Development of a Composer's Sketchbook

Georg BOENN
School of Electronics
University of Glamorgan
Pontypridd CF37 1DL
Wales, UK
gboenn@glam.ac.uk

Development of a Composer’s Sketchbook

1. Computer Assisted Composition
2. Algorithms
3. Implementation
4. Future?

Development of a Composer’s Sketchbook

1. Computer Assisted Composition

Development of a Composer’s Sketchbook

- open source
- GNU General Public License v2
- C++, OOP
- cross-platform: Linux, Win, Mac
- wxWidgets framework

Computer Assisted Composition

- Applications assist the composer to manage the manifold of:
  - musical ideas
  - symbolic representations
  - musical structures
  - sounds
  - performances

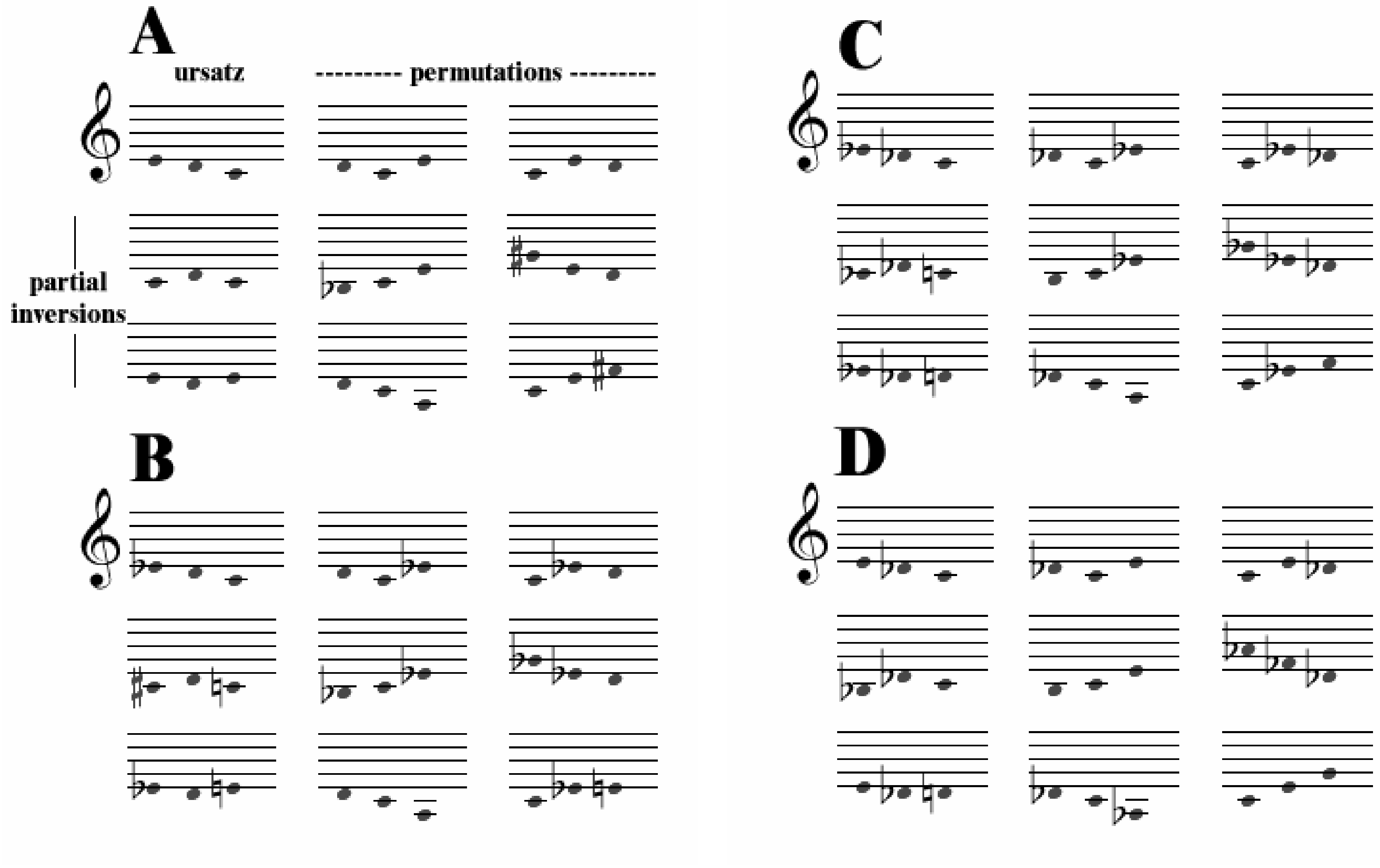
intelligent assistant

- sketchbook paradigm
- freedom of choice
- changes of initial parameters can trigger surprising twists in the work
- direct and immediate comparisons

invention and modelling of melodic structures

- user-defined database of musical cells within the program
- use of three-note cells which came out from the investigation of the major and minor third
- Analysis of pieces by Arnold Schoenberg and Charles Ives led to the following matrix:

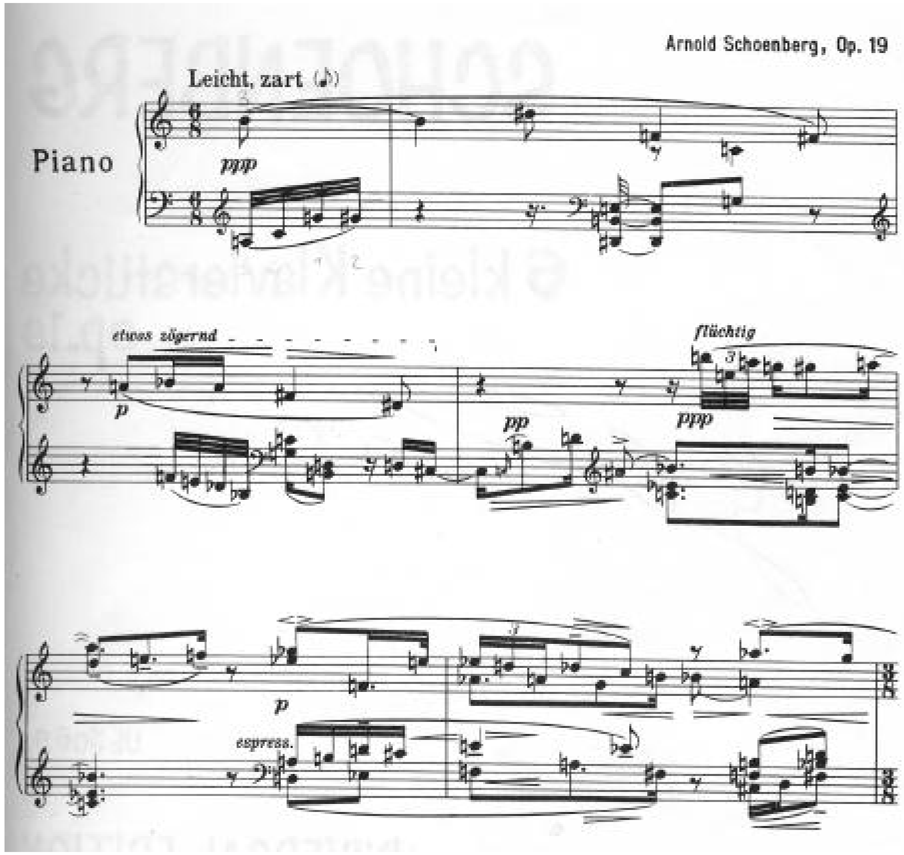
Do you know what the matrix is?

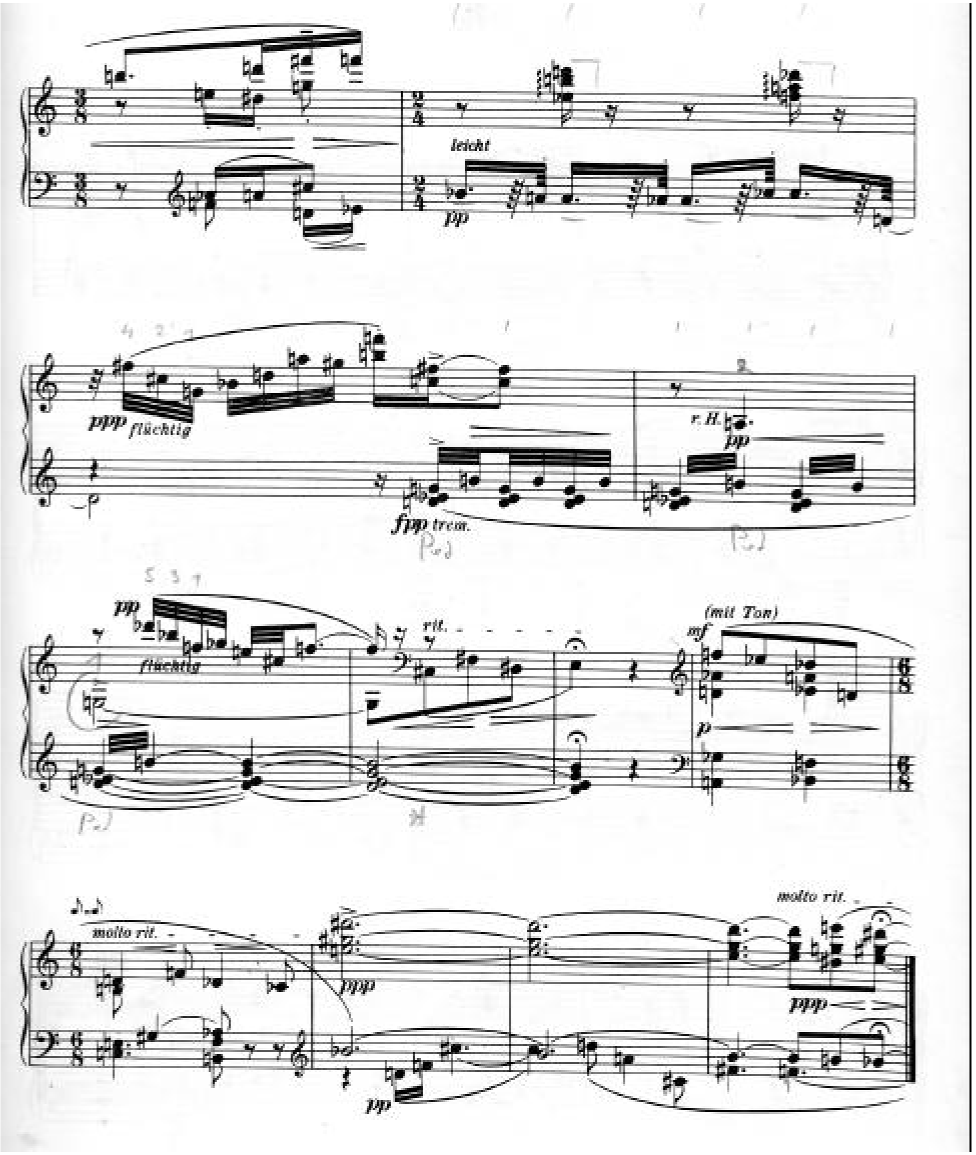


Do you know what the matrix is?

- 4 „usatz“ cells
- permutations (horizontal)
- partial inversion (vertical):
  - Invert the first interval but keep the second one untouched
  - Or, keep the first interval of the cell original and invert the second one

partial inversion

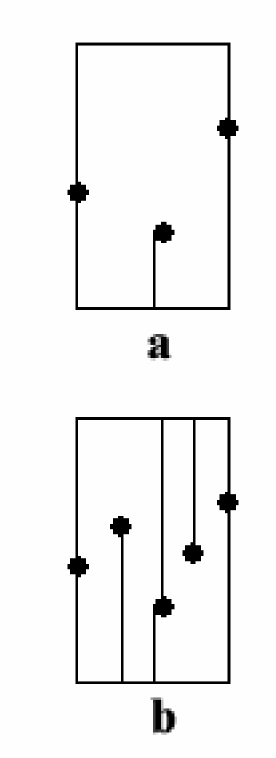




Development of a Composer's Sketchbook

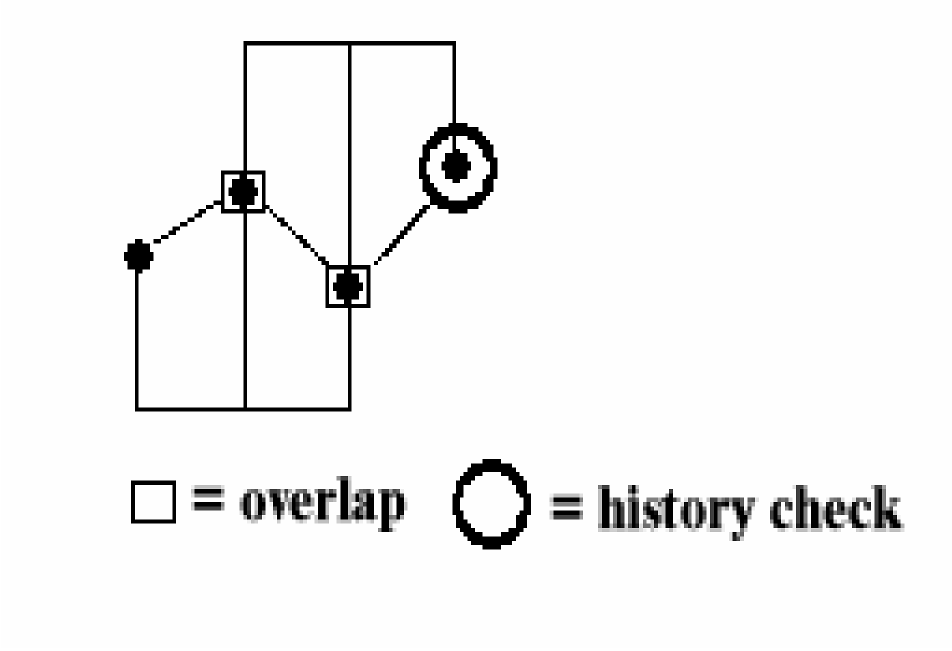
1. Computer Assisted Composition
2. Algorithms

Fractal Chaining



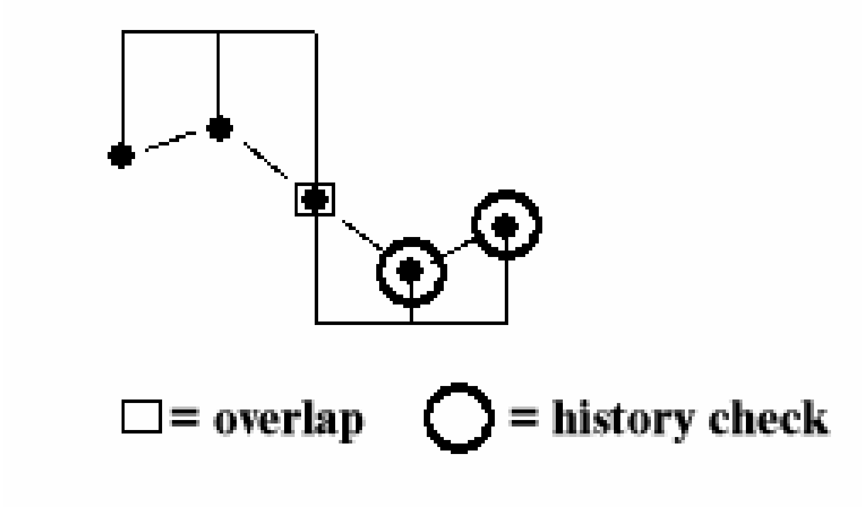
- Generative algorithms building chains from matrix cells
- Replacement of an interval by two different intervals summing up to the original interval (figure a)
- Recursive application of fractal chaining (figure b)

Chain overlapping 2 notes



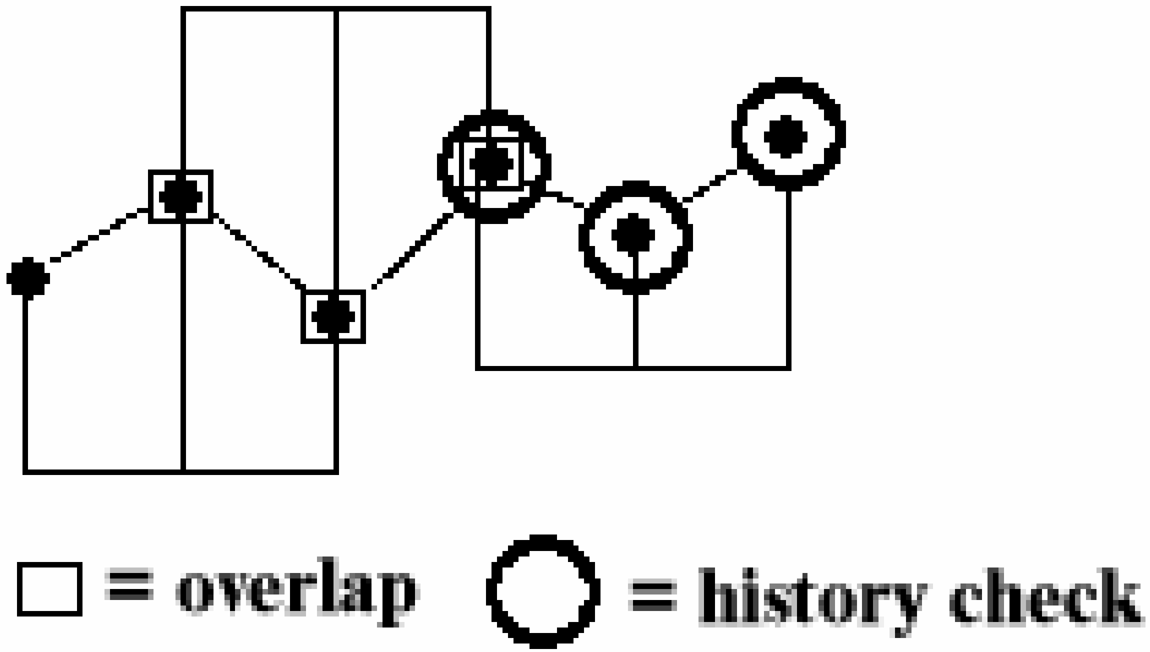
- Looking at the last interval of the sequence
- Search the matrix for a match
- => adding a new note to the sequence
- with or whithout history check: is a new pitch-class added to the sequence or not?

Chain overlapping 1 note

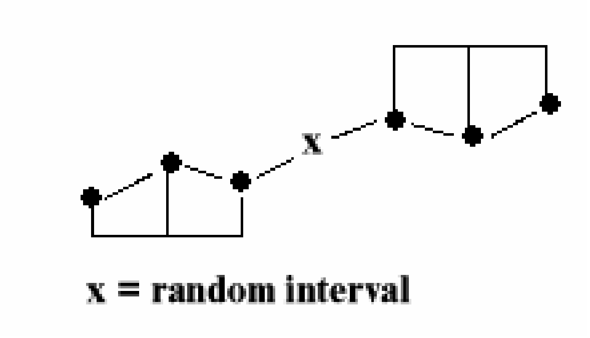


- First taking a random cell from the matrix
- Let one note overlap
- Check or not whether new pitch classes are added or not, in which case the program tries to fit a different cell from the database

Combining both algorithms



Chain without overlap



- take a random first interval from the matrix
- Use the resulting pitch-class as the basis for another cell chosen from the matrix
- No history check

Development of a Composer’s Sketchbook

1. Computer Assisted Composition
2. Algorithms
3. Implementation

Serialize it

```
84
CCompStaff 72 72 720 107 7 2 72 107

CCompClef 72 72 0 0 1 3

CCompNote 102 72 0 0 1 7 76

CCompNote 123 72 0 0 1 7 72

CCompNote 144 72 0 0 1 7 77

CCompNote 165 72 0 0 1 7 82

CCompNote 186 72 0 0 1 7 81
```

the database

```
////////////////////////////////////////////////////////////////////////////// ////////////////
// Name:        MakeMelody.h
// Purpose:     Class for calculating melodies
// Author:      Georg Boenn
// Modified by:       Georg Boenn
// Created:           07/01/05
// Modified:          Sun 17 Apr 2005 04:59:26 PM BST
// Copyright:   (c) Georg Boenn
// Licence:     GNU General Public License v2
////////////////////////////////////////////////////////////////////////////// ////////////////

#ifndef __MakeMelody_h__
#define __MakeMelody_h__

#include "LList.h"
#include "DList.h"
#include "CBuffer.h"
#include "Random.h"

const int CELLDB_MAX = 122;

const int b21[3] = {62,60,63};
const int b22[3] = {61,62,60};
const int b23[3] = {63,62,64};
const int b24[3] = {59,61,60};
const int b25[3] = {63,61,62};
const int b26[3] = {60,63,61};
const int b27[3] = {60,64,61};
const int b28[3] = {63,60,64};
const int n21[3] = {60,64,62};
const int n22[3] = {60,63,62};
...
```

```
MakeMelody::MakeMelody()
{

        bptr[0] = new Buffer(b21,3);
        bptr[1] = new Buffer(b22,3);
        bptr[2] = new Buffer(b23,3);
        bptr[3] = new Buffer(b24,3);
        bptr[4] = new Buffer(b25,3);
        bptr[5] = new Buffer(b26,3);
        bptr[6] = new Buffer(b27,3);
        bptr[7] = new Buffer(b28,3);

        bptr[8] = new Buffer(n21,3);
        bptr[9] = new Buffer(n22,3);
        bptr[10] = new Buffer(n23,3);
        bptr[11] = new Buffer(n24,3);
        bptr[12] = new Buffer(n25,3);
        bptr[13] = new Buffer(n26,3);
        bptr[14] = new Buffer(n27,3);
        bptr[15] = new Buffer(n28,3);

        bptr[16] = new Buffer(b31,3);
        bptr[17] = new Buffer(b32,3);
        bptr[18] = new Buffer(b33,3);
        bptr[19] = new Buffer(b34,3); // etcetera
```

Development of a Composer’s Sketchbook

1. Computer Assisted Composition
2. Algorithms
3. Implementation
4. Future?

Future?

- MIDI and RealTime Audio output on all platforms
- Rhythm classes
- Context-free grammar editor
- Polyphony
- Chord database
- Advanced Notation capabilities

References

- www.wxwindows.org
- www.mididesign.com
- www.boenn.de/composer