Pitch-class Set design in SuperCollider

Lucas SAMARUGA
lsamaruga@becarios.unq.edu.ar

Oscar Pablo DI LISCIA
odiliscia@unq.edu.ar

Universidad Nacional de Quilmes

Argentina

PCS Theory

(Allen Forte, 1974, The Structure of Atonal Music)
(Robert Morris, 1985, Pitch-Class composition)

<u>1-Taxonomy</u>

(Set Classes)

2-Structure

(Properties of Set Classes)

3-Relations

(Similarity)

<u>4-PCS design</u>

(Chains, Combinatorial Matrices)

1-Set Classes (fundamental concepts)

-**Pitch Class**(PC): a class formed by all the members (by octave equivalence) of each one of the twelve musical notes.

-**Integer representation of PC**: 0=C, 1=C# or Db, 2=D and so on. Usually 10 is replaced by "A" and 11 by "B" for printing neatness.

-**Different combinations of the 12 PC**: 2^{12} =4096. (1 of 0 PCs, 12 of 1 PC, 66 of 2 PC and so on).

-**Pitch-Class Set** (PCS): a set of PC (a particular PC combination). Ordering and repetition are not taken in account.

-**Transposition** (Tn) of a PC: Tn(a)= a+n (%12).

-**Inversion** (I) of a PC: I(a)= 12-a (%12).

-**Class equivalence**: If a **PCS** can be reduced to other **PCS** by **T** and/or **IT**, both **PCS** belongs to the same **Set Class** (SC).

-Set-Class names: Cardinal and ordinal numbers.
Ej. **PCS**={0, 1,6} **Set Class**=3-5

Thus, the 4096 different **PCS**, can be classified in 224 different **SC**.

Cardinal Number	Different SCs	Name
0	1 (“null set”)	0--1
1	1	1--1
2	6	2-1 TO 2-6
3	12	3-1 TO 3-12
4	29	4-1 TO 4-29
5	38	5-1 TO 5-38
6	50	6-1 TO 6-50
7	38	7-1 TO 7-38
8	29	8-1 TO 8-29
9	12	9-1 TO 9-12
10	6	10-1 TO 10-6
11	1	1-1 TO 1-1
12	1 (“aggregate”)	12--1
TOTAL	224	

4–PCS design

Combinatorial Matrices (CM, R. Morris, 1984, 1987)

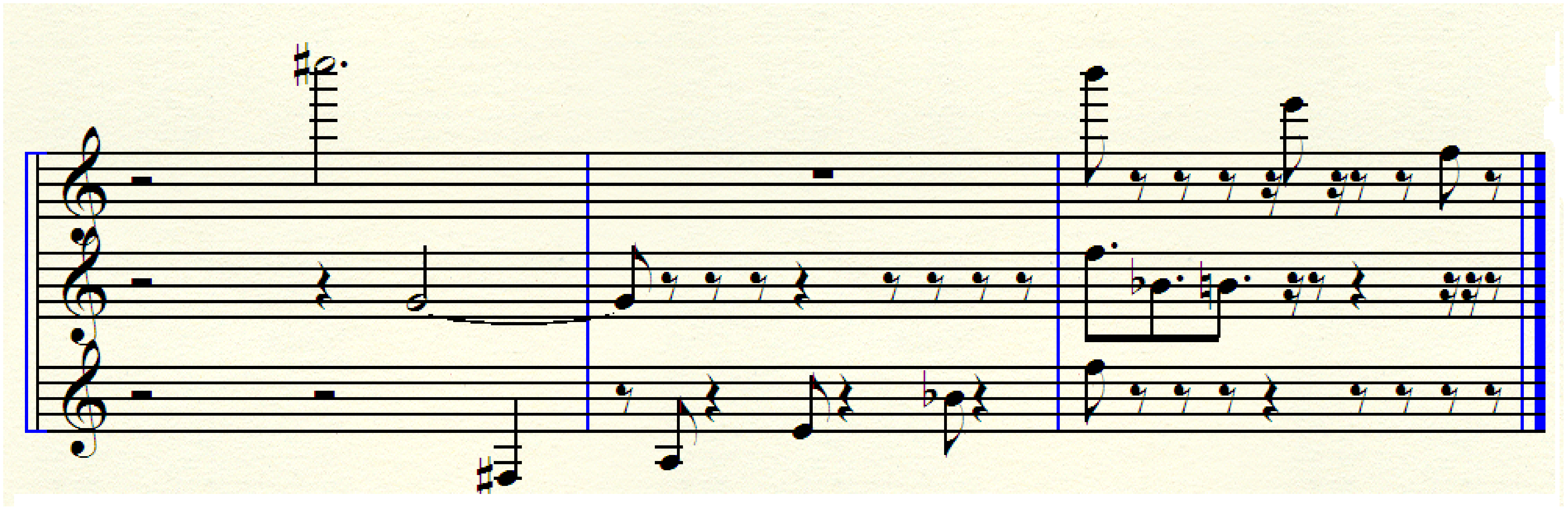
CM are two-dimensional arrays of PCS.

1	4	6	8
4	6	8	1
6	8	1	4
8	1	4	6

	461		8
4		861	
6	8		41
81		4	6

Composers may use various sonic dimensions in order to achieve both variety and coherence. See one of the possible musical realizations of the following CM, scored for Flute, Oboe and Clarinet:

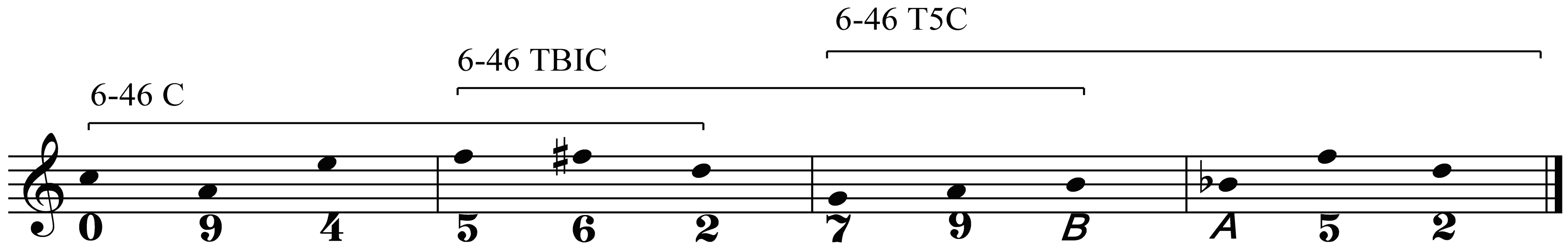
1		B	45
7		5A	B
6	A94		



Combinatorial Matrices generation from Chains of PCS

A chain is a succession of PCS that, being considered in adjacent pairs, form a PCS of a particular SC referred to as *norm*.

Chain: < {094} {562} {79B} {A52} >
norm= 6–46



A chain with a unique norm may be taken as a basis for constructing a CM with the same norm.

PCS chain: < {01} {268} {07} {15B} {67} {028} {16} {57B} >

01	**268**		
	07	**15B**	
		67	**028**
57B			**16**

The distribution of the PCS in the resulting CM may be further improved through swapping operations

1	**02**	**6**	**8**
B	**7**	**15**	**0**
0	**8**	**7**	**26**
57	**6**	**B**	**1**

Constructing chains from binary partitions of a PCS

5-15 {0,1,2,6,8}			
Partitions 1/4			
A	0|1268	1-1	4-16
B	1|0268	1-1	4-25
C	2|0168	1-1	4-16
D	6|0128	1-1	4-5
E	8|0126	1-1	4-5
Partitions 2/3			
F	01|268	2-1	3-8
G	02|168	2-2	3-9
H	06|128	2-6	3-5
I	08|126	2-4	3-4
J	12|068	2-1	3-8
K	16|028	2-5	3-8
L	18|026	2-5	3-8
M	26|018	2-4	3-4
N	28|016	2-6	3-5
O	68|012	2-2	3-1

F	01|268
RT6K	268|07
T1IK	07|15B
RT7IF	15B|67
T6F	67|028
RK	028|16
T7IK	16|57B
RT1IF	57B|01

PCS chain: < {01} {268} {07} {15B} {67} {028} {16} {57B} >

01	**268**		
	07	**15B**	
		67	**028**
57B			**16**

Scoring partition candidates

$$score(cand_i) = \frac{\sum_{n=0}^{C_{i-1}} dist(pc_n, cand_i)}{(S)C_i}$$

S	= chain size (number of positions)
C_i	= cardinality of the *ith* PCS to be added
$dist(pc_n, cand_i)$	= $S - (pos(pc_n, cand_i) + 2)$
$pos(pc_n, cand_i)$	= the last position in which the pc_n of $cand_i$ was found (i=0 to $S-1$, and n=0 to C_i). If the pc_n is not found, then $pos(pc_n, cand_i)=0$.

PCSLB-SC

-Developed by Lucas Samaruga with the advice of Pablo Di Liscia.

-Based on PCSLIB (by Pablo Di Liscia and Pablo Cetta, 2006) for *Pure Data.*

-Four main classes:

1.*PCS* Class. Defines a particular PCS (together with its properties and operations)

2.*PCSChain* Class. Defines a *chain* of *PCS* and its elaboration methods.

3.*PCSMatrix* Class. Defines a *combinatorial matri*x of *PCS* (together with its particular properties, generation methods and operations).

4-*SCTable* Class. Holds the *Set-class* (SC) table used for information retrieval from the *PCS* class.

PCS Class

-Is the core class of the quark, it holds all the operations and properties related to the PCS theory.
-A *PCS* can be built from its symbolic table name or can be created from an array of numbers, like *PCS[0, 1, 3, 5, 6].*

PCS instance methods to get the following data.

- *prime form*,
- *normal order*
- *name*
- *Z pair*
- *interval-class vector*,
- *invariance vector*

Basic operations aside of the inherited set operations.

- *twelve-tone operators (TTO)*
- *relations*
- *similarity*
- *status*

PCSChain Class

Is used to build chains with the procedures already explained.

PCSMatrix Class

Different methods for building CM
- -from arrays (a "free" matrix),
- -from chains
- -from Set Classes, to build different Morris's CM types
- -from TTOs

CM transformation
- -swapping
- -transposition
- -multiplications
- -inversion
- -rotation

CM Analysis
- -sparseness
- -fragmentation
- -PC histogram

Constructing *chains* with more than one *norm*

Vertical norm=X
Horizontal norms= a, b, c, d, e

Chain

a		*b*		*c*		*d*		*e*	
**	***	**	***	**	***	**	***	**	***
	x		*x*		*x*		*x*		

Resulting CM

(sc)	*x*	*x*	*x*	*x*	*x*
a	**	***			
b		**	***		
c			**	***	
d				**	***
e	***				**

Create PCS out of their Set Class Names:

```
a = PCS('5-1');
b = PCS('5-21');
c = PCS('5-35');
d = PCS('5-7');
e = PCS('5-33');
x = PCS('5-12'); // 5-Z12
```

Create a *PCSChain,* set its initial norm and add a starting partition.

```
~chain = PCSChain.new.norm_(a);
~chain.candidates(false);
~chain.addCand(7);
```

Construct the chain by adding the best partitions candidates with the following ad hoc algorithm

```
[x, b, x, c, x, d, x, e].do({ arg pcs;
	~chain.norm = pcs;
	~chain.candidates(false);
	~chain.candList.notEmpty.if({
		~chain.addCand(
			~chain.scores.indexOf(
			~chain.scores.maxItem
		);
	);
	}, {
	"candidates for %"
	.format(pcs.name).throw;
	});
});
```

Create a PCSMatrix from the generated chain.

```
~matrix = PCSMatrix.fromChain(~chain);
```

set-class	4-11	X (5-Z12)	X (5-Z12)	X (5-Z12)	X (5-Z12)
A (5-1)	03	124			
B (5-21)		67	3AB		
C (5-35)			14	68B	
D (5-7)				5A	349
E (5-33)	02A				68

Improve the CM by swapping and duplicate one PC to keep the first column in norm.

```
~matrix.swapping;
~matrix.addAt(3, 0, PCS[9]);
```

set-class	X (5-Z12)	X (5-Z12)	X (5-Z12)	X (5-Z12)	X (5-Z12)
A (5-1)	02	1	4		3
B (5-21)	A	7	3	B	6
C (5-35)		6	1B	8	4
D (5-7)	39	4		5A	9
E (5-33)	0	2	A	6	8