

Computing for Musicology (Course code: F104N5)

3. Words which mean music

J.N. Oliveira

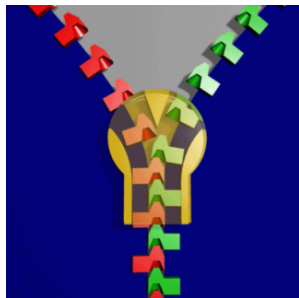
Dept. Informática,
Universidade do Minho
Braga, Portugal

March 2009 (last update: March 2018)

Licenciatura em Música
(<http://www.musica.ilch.uminho.pt/>)
Universidade do Minho
Braga

Zipping

- So far we have seen combinators of words whose outcome is another word.
- Other combinators exist which yield more complex results.
- **Zipping** is among these: it joins two words in the same way a *zipper* joins two edges of fabric (eg. in clothing).



Zipping words

Run

```
zip "Mendelssohn" "Haydn"
```

to obtain

```
[('M', 'H'), ('e', 'a'), ('n', 'y'), ('d', 'd'), ('e', 'n')]
```

that is:

'M'		'H'
'e'		'a'
'n'		'y'
'd'		'd'
'e'		'n'

Notation (a, b) means “the *pair* of a and b ”.

Zipping melodic words with rhythmic words

- Consider word "Abegg". It can be read as the family name of **Pauline von Abegg**, the young friend of Robert Schumann (1810-1856) and Clara Schumann (1819-1896), or as the following note sequence



- Now consider word "11112". This can be read as the word spelling out number 11.112, or the word denoting rhythmic pattern



(four notes of the same duration followed by another twice as long).

Zipping melodic words with rhythmic words

Let us zip both words:

zip "ABegg" "11112" =
 [('A', '1'), ('B', '1'), ('e', '1'), ('g', '1'), ('g', '2')]

The outcome is a sequence of pairs whose left character indicates pitch and whose right one indicates duration (time span). In other words, it is a sequence of **notes**, or music **events**:

A - - b - - e - - g - - g
 1 - - 1 - - 1 - - 1 - - 2

which goes on as follows:

Variations on name Abegg -- Thema (op. 1)

R. Schumann (1810-1856)

Zipping melodic words with rhythmic words

- Of course, it will be much harder to represent the whole of Schumann's score,

Animato. M.M. ♩ = 108.

Thema.



using simple words only. (Look at what's been left out: octaves, bar lines, accidentals, chords, slurs, etc)

- We need more sophistication in our device. In particular, we will need to use **rational numbers** to indicate note durations.

Zipping in general

- Note that in Haskell you can build sequences of any kind, eg. of rational numbers

$$[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}]$$

- So we can zip words with sequences of numbers, as in

$$\text{zip "ABegg"} [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}]$$

where the fractions indicate absolute note lengths ($\frac{1}{4} =$



- Anticipating the need for sharps, flats and the like, we may also use words to denote the notes themselves, for instance

$$\text{zip ["A", "B", "e", "g", "g", "\^G", "A"]} [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}]$$

(bearing ABC notation in mind) instead of letters alone.

Exercises

Exercise 1: Write the following Haskell code into a `.hs` file,

```
import Ratio
-- to understand rational numbers
abeggTheme = zip melody rhythm
where melody = ["A", "B", "e", "g", "g", "^G", "A"]
        rhythm = cycle [1/4, 1/4, 1/4, 1/4, 1/2]
```

and run this. Check that you obtain

```
abeggTheme =
[(("A", 1/4), ("B", 1/4), ("e", 1/4), ("g", 1/4), ("g", 1/2), ("^G", 1/4), ("A", 1/4))]
```



Exercises

Exercise 2: To obtain



from *abeggTheme* further add the following import declarations to your working file,

```
import lpm
import Abc
```

in order to load the *lpm.hs* and *Abc.hs* libraries. (These interface Haskell with the ABC PLUS packages for doing computer music, by Guido Gonzato.) Then add

```
test = abcPlay "F" "3/4" abeggTheme
```

so that you can run command *test* when GHCi interprets your file. The two extra parameters are the signature "F" (F major) and the metre ($\frac{3}{4}$).



Music events and lines

So far,

- A music **event** is regarded as a pair (p, d) where p is a note **pitch** and d is the **duration** (time span) of the event.
- A music **line** (monophonic piece of music) is therefore a *sequence* (list) of such events.
- (Polyphonic) music arises from the **combination** and **transformation** of an arbitrary number of music lines.
- Clearly, such transformations have to do with pitch only, some others with duration only and some others with both.

Transforming music

For example,

- **augmentation** of  leading to



has to do with rhythm only (durations). Similarly for **diminution**:



- **Contrary** motion has to do with pitch only,



be this strict or not.

- **Retrograde** motion has to do with both:



You said “reverse”?

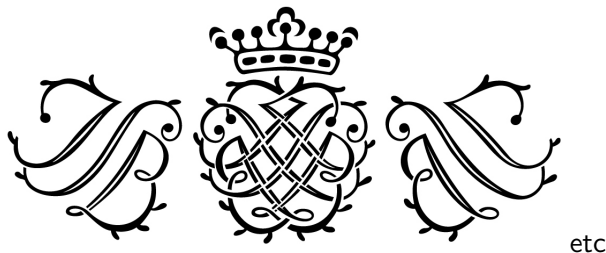
Have a look at the following seal:



What does it mean?

You said “reverse”?

Easy:



— nothing but J.S. Bach (1685-1750)'s seal, where letters 'J', 'S', 'B' overlap each other, mirrored (ie. reversed).

Bach did the same trick in music — a lot. Let us see a well-known example.

Musical Offering BWV 1079

Canon a 2

(nr.1 of Canones diversi super thema regium, BWV 1079)

J.S. Bach (1685-1750)

The image displays the musical score for 'Canon a 2' from 'Canones diversi super thema regium, BWV 1079' by J.S. Bach. The score is written for two voices (treble and bass clef) in common time (C) and B-flat major (two flats). It consists of five systems of two staves each. The first system shows the beginning of the piece, with the second voice entering in the second measure. The subsequent systems continue the canon, showing the intricate interplay between the two parts. The notation includes various rhythmic values and accidentals. A red square with a musical note icon is located at the bottom right of the score.

Musical Offering

Canon structure in terms of units a and b :

a	b
$reverse\ b$	$reverse\ a$

In Haskell:

- upper part: $a \# b$
- lower part: $(reverse\ b) \# (reverse\ a)$ the same as $reverse\ (a \# b)$.

Exercise 3: A *palindrome* is a sequence that reads the same backward as forward, eg. *madam*. Complete

isPalindrome s = ...

so as to check whether a given sequence is a palindrome or not.



Musical Offering

Here is how this canon was written in the original score:

Canones diversi super thema regium.

Canon a 2.

1.

The image shows a musical score for 'Canon a 2' from 'Canones diversi super thema regium'. It consists of three staves of music. The first staff is the main theme, starting with a treble clef, a key signature of two flats (G minor), and a common time signature (C). The second staff is the first canon, starting with a bass clef and a common time signature. The third staff is the second canon, starting with a bass clef and a common time signature. The piece ends with a reversed C-clef on the third staff.

in a single line ($a \mp b$).

The reversed C clef at the end has the obvious meaning...

Putting parts together

- Let us define

$$v1 \# v2 = (v1, v2)$$

in order to express the fact that parts $v1$ and $v2$ (voices) play simultaneously

- Thus we write

$$(a \# b) \# (\text{reverse}(a \# b))$$

to denote the structure of this canon, which clearly spells out the “mathematics” of its construction, based on **retrograde** motion.

- Finally, the *ad infinitum* canon as Bach intended it:

$$\text{cycle}(a \# b) \# \text{cycle}(\text{reverse}(a \# b))$$

Role of pairing

Note the role of **pairing** in music construction:

- music events are pairs (*pitch, duration*)
- two part polyphonic music is a pair of **parts**.

It is therefore convenient to have a special notation for handling pairs: given two operators f and g , notation

$$f \times g$$

will be adopted to denote the operation which applies f and g “in parallel”, to each component of a given input pair:

$$(f \times g)(a, b) = (f(a), g(b))$$

For instance, $(succ \times succ)(\text{'A'}, 2)$ yields $(\text{'B'}, 3)$.

Augmentation/diminution in Haskell

- Consider the first music event ("A", $\frac{1}{4}$) of Schumann's *Abegg* theme.
- To halve its duration we may write ("A", $\frac{1}{8}$) explicitly or, alternatively, evaluate

$$(id \times (/2)) ("A", \frac{1}{4})$$

thereby obtaining the same outcome, where *id* (= *identity*) is the built-in function which does nothing, that is, its output is the same as its input — thus "A" remains unchanged.

- Therefore,

$$map (id \times (/2)) p$$

will have the effect of applying $id \times (/2)$ to all music events in part *p*.

Augmentation/diminution in Haskell

- For instance,

$$\text{map } (id \times (/2)) \text{ abeggTheme}$$

(recall Exercise 1 from slide 8) will yield

$$[("A", \frac{1}{8}), ("B", \frac{1}{8}), ("e", \frac{1}{8}), ("g", \frac{1}{8}), ("g", \frac{1}{4}), ("^F", \frac{1}{8}), ($$

- In summary,

$$\text{map } (id \times (/n)) p$$

will **augment**/**diminish** part p depending on whether n is smaller or larger than 1. Note that the same effect can be obtained by using rational numbers, with great degree of freedom.

Augmentation/diminution in Haskell

For instance,

```
map (id × (( $\frac{3}{2}$ )*)) abeggTheme
```

yields the (somewhat exquisite) augmentation of the Abegg theme which follows,

```
[("A",  $\frac{3}{8}$ ), ("B",  $\frac{3}{8}$ ), ("e",  $\frac{3}{8}$ ), ("g",  $\frac{3}{8}$ ), ("g",  $\frac{3}{4}$ ), ("^G",  $\frac{3}{8}$ ), ("A",  $\frac{3}{8}$ )]
```

that is



Retrograde motion in Haskell

We have already studied and defined the function *reverse* which reverses a list,

$$\begin{aligned} \text{reverse } [] &= [] \\ \text{reverse } (a : l) &= (\text{reverse } l) ++ [a] \end{aligned}$$

Applied to a melodic line *p*, *reverse* will yield *p* in retrograde motion.

Exercise 4: Evaluate and play the *royal theme*

$$\begin{aligned} \text{royalTheme} = & \\ & [(\text{"C"}, \frac{1}{2}), (\text{"E"}, \frac{1}{2}), (\text{"G"}, \frac{1}{2}), (\text{"A"}, \frac{1}{2}), (\text{"=B"}, \frac{1}{2}), (\text{"z"}, \frac{1}{4}), (\text{"G"}, \frac{1}{2}), (\text{"^F"}, \end{aligned}$$

of Bach's BWV 1079 in retrograde motion.



Exercises

Exercise 5: Use *zip* to compute the length of the shortest of two given words.



Exercise 6: Given a pair (a, b) , *fst* selects the first element of (a, b) and *snd* the second, that is,

$$\text{fst } (a, b) = a, \text{ snd } (a, b) = b$$

Run the following expressions:

```
map fst [(1, 2), (3, 4)]
```

```
map snd (zip "Mendelssohn" (cycle [0, 1]))
```



Exercises

Exercise 7: Suppose you have a list of numbers a_1, a_2, \dots, a_n and you want to add them up to obtain number $a_1 + a_2 + \dots + a_n$. We may consider defining a new operator for this, say *addAll*, such that

$$\text{addAll } [1, 3, 5] = 9$$

$$\text{addAll } [0, 0, 0] = 0$$

etc.

1. Build *addAll* after inspecting and completing the following properties:

$$\text{addAll } [] = 0$$

$$\text{addAll } [a] = \dots$$

$$\text{addAll } (l ++ r) = \dots$$

2. Repeat the same exercise for the operation *count* which counts how many elements can be found in a given sequence.



Further readings

- Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, January 1999.
- P. Hudak: *The Haskell School of Expression - Learning Functional Programming Through Multimedia*. Cambridge University Press, 2000. ISBN 0-521-64408-9.