computers and music (2)

music programming and beyond

learning goals:

- familiarity with some simple musical operations
- knowledge of the goals and ideas of music processing
- basic familiarity with the main concepts behind the SALIERI system
- overview knowledge of some challenging musical problems
- basic knowledge of an example of an “intelligent” composition system (Emmy)

Fundamental musical operations:

- Concatenation: append one sequence to another
  (analogous to string concatenation: $abc + xyz = abcxyz$)

  `[`$n_1\ n_2\ \ldots\ n_k$`] + [`$m_1\ m_2\ \ldots\ m_l$`]`
  `=` $[n_1\ n_2\ \ldots\ n_k\ m_1\ m_2\ \ldots\ m_l]$

  For example:

```
[ c2/4 e2/4 ] + [ g2/8 f2/8 e2/2 ]
= [ c2/4 e2/4 g2/8 f2/8 e2/2 ]
```

Fundamental musical operations (continued):

- Retrograde: reverse musical sequence in time
  (analogous to string reversal: $wxyz \rightarrow zyxw$)

  `retro([`$n_1\ n_2\ \ldots\ n_k$`]) = [`$n_k\ \ldots\ n_2\ n_1$`]`

  For example:

```
retro([ c2/4 e2/4 g2/8 f2/8 e2/2 ])
    = [ e2/2 f2/8 g2/8 e2/4 c2/4 ]
```

Fundamental musical operations (continued):

- Transposition: shift up or down in pitch
  (analogous to constant increment: $1234 \rightarrow 3456$)

```
c   c#/d&   d   d#/e&   e   f   f#/g&   g   g#/a&   a   a#/b&   b

0     1     2     3     4   5     6     7     8     9     10    11
```

  For example:

```
transp([ c2/4 e2/4 g2/8 f2/8 e2/2 ], 2)
           0     4     7     5     4   +2
           2     6     9     7     6
     = [ d2/4 f#2/4 a2/8 g2/8 f#2/2 ]
```

Fundamental musical operations (continued):

- Inversion: flip direction of musical intervals
  (analogous to forming ‘vertical mirror image’: $4567 \rightarrow 4321$)

  For example:

```
invert([ c2/4 e2/4 g2/8 f2/8 e2/2 ])
           0     4     7     5     4
             +4    +3    -2    -1
             -4    -3    +2    +1
           0    -4    -7    -5    -4
           0     8     5     7     8
     = [ c2/4 g#1/4 f1/8 g1/8 g#1/2 ]
```

Fundamental musical operations (continued):

- Combination: combine melody from one sequence with rhythm from another (shorter sequence is repeated)

  For example:

```
comb([ c2/4 e2/4 g2/8 f2/8 e2/2 ],
     [_/8 _/8 _/4])
   = [ c2/8 e2/8 g2/4 f2/8 e2/8 ]
```

Music Programming

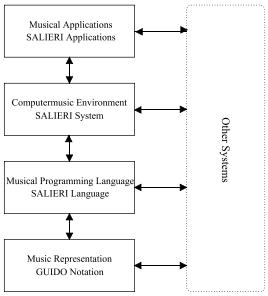
Goals & Ideas:

- build / specify music in the same way as algorithms / software
- process music algorithmically (like, *e.g.*, numerical or image data)
- build / specify interactive music systems (instruments, installations, . . .)

Various Realisations:

- *Music (or sound) programming languages (+ compilers / interpreters):*
  - extensions of existing programming languages (C, LISP, . . .)
  - new, dedicated programming languages (CSound, SALIERI, Elody, . . .)
- *Interactive music systems:*
  - reactive, graphical environments (MAX, Pure Data (PD), Open Music, . . .)
  - event-based programming environments (Realtime SALIERI, Realtime CSound, . . .)

The SALIERI Approach



The SALIERI System

- interactive software environment for creating, manipulating, and analysing musical material
- based on a music programming language with native music data types based on GUIDO Music Notation
- facilitates exploring algorithmic and mathematical concepts in music

[Holger Hoos, Kai Renz, Jürgen Kilian, Thomas Helbich; 1993–2000]

The SALIERI Music Programming Language

- Universal programming language with special support for music processing
- Interpreted language (internally compiled for increased efficiency)
- Uses dynamic typing (types of variables are not declared, but determined at run-time)
- At run-time, all objects reside in a workspace, where they can be inspected (viewed/played), modified, and deleted.

The SALIERI Music Programming Language (continued)

- *Built-in functions*
  - Musical generators, variators, and descriptors
  - Uniform functions for serial datatypes (sequences, strings, lists)
  - Standard control structures (while, if, ...)
  - Standard functions for basic datatypes (arithmetic, logic, etc.)
- *User-defined functions*
  - Functional objects, can be created, copied, deleted
  - Arbitrary number and types of parameters
  - Arbitrary result types
  - Recursive function definitions and calls supported

Example 1: Scales

```
% create and play a scale (c-major):
cm := [c1/4 d1/4 e1/4 f1/4 g1/4 a1/4 b1/4 c2/4];
play(cm);

% set rhythm to all 16th notes and play:
cmf := comb(cm,[_/16]);
end := [c1/4];
play(cmf + end);

% play twice (plus end note):
play(cmf + cmf + end);

% play scale over two octaves
% (= scale + version transposed by 12 semitones = 1 octave):
play(cmf + transp(cmf,+12) + end);
```

Example 2: More scales

```
% create a chromatic scale and play it:
chr :=  [ c1/32 c# d d# e f f# g g# a a# b ];
play(chr);

% generate a chromatic scale over 4 octaves
% by repeated transposition and concatenation:
s := [];
c := chr;
loopn(4,
  's := s + c;
  c := transp(c,+12);
  ');

% play this upwards and downwards (plus end note):
play(s + retro(s) + [c1/4]);
```

Example 3: Simple composition

```
seq := [ a1/4 c b c# g# f# e& g d b& e f ];
seq2 := comb(seq, [_/2 _/4] + [_/4 _/2]
                + [_/8 _/8 _/4 _/4] + [_/4. _/8 _/8 _/8]);

v1 := transp(seq2,-24);
v2 := [_/2]+transp(invert(seq2),-12);
A := {v1,v2};
```

Example 3: Simple composition (continued)

```
seq3 := comb(retro(seq), [_/8 _/4 _/8 _/4] + [_/8 _/8 _/4. _/8]
                       + [_/4 _/8 _/8 _/2]);

v3 := transp(seq3,-24);
v4 := [_/4]+transp(retro(seq3),-7);
B := {v3,v4};

v5 := invert(seq2);
v6 := [_/4]+transp(invert(seq3),-12);
B2 := {v3,v5,v6};

piece := A+B+B2;
play(piece);
```

Music, Creativity, and Artificial Intelligence

Music is an Artform!

Most musical processes and activities inherently rely on human intelligence and creativity

$\rightsquigarrow$ Can these (ever) be fully automated?

Some challenging musical problems:

- *composition:*
  compose an interesting piece of music
  (variations: in the style of ..., similar to ...)
- *performance:*
  deliver a compelling performance of a musical piece
  (similar variations as for composition)
- *instrument design:*
  convincingly synthesise an acoustic instrument
  (*e.g.*, piano, oboe, violin, singing voice)

More musical challenges:

- *similarity:* decide whether two pieces or fragments of music are similar to each other
  (variations: given a piece, find all similar pieces from a given collection)
- *variation:* given two pieces of music, decide whether one is a variation of the other
  (related: given a piece of music, create an interesting variation of it)
- *perception:* given a piece or fragment of music, decide whether it is perceived as interesting, beautiful, merry, sad, ...

...

All these problems are essentially unsolved.

But:

- they are studied extensively, *e.g.*, in Artificial Intelligence, and some progress has been made over the past decades;
- they are similar in difficulty and significance to
  - automated theorem proving and discovery,
  - automated writing of prose or poetry,
  - automated translation of prose or poetry,
  - automated painting or sculpting, *and*
  - automated conversation (chat-bots, Turing test).

Example (the most impressive I'm currently aware of):

David Cope's "Experiments in Musical Intelligence" (Emmy)

$\rightsquigarrow$ automated system that composes in the style of certain composers based on database of pieces using randomised recombination

Emmy + limited amount of human tuning and selection passes "Musical Turing Test", *i.e.*, human experts (composers) cannot reliably distinguish its compositions from pieces by human masters, *e.g.*, J.S. Bach.

Food for Thought:

- If you had invented an algorithm that could compose convincing pieces in the style of the Beatles (Spice Girls, ABBA, Ravel, Mozart), would you tell anyone? What would you use it for? Why would you have spent the time inventing it? What would the fact that you could create it say about the work of the Beatles (Spice Girls, ...)?
- If you had an algorithm that could play the piano as well as the best human players, would you rather listen to it or to a friend who doesn't play quite as well? Why?
- How would you feel about a machine that can listen to music and reliably predict whether you like it? Would this be useful? Interesting? Disturbing?

More Food for Thought:

- Is there a clear boundary between computer-assisted composition and automated composition?
- What kind of algorithmic / software tools would you enjoy using for creatively working with music? Are these likely to be available? Could you make them yourself?
- What is a musical work? The score? The autograph (i.e., the composer's original manuscript)? A particular performance? The set of all performances? The composer's vision? A critic's interpretation?

Some references and pointers:

- SALIERI: http://www.salieri.org/
- GUIDO: http://www.salieri.org/GUIDO
- GUIDO NoteServer: http://www.noteserver.org/
- Holger Hoos' homepage (papers and compositions): http://www.cs.ubc.ca/~hoos
- Pure Data (PD): http://www.pure-data.org/about/
- Computer Music Journal (http://mitpress2.mit.edu/e-journals/Computer-Music-Journal; available at the UBC Music Library)
- Tom Johnson: Self-Similar Melodies. Editions 75, 1996
- David Cope's "Experiments in Musical Intelligence" (http://arts.ucsc.edu/faculty/cope/Emmy.html)

More references and pointers:

- Douglas Hofstaedter's work (http://www.psych.indiana.edu/people/homepages/hofstadter.html), interesting thought on music can be found in his books "Metamagical Themas" and "Goedel, Escher, Bach"
- Leonhard Bernstein: The Unanswered Question. Harvard University Press, 1981.
- Various papers by Roger Dannenberg (http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/rbd/www/bib.html) and Belinda Thom (http://www-2.cs.cmu.edu/~bthom/pubs.html)
- Musical Dice Game (http://sunsite.univie.ac.at/Mozart/dice/)