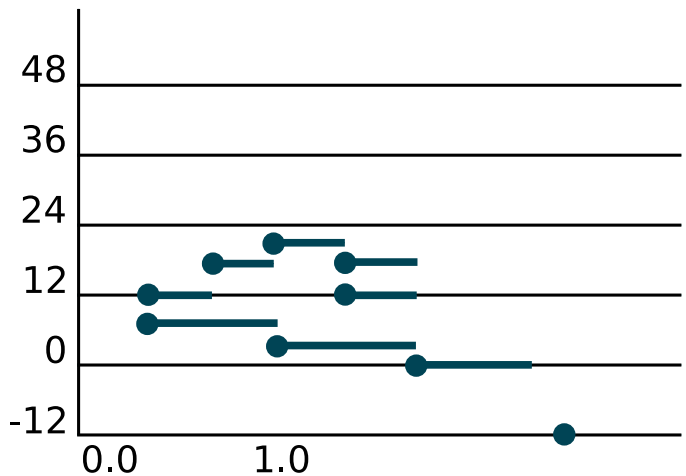


Musical pieces

The set $\mathcal{P}$ of musical pieces is the set of all pairs $P = (N, L)$ s.t.

- $N \subseteq \mathcal{N} \times \mathbb{R}_0^+$ is a finite set of played notes;
- $L \in \mathbb{R}^+$ is the length of the entire piece, with $L \geq s + l$ for all $((t, l), s) \in N$.

The components of a piece P are denoted by N_P and L_P, resp.



The music algebra $\mathbf{M}$

The music algebra, which we denote by $\mathbf{M}$, is a many-sorted algebra whose data domains are the sets $\mathbb{Z}$, $\mathbb{R}^+$, and $\mathcal{P}$.

Before we continue to describe the operations of $\mathbf{M}$, we introduce the following auxiliary notation for convenience:

- Given a set N of played notes and a mapping f on played notes, we let $N[\![(n,s) \mapsto f(n,s)]\!] = \{f(n,s) \mid (n,s) \in N\}$.
- Similarly, if f is a mapping on notes, we let $N[\![n \mapsto f(n)]\!] = \{(f(n),s) \mid (n,s) \in N\}$.

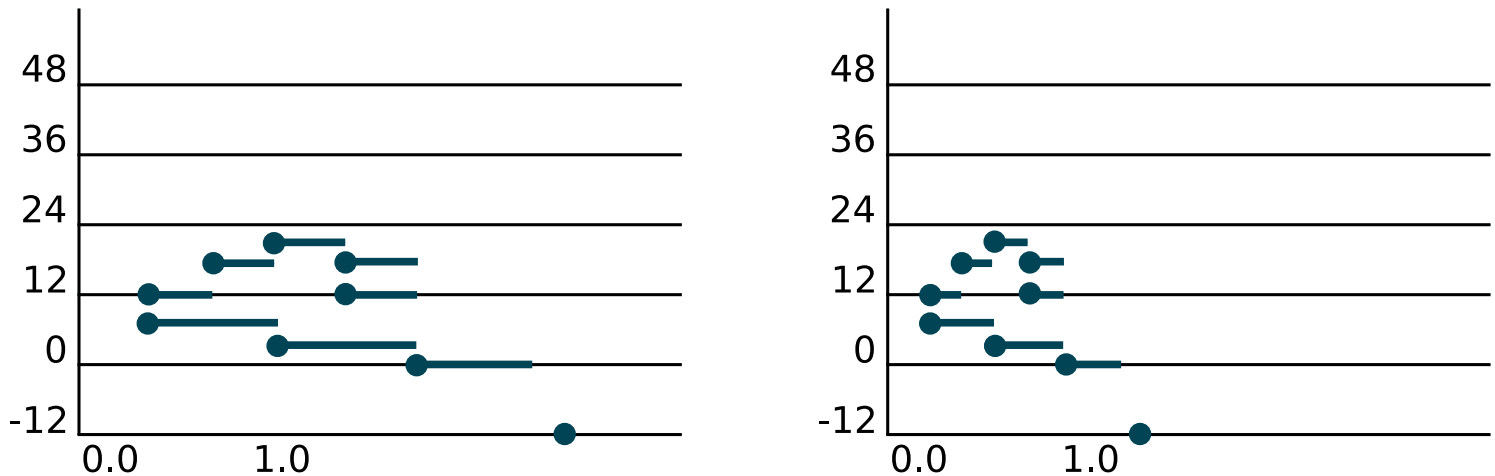
From now on, $P = (N, L)$ and $P' = (N', L')$ are musical pieces.

Scale

For every factor $a \in \mathbb{R}^+$, the operation

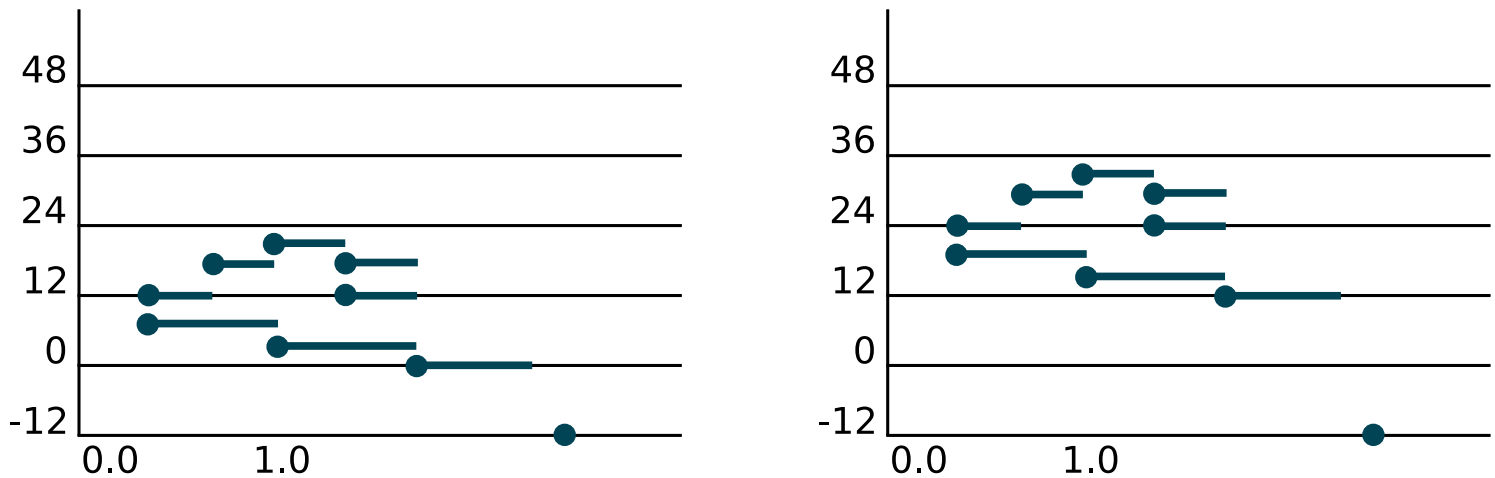
$$\mathit{scale}(P, a) = (N[\![(n, s) \mapsto ((t_n, a \cdot l_n), a \cdot s)]\!], a \cdot L)$$

scales the piece P by the factor a.



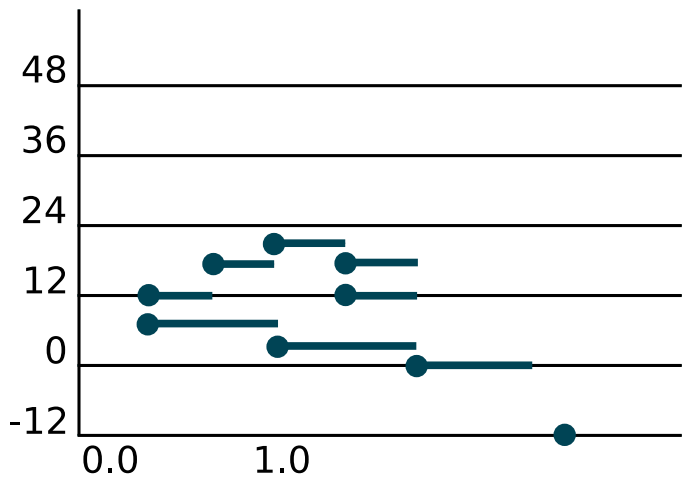
Raise

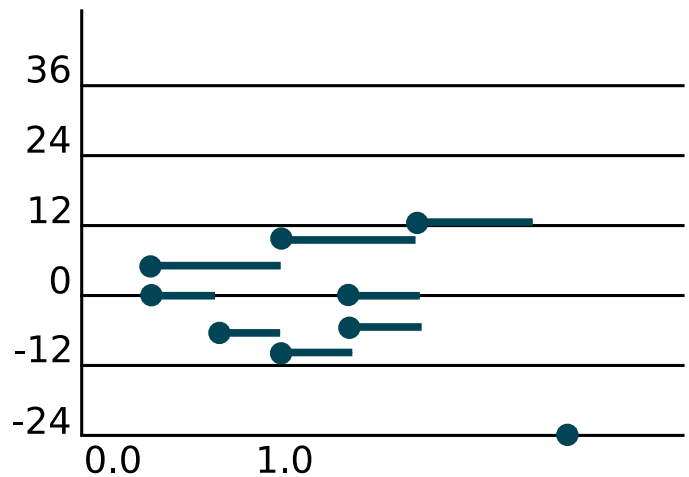
For every tone $t \in \mathbb{Z}$, $\mathit{raise}(P, t) = (N[\![n \mapsto (t_n + t, l_n)]\!], L)$ transposes every tone in the piece P by t.



Invert

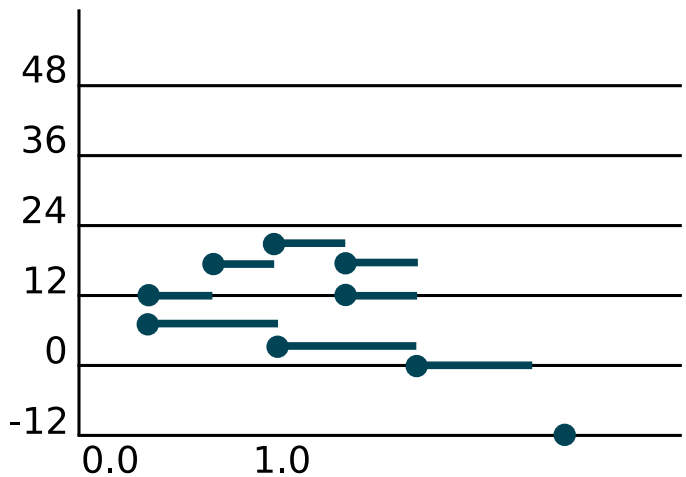
The operation $\mathit{inv}(P) = (N[\![n \mapsto (-t_n, l_n)]\!], L)$ mirrors every note in the piece P around the tone 0.

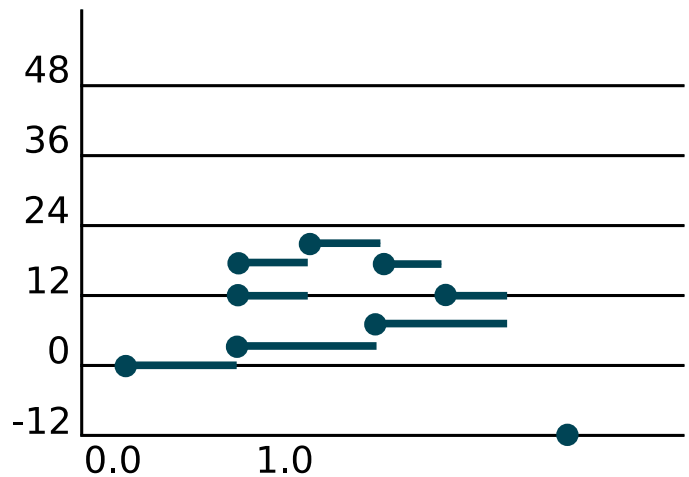




Backwards

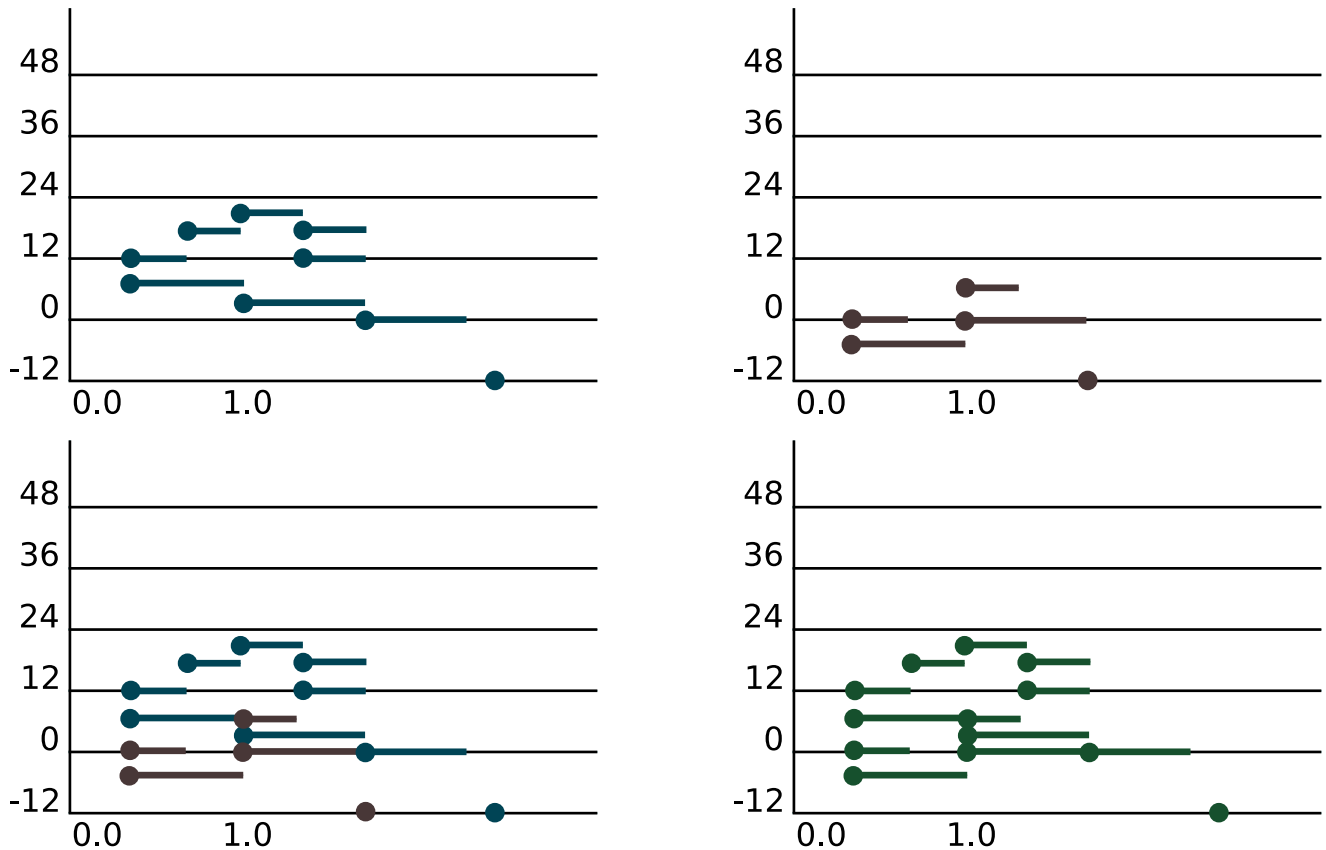
The operation $\mathit{back}(P) = (N[\![(n, s) \mapsto (n, L - s - l_n)]\!], L)$ yields the piece P played in retrograde, i.e. backwards.





Overlay

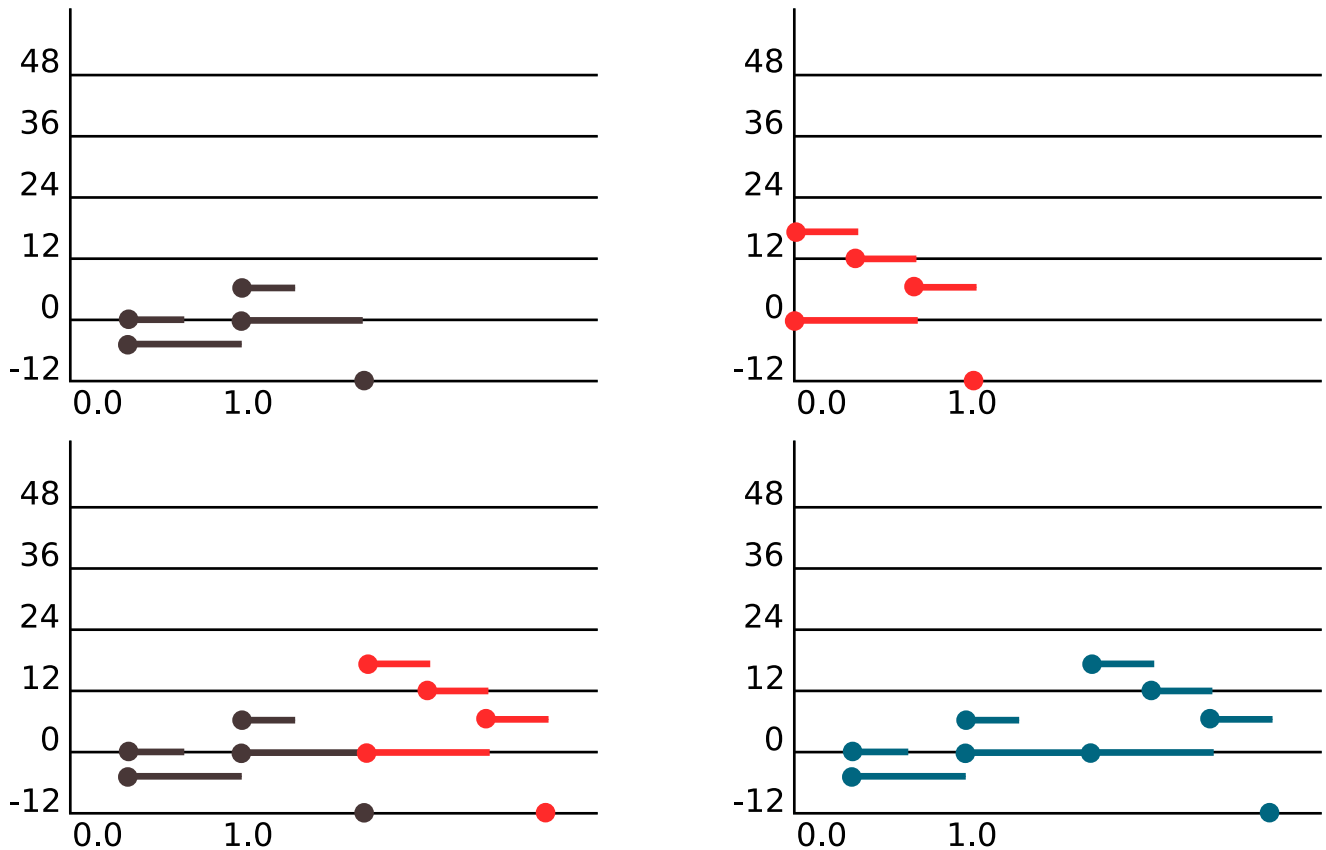
The operation $\mathit{overlay}(P, P')$ yields the overlay of the pieces P and P', given by $\mathit{overlay}(P, P') = (N \cup N', \max(L, L'))$.



Concatenation

The concatenation of pieces P and P' is given by

$$\mathit{concat}(P, P') = (N \cup N'[\![(n, s) \mapsto (n, L + s)]\!], L + L').$$

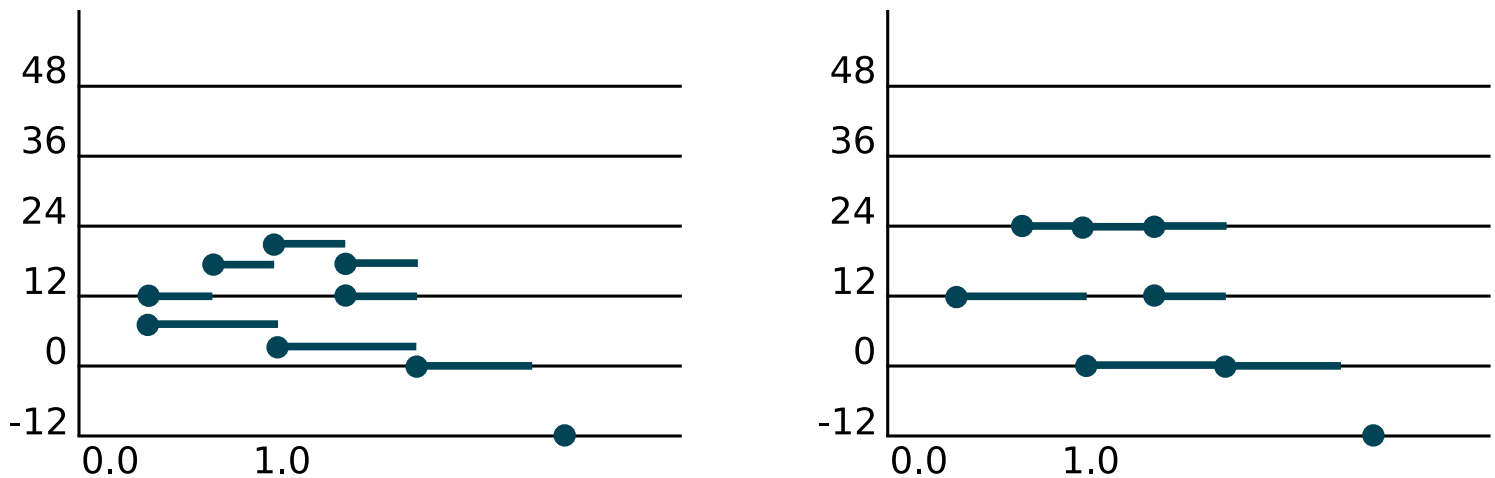


Snap-to-grid

Let $S \subseteq \mathbb{Z}$ be a finite nonempty set of tones. Then $\mathit{snap}_S(P)$ adjusts all tones of notes in P to the nearest tone in S. Formally, for $t \in \mathbb{Z}$, let $\Delta(t) = \min_{s \in S} |t - s|$, and let $\alpha(t) \in S$ be given by

$$\alpha(t) = \begin{cases} t + \Delta(t) & \text{if } t + \Delta(t) \in S \\ t - \Delta(t) & \text{otherwise.} \end{cases}$$

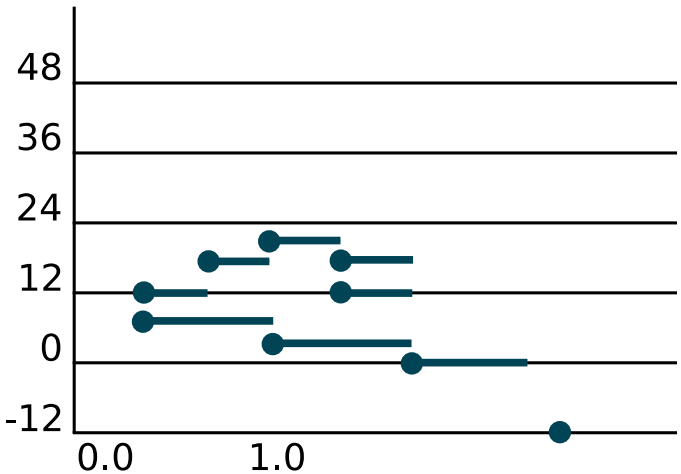
Now, we let $\mathit{snap}_S(P) = (N[\![n \mapsto (\alpha(t_n), l_n)]\!], L)$.



Length, Mute, and Highest

- The operation $\mathit{length}(P) = L$ returns the length of P, and
- $\mathit{mute}(P) = (\emptyset, L)$ yields a silent piece of the same length as P.
- If $N \neq \emptyset$, let $q = \max\{s + l_n \mid (n, s) \in N\}$, i.e., q is the point in time where the last notes in P end. The highest tone at the end of P is returned by

$$\mathit{highest}(P) = \begin{cases} \max\{t_n \mid (n, s) \in N \text{ and } s + l_n = q\} & \text{if } N \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$



Evaluating trees in $\mathbf{M}$

Let $\Sigma_{\mathbf{M}}$ denote the ranked alphabet consisting of

- the operations of $\mathbf{M}$, viewed as symbols of appropriate ranks, and
- all elements of $\mathbb{Z} \cup \mathbb{R}^+ \cup \mathcal{N}$, viewed as constants, i.e. symbols of rank 0. Here, every note $n \in \mathcal{N}$ is identified with the corresponding one-note piece $(\{(n, 0)\}, l_n)$.

For a well-typed tree $t \in \mathrm{T}_\Sigma$ (where well-typedness is defined in the obvious way), we let $\mathrm{val}_{\mathbf{M}}(t)$ denote its value, obtained by recursively evaluating subtrees and applying the operation in the root of the tree to the results.

Generating a variation on a theme

- In a variation, the subject is introduced together with an answer and possibly a countersubject.
- If there are several voices available, then the subject appears at some point in all of them.
- The subject can be explored and developed in various ways, for instance by performing it in inversion, retrograde, diminution, augmentation, as a stretto or as a false entry.

The implementation

- The subject is either generated by the regular tree grammar Subject, or derived from a midi file.
- The subject is given as input to the ttd transducer Themes that derives from it an answer, a countersubject, and an accompaniment.
- The output from Themes is the tree $\mathit{concat}[t_{\mathrm{subj}}, t_{\mathrm{ans}}, t_{\mathrm{csubj}}, t_{\mathrm{acc}}]$.

The declaration of Themes

generators.ttdTransducer("Themes"):
({subject : 1, concat : 2},
{#include(../signature)},
{S, P, A},
{P, A},
{ S[subject[x_1]] → *concat*[subject[P[x_1]],
answer[*raise*[*inv*[P[x_1]], 7]],
csubject[*back*[P[x_1]]],
accompaniment[A[x_1]]],
P[*concat*[x_1, x_2]] → *concat*[P[x_1], P[x_2]],
A[*concat*[x_1, x_2]] → *concat*[A[x_1], A[x_2]] weight 2,
A[*concat*[x_1, x_2]] → *scale*[P[x_1], 1 + *length*[P[x_2]]/*length*[P[x_1]]] },
S).

The exposition

- The themes are then arranged by the ttd transducer Exposition over the five voices.
- The output tree t_{exp} is of the form *concat*$[t_1, \ldots, t_4]$, where $t_i =$ *overlay*$[t_i^{\text{bas}}, t_i^{\text{bar}}, t_i^{\text{ten}}, t_i^{\text{alt}}, t_i^{\text{sop}}]$ for every $i \in [4]$.

Sop.			subj.	countersubj.
Alt.		answer	countersubj.	
Ten.	subject	countersubj.		
Bar.				answer
Bas.		accomp.	accomp.	accomp.

Conclusion

- Treebag is available for download at `http://www.cs.umu.se/~drewes/treebag/`
- The complete system is found at `http://www.cs.umu.se/~johanna/algebra/`
- We thank the Kempe foundation for their support.