Computing for Musicology (0809.F104N5) 3. Words which mean music

J.N. Oliveira

Dept. Informática,
Universidade do Minho
Braga, Portugal

March 2009
Licenciatura em Música
(http://www.musica.reitoria.uminho.pt/licenciatura.html)
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

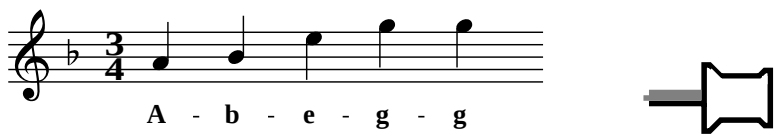
$$[('M','H'),('e','a'),('n','y'),('d','d'),('e','n')]$$

that is:

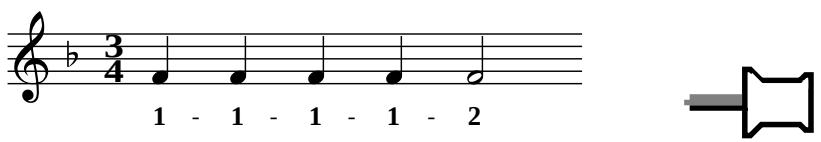
'M'	'H'
'e'	'a'
'n'	'y'
'd'	'd'
'e'	'n'

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 (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



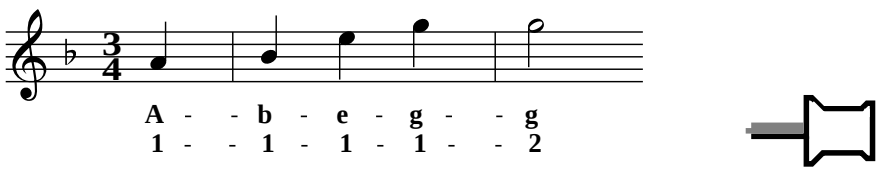
(four notes of the same duration followed by another twice as long).

Zipping melodic words with rhythmic words

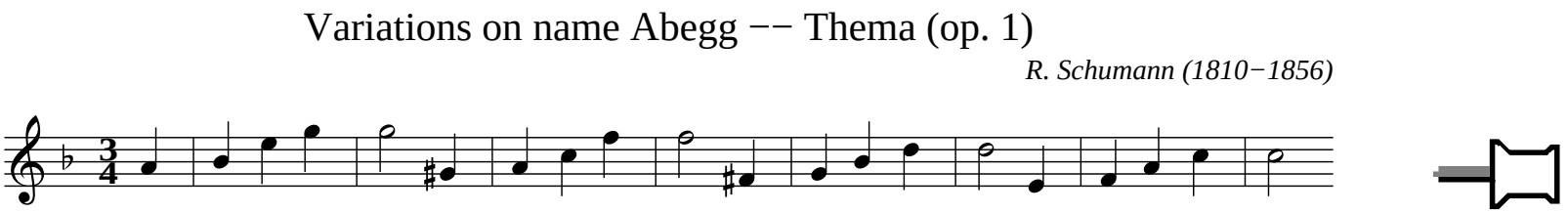
Let us zip both words:

$$zip\ \texttt{"ABegg"}\ \texttt{"11112"} =$$
$$[(\texttt{'A'},\texttt{'1'}),(\texttt{'B'},\texttt{'1'}),(\texttt{'e'},\texttt{'1'}),(\texttt{'g'},\texttt{'1'}),(\texttt{'g'},\texttt{'2'})]$$

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**:

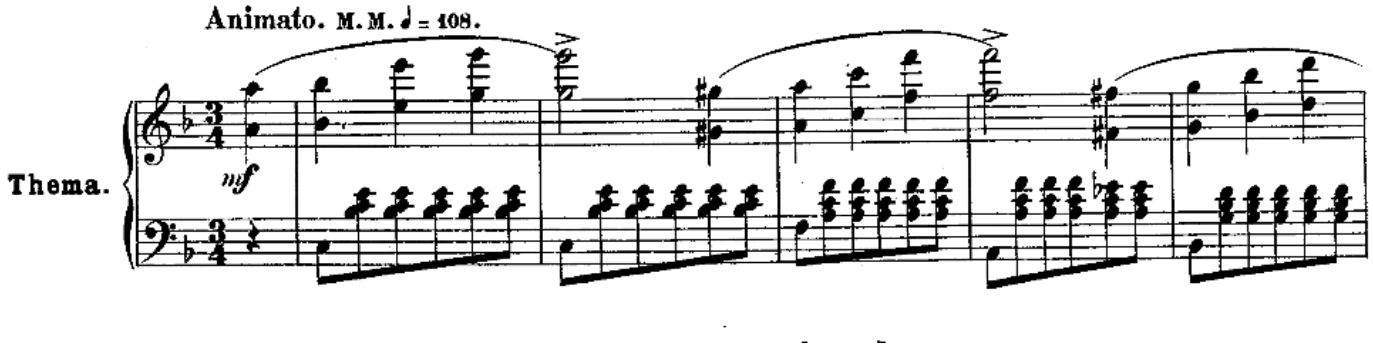


which goes on as follows:



Zipping melodic words with rhythmic words

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



etc

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.

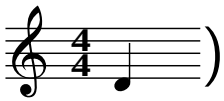
Zipping in general

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

$$[1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 2]$$

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

$$\mathit{zip}\ \texttt{"ABegg"}\ [1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 2]$$

  where the fractions indicate absolute note lengths ($1 \,\%\, 4 =$ )

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

$$\mathit{zip}\ [\texttt{"A"}, \texttt{"B"}, \texttt{"e"}, \texttt{"g"}, \texttt{"g"}, \texttt{"\^{}G"}, \texttt{"A"}]$$
$$[1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 4, 1 \,\%\, 2, 1 \,\%\, 4, 1 \,\%\, 4]$$

  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]
```

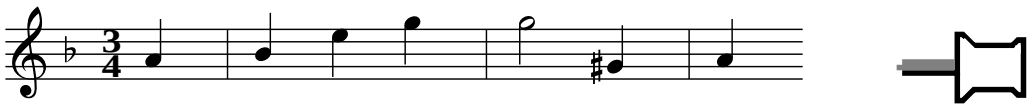
and run this. Check that you obtain

```
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 $(3 \,\%\, 4)$.

□

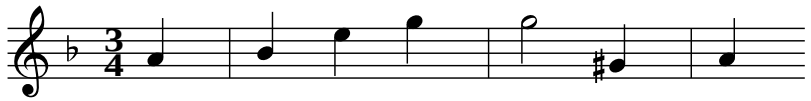
Music events and lines

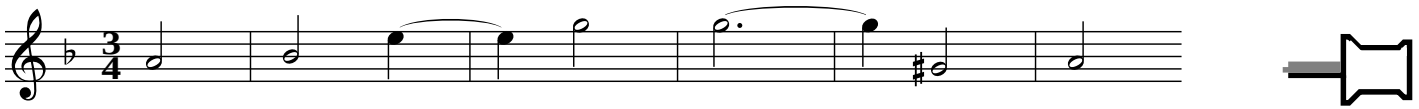
So far,

- 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.

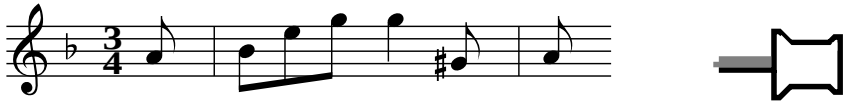
Transforming music

For example,

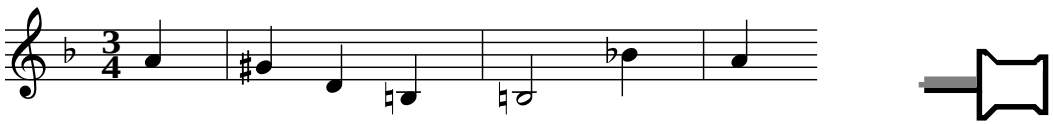
- **augmentation** of  leading to



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

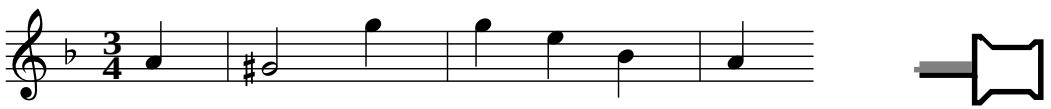


- **Contrary** motion has to do with pitch only,



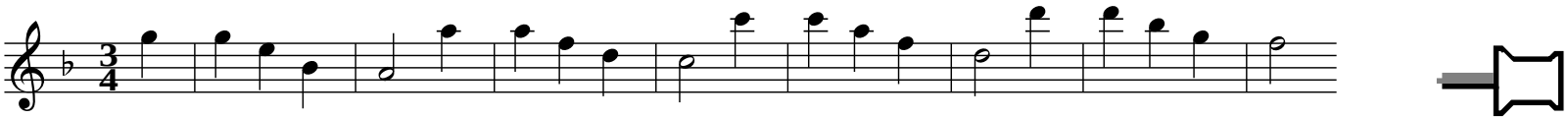
be this strict or not.

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



Retrograde motion

Schumann himself bases the second part of the *Abegg* theme on (part of) this retrograde cell:



How do we describe these musical **transforms** in the Haskell language?

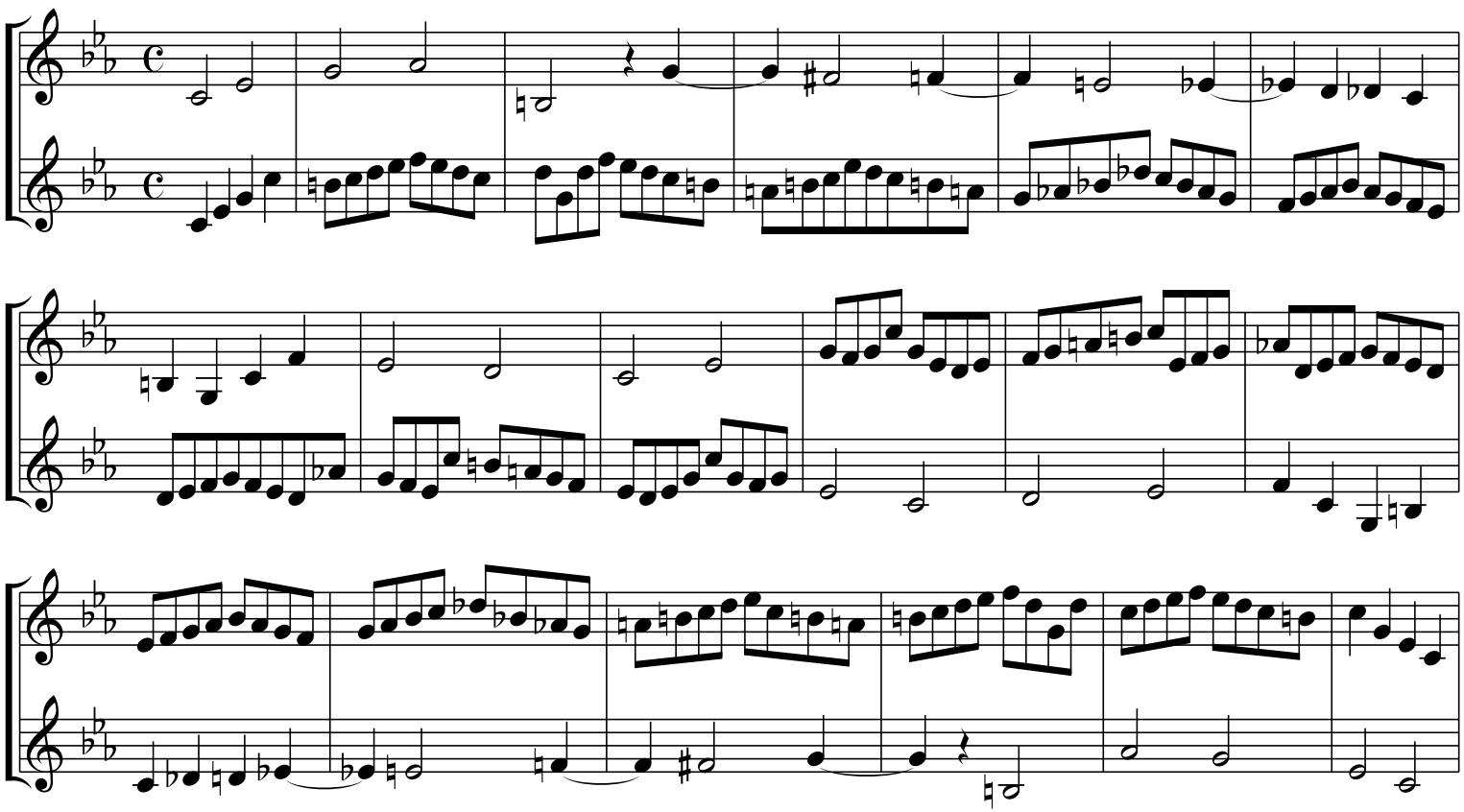
> **Retrograde** motion is the easiest to capture, as the *reverse* operation does this for free:
>
> *reverse* `"ABegg"` = `"ggeBA"`

A more thorough illustration of retrograde motion is given in the slide which follows.

Musical Offering

Canon a 2
(nr.1 of Canones diversi super thema regium)
Musical Offering (BWV 1079)

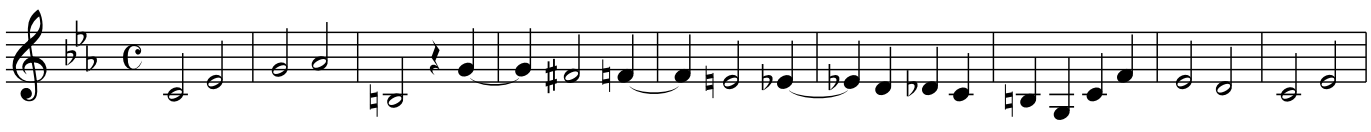
J.S. Bach (1685−1750)

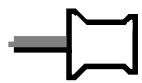


Musical Offering

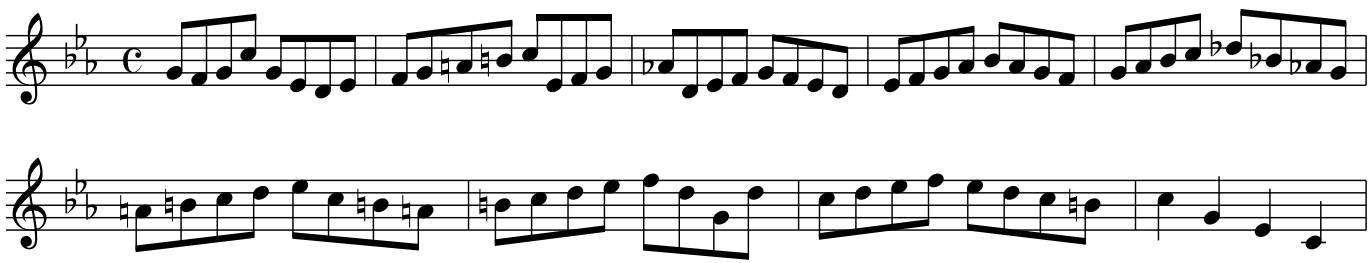
Clearly, two units can be found in this cannon — the King's theme (a) and Bach's own answer (b):

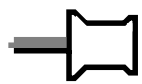
a =





b =





Musical Offering

Structure of canon in terms of units a and b:

a	b
reverse b	*reverse* a

In Haskell:

- upper part: $a +\!\!+ b$
- lower part: $(\mathit{reverse}\ b) +\!\!+ (\mathit{reverse}\ a)$ the same as $\mathit{reverse}\ (a +\!\!+ b)$.

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

$$\mathit{isPalindrome}\ s = \ldots$$

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

□

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

$$(a +\!\!+ b) \# (reverse\ (a +\!\!+ b))$$

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

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

$$cycle\ (a +\!\!+ b) \# cycle\ (reverse\ (a +\!\!+ b))$$

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

$$f \asymp g$$

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

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

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

Augmentation/diminution in Haskell

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

$$(id \mathbin{>\!\!<} (/2))\ (\texttt{"A"}, 1 \,\%\, 4)$$

  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* $(id \mathbin{>\!\!<} (/2))$ *p* will have the effect of applying $id \mathbin{>\!\!<} (/2)$ to all music events in part *p*.

Augmentation/diminution in Haskell

- For instance,

  $$map\ (id \gg\!\prec (/2))\ abeggTheme$$

  (recall Exercise 1 from slide 8) will yield

  $$[(\texttt{"A"}, 1 \,\%\, 8), (\texttt{"B"}, 1 \,\%\, 8), (\texttt{"e"}, 1 \,\%\, 8), (\texttt{"g"}, 1 \,\%\, 8), (\texttt{"g"}, 1 \,\%\, 4), (\texttt{"\^{}F"}, 1 \,\%\, 8), (\texttt{"G"}, 1 \,\%\, 8)]$$

- In summary,

  $$map\ (id \gg\!\prec (/n))\ p$$

  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

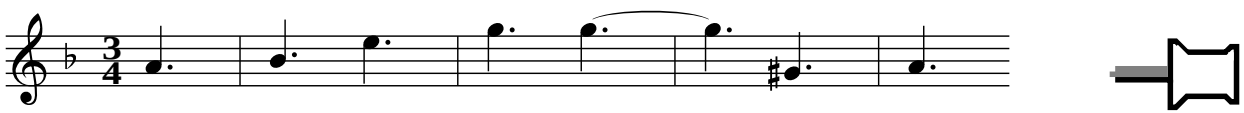
For instance,

$$map\ (id \gg\!< ((3 \,\%\, 2)*))\ abeggTheme$$

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

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

that is



Retrograde motion in Haskell

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

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

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

Exercise 4: Evaulate and play the *royal theme*

$$\begin{aligned} \mathit{royalTheme} = [&(\texttt{"C"}, 1 \,\%\, 2), (\texttt{"E"}, 1 \,\%\, 2), (\texttt{"G"}, 1 \,\%\, 2), (\texttt{"A"}, 1 \,\%\, 2), (\texttt{"=B,"}, 1 \,\%\, 2), (\texttt{"z"}, 1 \,\%\, 4), (\texttt{"G"}, 1 \,\%\, 2), (\texttt{"\^{}F"}, 1 \,\%\, 2), (\texttt{"=F"}, 1 \,\%\, 2), (\texttt{"=E"}, 1 \,\%\, 2), (\texttt{"\_E"}, 1 \,\%\, 2), (\texttt{"=D"}, 1 \,\%\, 4), (\texttt{"\_D"}, 1 \,\%\, 4), (\texttt{"C"}, 1 \,\%\, 4), (\texttt{"=B,"}, 1 \,\%\, 4), (\texttt{"G,"}, 1 \,\%\, 4), (\texttt{"C"}, 1 \,\%\, 4), (\texttt{"F"}, 1 \,\%\, 4), (\texttt{"E"}, 1 \,\%\, 2), (\texttt{"D"}, 1 \,\%\, 2), (\texttt{"C"}, 1 \,\%\, 2), (\texttt{"E"}, 1 \,\%\, 2)] \end{aligned}$$

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,

$$fst\ (a, b) = a\ ,\ snd\ (a, b) = b$$

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 $a1, a2, ..., an$ and you want to add them up to obtain number $a1 + a2 + ... + an$. We may consider defining a new operator for this, say *addAll*, such that

$$\textit{addAll}\ [1, 3, 5] = 9$$
$$\textit{addAll}\ [0, 0, 0] = 0$$

etc.

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

$$\textit{addAll}\ [\,] = 0$$
$$\textit{addAll}\ [a] = ...$$
$$\textit{addAll}\ (l +\!\!+ r) = ...$$

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.