Anatomy of the six-part all-partition array as used by Milton Babbitt

Bemman, Brian; Meredith, David

Publication date:
2014

Document Version
Early version, also known as pre-print

Anatomy of the Six-part All-partition Array as used by Milton Babbitt: Preliminary Efforts Towards a Computational Method of Automatic Generation

Brian M. Bemman
David Meredith

Aalborg University
Dept. of Architecture, Design and Media Technology

RMA Music and Mathematics Study Day
Saturday, 12 April, 2014

Intention

- Research represents a preliminary effort at using computational methods to automatically generate and parse all-partition array structure.

1. Formally define the internal structures of six-part, all-partition arrays.

2. Provide a template representative of the organization of their pitch-class structure based on additional formalized constraints.

3. Demonstrate the computational difficulties observed in initial attempts to automatically parse all-partition array structures.

1. Background

Some definitions…

What is an all-partition array?

Lyne $(x,\overline{x})$	2 / 3 10 11 9		9
Lyne $(\overline{x},x)$	1 8		8
Lyne $(y,\overline{y})$	0	0 11 1 5	6
Lyne $(\overline{y},y)$	7 /	10 9 4 2 8 3 6 7	7
Lyne $(z,\overline{z})$	6 5		5 0 10 4 11 2 3
Lyne $(\overline{z},z)$	4 /		1
	$52^2 1^3$	84	71^5

All-combinatorial Hexachords

- All-combinatorial hexachords –

Let *a* be $\{a_0, a_1, \ldots, a_5\}$ then *a* is *all-combinatorial iff*

$$\exists\, w, x, y, z :$$

$a \xrightarrow{P_w, I_x, R_y, RI_z} a$ ex. $\{0,1,2,6,7,8\} \xrightarrow{P_6} \{6,7,8,0,1,2\}$

AND

$\exists\, x : \quad a \xrightarrow{I_x} \overline{a}$ ex. $\{0,1,2,6,7,8\} \xrightarrow{I_5} \{5,4,3,11,10,9\}$

Hexachordally Combinatorial Rows

- Hexachordally combinatorial rows, *h* –

Let *A* be $(a_0, a_1, \ldots, a_{11})$, *let B* be $(b_0, b_1, \ldots, b_{11})$ and

Let *a* be $\{a_0, a_1, \ldots, a_5\}$, *let b* be $\{b_6, b_7, \ldots, b_{11}\}$ then

$$A \ h \ B \text{ iff } a = b$$

Integer Partition vs. Integer Composition

- In number theory, an **integer partition** is a way of representing an integer *n* as an unordered sum of positive integers.

When $n = 12$

$$3 + 3 + 2 + 2 + 1 + 1 \equiv 2 + 3 + 2 + 1 + 3 + 1$$

- An **integer composition** is an *ordered* integer partition. In the above example, these would not be equivalent.

- In an all-partition array, we must include zero in many integer compositions. We call such instances, **weak integer compositions**.

When $n = 12$

$$6 + 6 + 0 + 0 + 0 + 0 \neq 0 + 6 + 0 + 0 + 6 + 0$$

- All compositions can be trivially considered weak and are also infinitely so. In an all-partition array, these are **bounded** by part with the number of summands corresponding to the number of parts.

1. Background

✓ Some definitions…

⇨ What is an all-partition array?

Lyne $(x, \overline{x})$	2 / 3 10 11 9		9
Lyne $(\overline{x}, x)$	1 8		8
Lyne $(y, \overline{y})$	0	0 11 1 5	6
Lyne $(\overline{y}, y)$	7 /	10 9 4 2 8 3 6 7	7
Lyne $(z, \overline{z})$	6 5		5 0 10 4 11 2 3
Lyne $(\overline{z}, z)$	4 /		1
	$52^2 1^3$	84	71^5

…a twelve-tone structure organized into pairs of hexachordally combinatorial rows and then parsed into a sequence of discrete, vertical aggregates by distinct integer compositions.

All all-partition arrays

- Organization based on the principle of *h*.
- Implicit feature of *h*-related rows that their pairing forms both linear and vertical aggregates, four in total.
- This structure in music theory is called an **array**.

$$
\begin{array}{r|l|}
\text{Row } (x, \overline{x}) & (\overbrace{11, 4, 3, 5, 9, 10}^{\{x\}}, 1, 8, 2, 0, 7, 6) \\
\text{Row } (\overline{x}, x) & (6, 7, 0, 2, 8, 1, \underbrace{10, 9, 5, 3, 4, 11}_{\{x\}})
\end{array}
$$

- A **type** of row refers to its hexachord content. A row of type $(x, \overline{x})$ is constructed from a hexachord $\{x\}$ and its complement $\{\overline{x}\}$ and is of the same row type as all other $(x, \overline{x})$ rows. When $x \neq y$, a row class contains rows of a different type $(x, \overline{x})$ and $(y, \overline{y})$, however, $(x, \overline{x}) \sim (y, \overline{y})$ under P, I, R, RI.
- The concatenation of linear aggregates (often but not necessarily) of the same row type is referred to as a **lyne**.

$$
\begin{array}{l|l|l|l}
\text{Lyne } (x, \overline{x}) & (\overbrace{11, 4, 3, 5, 9, 10}^{\{x\}}, 1, 8, 2, 0, 7, 6) & (\overbrace{5, 4, 11, 9, 3, 10}^{\{x\}}, 1, 2, 6, 8, 7, 0) & \ldots \\
\text{Lyne } (\overline{x}, x) & (6, 7, 0, 2, 8, 1, 10, 9, 5, 3, 4, 11) & (0, 7, 8, 6, 2, 1, 10, 3, 9, 11, 4, 5) & \ldots
\end{array}
$$

- Lyne pairs are often distinguished from each other by register or in the case of pieces for ensemble, by instrument.

Set-class membership of the D-hexachord

- The number of lynes in an all-partition array is determined by the number of distinct members of its rows' constituent hexachords.
- A row class constructed from two D-hexachords will yield six row types of eight rows each for a total of 48 rows in its row class.

A	(0, 1, 2, 3, 4, 5)
B	(0, 2, 3, 4, 5, 7)
C	(0, 2, 4, 5, 7, 9)
D	(0, 1, 2, 6, 7, 8)
E	(0, 1, 4, 5, 8, 9)
F	(0, 2, 4, 6, 8, 10)

T_0	(0, 1, 2, 6, 7, 8)	T_0I	(4, 5, 6, 10, 11, 0)
T_1	(1, 2, 3, 7, 8, 9)	T_1I	(5, 6, 7, 11, 0, 1)
T_2	(2, 3, 4, 8, 9, 10)	T_2I	(0, 1, 2, 6, 7, 8)
T_3	(3, 4, 5, 9, 10, 11)	T_3I	(1, 2, 3, 7, 8, 9)
T_4	(4, 5, 6, 10, 11, 0)	T_4I	(2, 3, 4, 8, 9, 10)
T_5	(5, 6, 7, 11, 0, 1)	T_5I	(3, 4, 5, 9, 10, 11)
T_6	(0, 1, 2, 6, 7, 8)	T_6I	(4, 5, 6, 10, 11, 0)
T_7	(1, 2, 3, 7, 8, 9)	T_7I	(5, 6, 7, 11, 0, 1)
T_8	(2, 3, 4, 8, 9, 10)	T_8I	(0, 1, 2, 6, 7, 8)
T_9	(3, 4, 5, 9, 10, 11)	T_9I	(1, 2, 3, 7, 8, 9)
T_{10}	(4, 5, 6, 10, 11, 0)	$T_{10}I$	(2, 3, 4, 8, 9, 10)
T_{11}	(5, 6, 7, 11, 0, 1)	$T_{11}I$	(3, 4, 5, 9, 10, 11)

- Discrete vertical presentations of aggregates are distinguished according to the partitioning of members from each lyne into segments.

- For an integer partition of 2 + 2 + 2 + 2 + 2 + 2, its shorthand can be written as 2^6, where the prime denotes segment length and exponent denotes parts.

- When the unordered segments in an integer partition are distributed by lyne, they become ordered and thus form an integer composition.

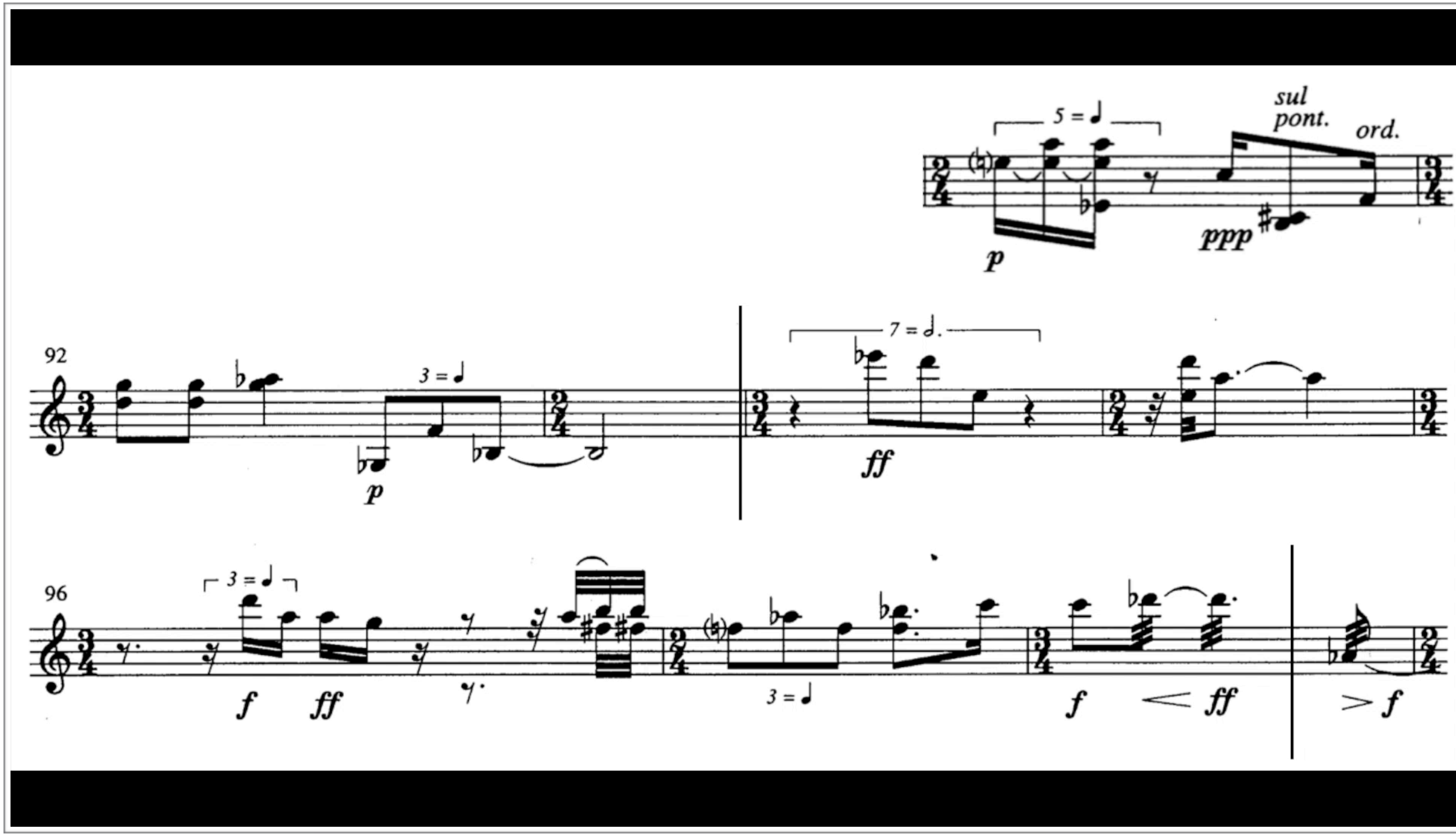
One possible composition sequence

Lyne $(x, \overline{x})$	11 4	3 5	...
Lyne $(\overline{x}, x)$	6 7	0 2 8	...
Lyne $(y, \overline{y})$	5 6	11 1 7	...
Lyne $(\overline{y}, y)$	2 9	10	...
Lyne $(z, \overline{z})$	0 5	4 6	...
Lyne $(\overline{z}, z)$	1 8	9	...
	2^6	$3^2 2^2 1^2$	

One possible block

Lyne $(x, \overline{x})$	2 / 3 10 11 9		9	9 5 4 1	1 6 0 2	2	2 7	7 8
Lyne $(\overline{x}, x)$	1 8		8	8 6 0 7 10	10	11 3 5	5	4 9
Lyne $(y, \overline{y})$	0	0 11 1 5	6			6	6 9 4 10 8 3	3 2
Lyne $(\overline{y}, y)$	7 /	10 9 4 2 8 3 6 7	7	11	11	1	0	0 5
Lyne $(z, \overline{z})$	6 5		5 0 10 4 11 2 3	3		7 9 8	1	1 10
Lyne $(\overline{z}, z)$	4 /		1	2	7 9 3 8 5 4	4 0 10	11	11 6
	$52^2 1^3$	84	71^5	541^3	641^2	$3^3 1^3$	621^4	2^6

- A **block** is the presentation of the aggregate by all lynes.
- A six-part array contains 58 distinct integer partitions into eight blocks.
- Musically, there are just as many ways of articulating block boundaries as obscuring them. Nonetheless, block boundaries signal both the commencement of new rows and salient structural divisions.



Lynes 1, 2

Lynes 3, 4

Lynes 5, 6

Integer compositions…

2^6 | 6^2 |

Intention

- Research represents a preliminary effort at using computational methods to automatically generate and parse all-partition array structure.

1. Formally define the internal structures of six-part, all-partition arrays.

2. Provide a template representative of the organization of their pitch-class structure based on additional formalized constraints.

3. Demonstrate the computational difficulties observed in initial attempts to automatically parse all-partition array structures.

2. The Anatomy

- Reductionist approach.
- Concerned less with musical nuance and more with finding internal parts and how these are organized to produce a unified whole.
- A formal description of these parts will allow for the use of computational methods in analyzing current pieces and producing different pieces with the same type of structure.

Some Constraints of Six-part Arrays Types

- Both the **Babbitt array type** and **Smalley array type** fulfill the following basic criteria…

1. Each lyne contains rows of the same type.

2. Lyne pairs are *h* related.

3. All rows are distinct and appear once i.e. *hyper-aggregate*.

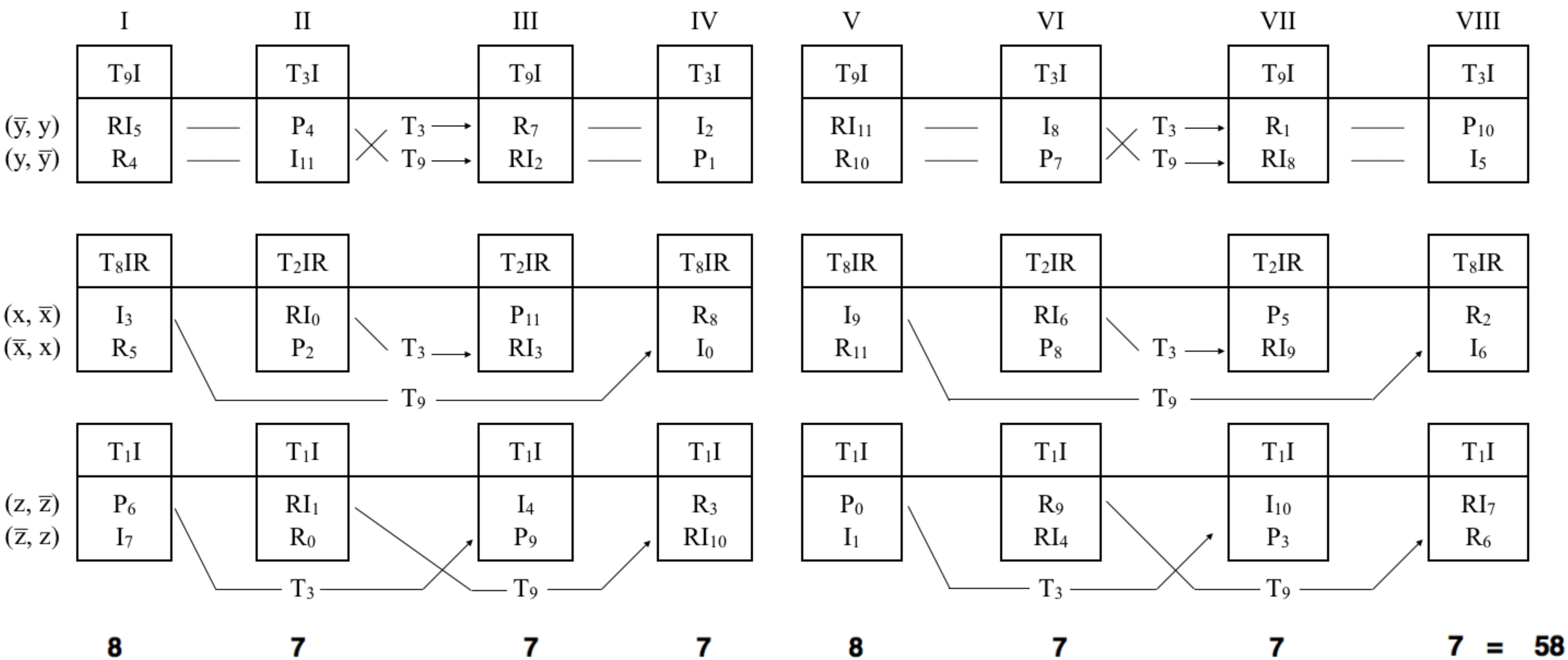
4. Row classes are divided into two T_6 related **sections**, each containing 24 rows.

- …But differ structurally by how *h*-related rows are consistently paired.

5. A Babbitt array: Four distinct *k*-combinations where $r = 2$ (excluding two combinations)

   A Smalley array: Six distinct *k*-combinations where $r = 2$

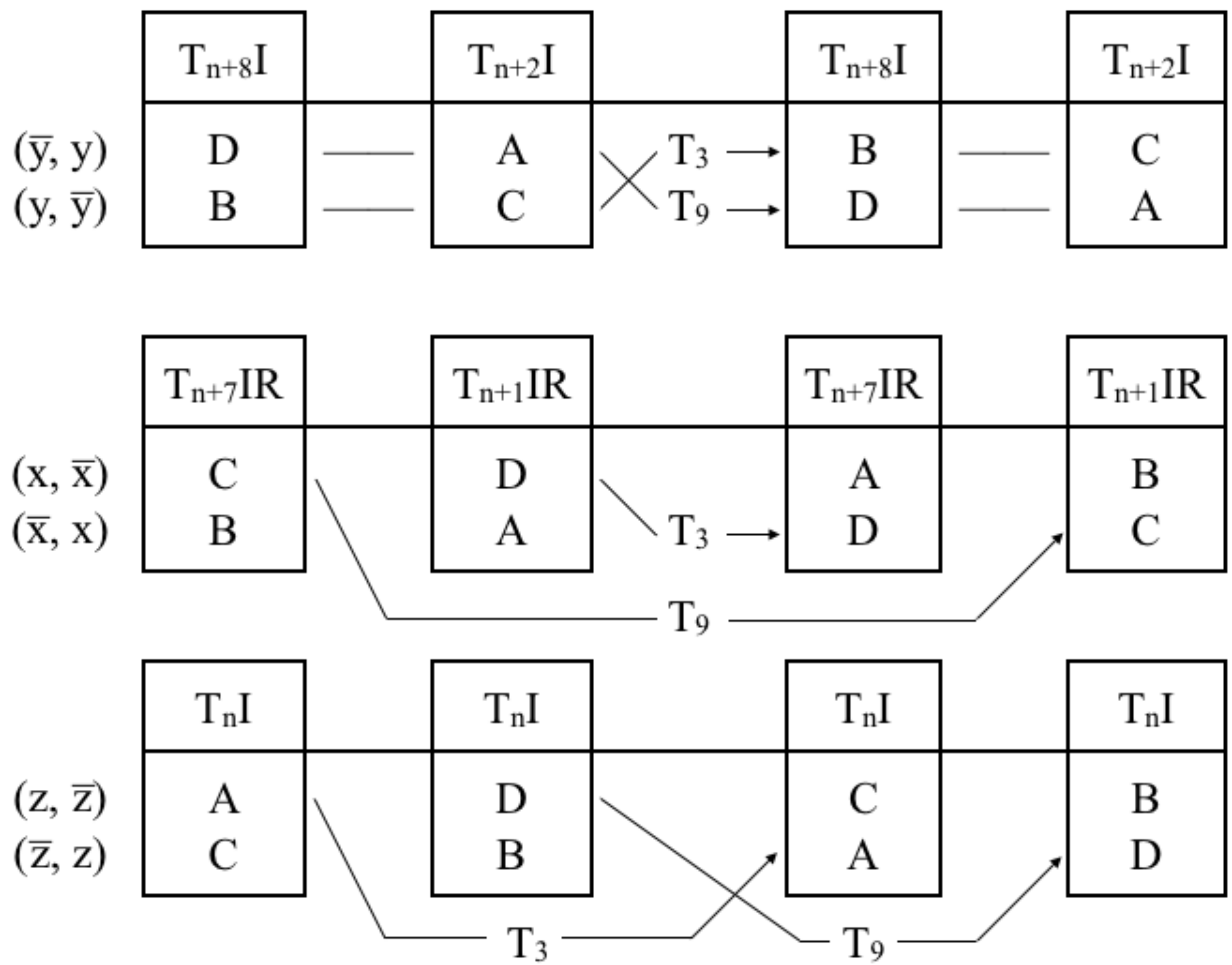
Babbitt Array Type as Found in Babbitt's *About Time*



Complement Transformations T_3 and T_9 and integer partitions below

Template Sufficient to Describe All Babbitt Array Types

$$\{A, B, C, D\} = \{P, I, R, RI\}$$



Found also in Babbitt's Arie da Capo, Tableaux, Playing for Time, and others (all based on different permutations for P0).

$$\{\{x, y\} : x \in \{A, B\}\,,\, y \in \{C, D\}\}$$

D B	A C	B D	C A
C B	D A	A D	B C
A C	D B	C A	B D

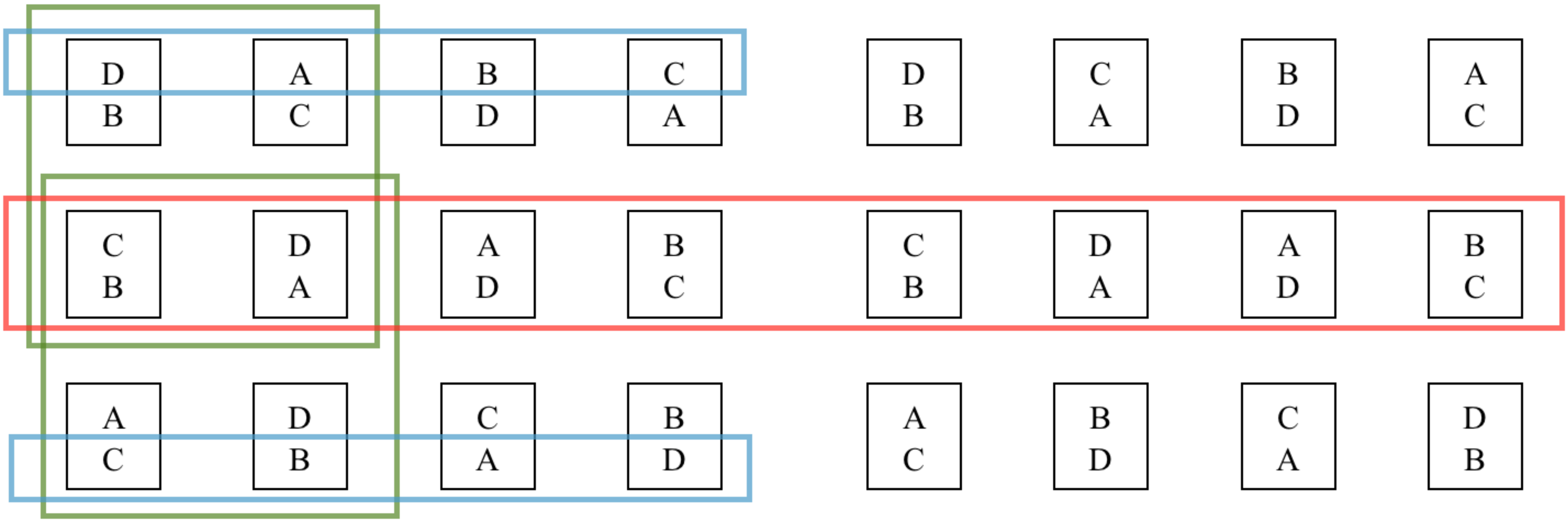
Row pairing constraints by lyne

$$\{\{x_0, y_0\} : x_0 \neq y_0 \wedge (x_0 \in \{A, C\}\,,\, y_0 \in \{A, C\}) \wedge (x_0 \in \{B, D\}\,,\, y_0 \in \{B, D\})\}$$

$$\{\{x_1, y_1\} : x_1 \neq y_1 \wedge (x_1 \in \{A, D\}\,,\, y_1 \in \{A, D\}) \wedge (x_1 \in \{B, C\}\,,\, y_1 \in \{B, C\})\}$$

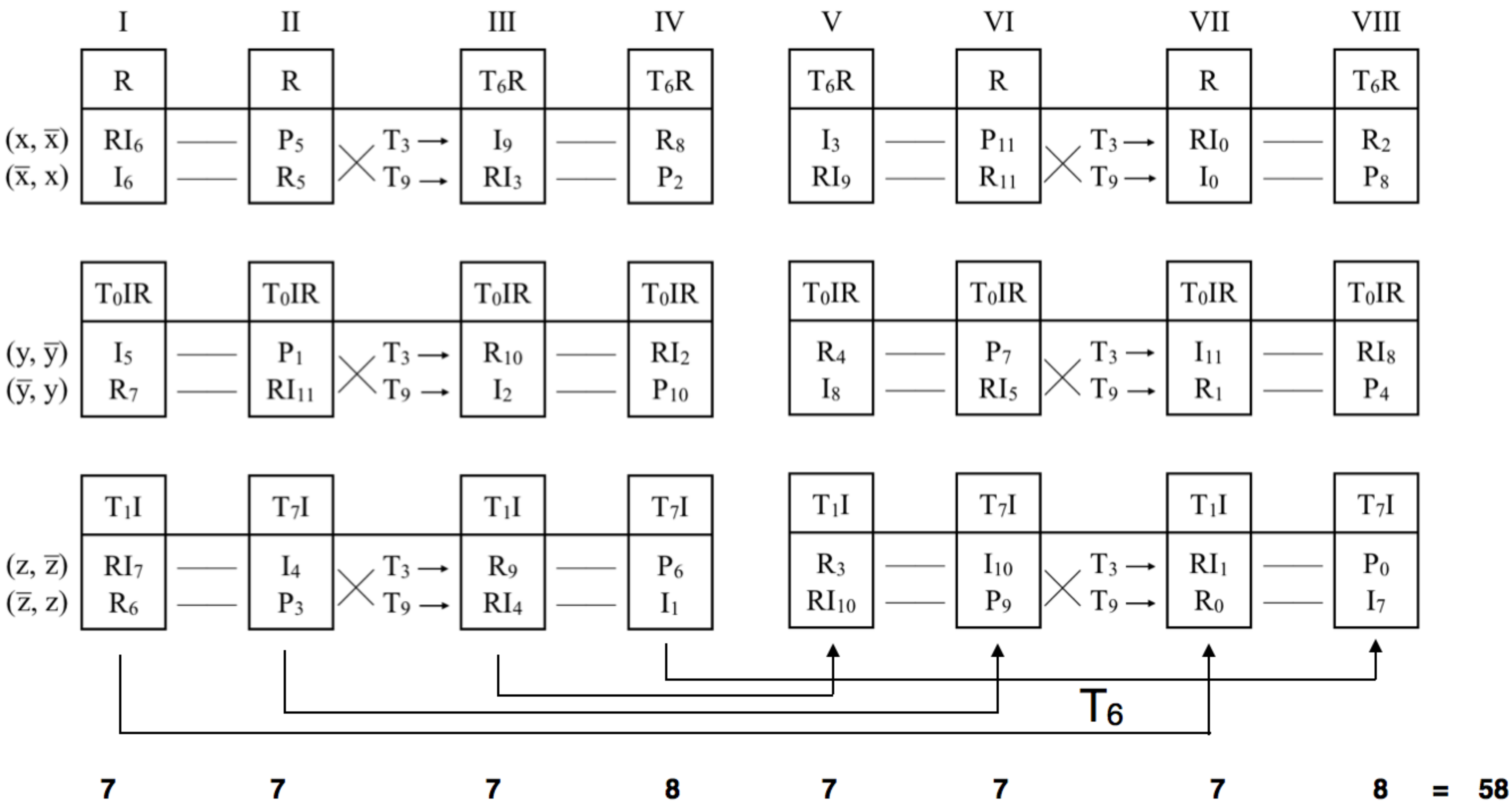
$$\{\{x_2, y_2\} : x_2 \neq y_2 \wedge (x_2 \in \{A, C\}\,,\, y_2 \in \{A, C\}) \wedge (x_2 \in \{B, D\}\,,\, y_2 \in \{B, D\})\}$$

Constraints in Sections



Four Distinct *k*-combinations (excluding {A,B} and {C,D}),
Non-distinct permutations, Retrograde permutations

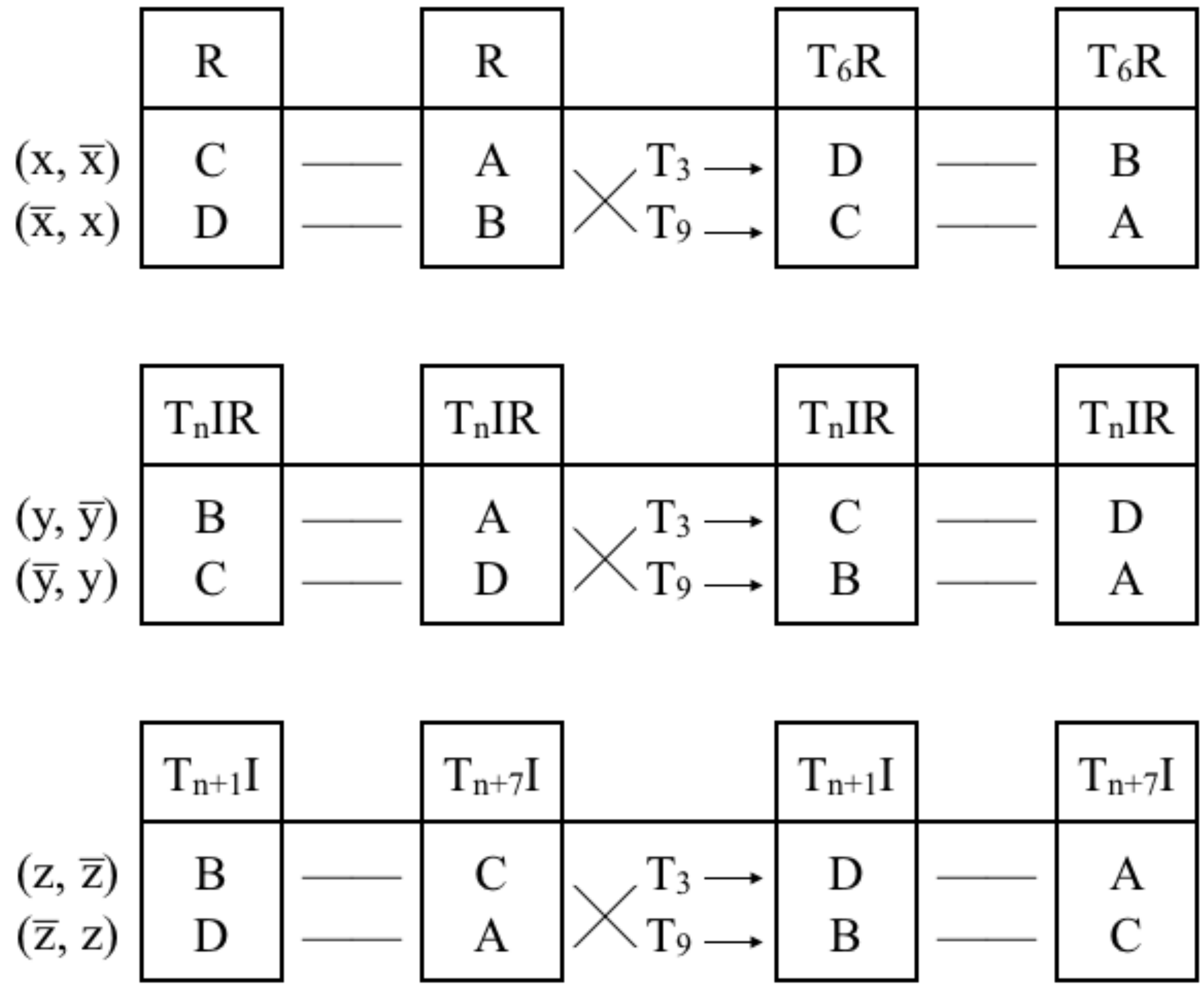
Smalley Array Type as Found in Babbitt's *Sheer Pluck*



Complement Transformations, T_3 and T_9 and integer partitions below

Template Sufficient to Describe All Smalley Array Types

$$\{A, B, C, D\} = \{P, I, R, RI\}$$



Found also in Babbitt’s Joy of More Sextets (translated with reordered lynes and lyne pairs) and Groupwise (with different sequence of integer partitions).

$$\{\{x,y\}: x \neq y \wedge \{x,y\} \subset \{A,B,C,D\}\}$$

C D	A B	D C	B A
B C	A D	C B	D A
B D	C A	D B	A C

Row pairing constraints by lyne

$$\{\{x_0,y_0\}: x_0 \neq y_0 \wedge (x_0 \in \{A,B\}\,,\, y_0 \in \{A,B\}) \wedge (x_0 \in \{C,D\}\,,\, y_0 \in \{C,D\})\}$$

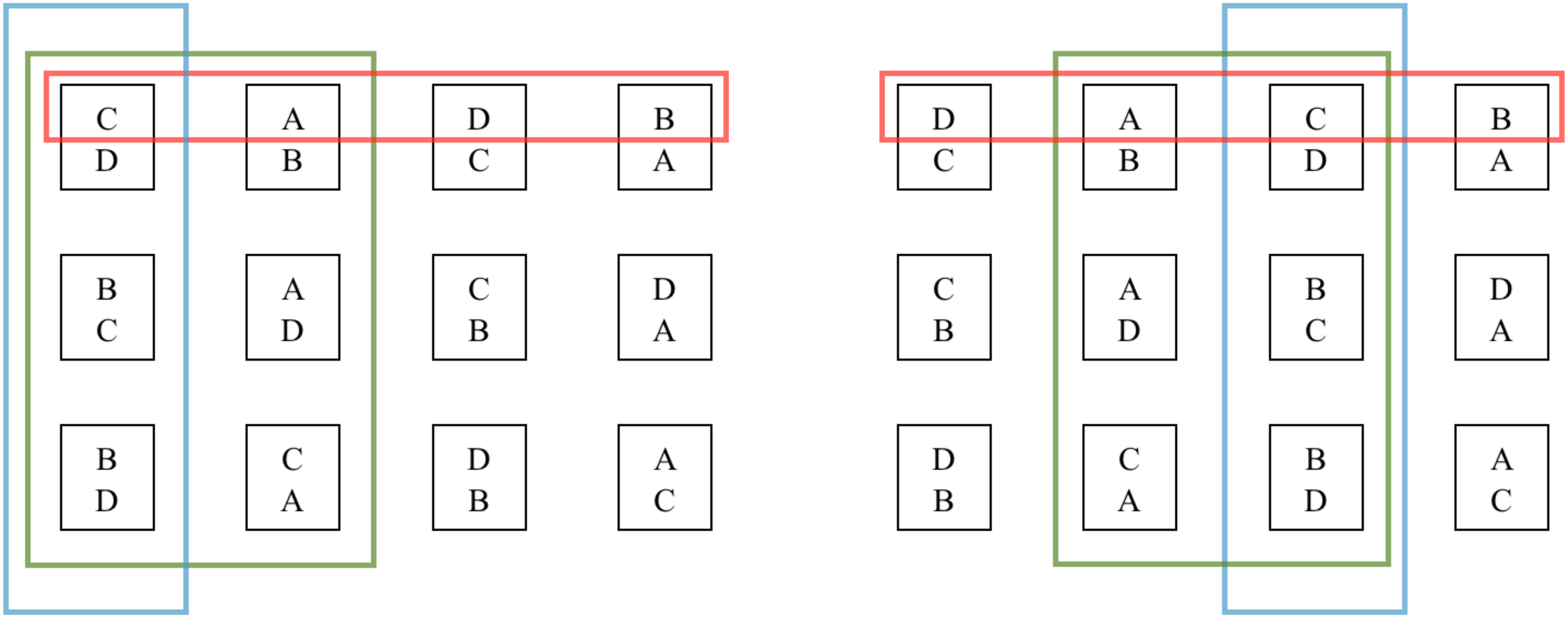
$$\{\{x_1,y_1\}: x_1 \neq y_1 \wedge (x_1 \in \{A,D\}\,,\, y_1 \in \{A,D\}) \wedge (x_1 \in \{B,C\}\,,\, y_1 \in \{B,C\})\}$$

$$\{\{x_2,y_2\}: x_2 \neq y_2 \wedge (x_2 \in \{A,C\}\,,\, y_2 \in \{A,C\}) \wedge (x_2 \in \{B,D\}\,,\, y_2 \in \{B,D\})\}$$

Lyne pair pairing constraints by block

$$\{\{p,q\}: p \neq q \wedge p = \{x_0,y_0\} \cup \{x_1,y_1\} \cup \{x_2,y_2\} \not\ni A\,,\, q = \{x_0,y_0\} \cup \{x_1,y_1\} \cup \{x_2,y_2\}\}$$

Constraints in Sections



Six Distinct *k*-combinations, Distinct permutations,
Non-distinct blocks

Intention

- Research represents a preliminary effort at using computational methods to automatically generate and parse all-partition array structure.

1. Formally define the internal structures of six-part, all-partition arrays.
2. Provide a template representative of the organization of their pitch-class structure based on additional formalized constraints.
3. Demonstrate the computational difficulties observed in initial attempts to automatically parse all-partition array structures.

3. Parsing the Pitch-class Structure

- Automating the organization of pitch-class structure is relatively straightforward. Parsing it however, is not a computationally trivial problem to solve.

- Babbitt used only <u>two</u> distinct sequences of integer compositions (one for each type), why?

Possible Combinations

- The difficulty in parsing this structure can be demonstrated by constructing a formula that determines the distinct number of possible combinations of internal structure = number of calculations required of a program.

- Given constraints1–3…

$$\overbrace{p!}^{\text{partition sequences}} \cdot \underbrace{(s!)^p}_{\text{composition combinations}} \cdot \overbrace{(c! + (c+1)!)}^{\text{within lyne pair ordering}} \cdot \underbrace{(r!)^c}_{h\text{-relation pairings}} \cdot \overbrace{t}^{\text{D-hexachord rows}}$$

where *p* is the number of required integer partitions, *s* is the number of lynes, *c* is the number of lyne pairs (*s*/2), *r* is the number of rows in a given row type, and *t* is a constant of the number of distinct rows built from a D-hexachord (6! · 6!).

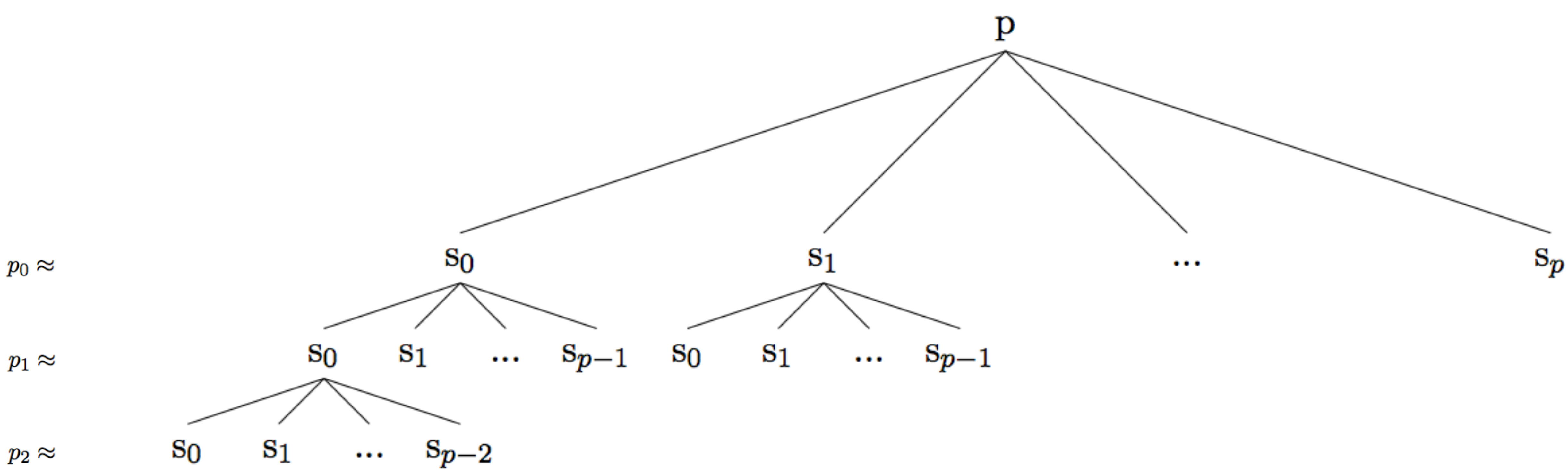
- With the appropriate values assigned for six-part all-partition array…

$$58! \cdot (6!)^{58} \cdot (3! + (3+1)!) \cdot (8!)^3 \cdot 518,400$$

$$n \approx 1.27 \cdot 10^{265}$$

- The value of *n* is far beyond intractable and the culprits are obviously the terms $(6!)^{58}$ and 58!

Brute force search for possible successful integer compositions in *Sheer Pluck*



Where *p* is the number of integer partitions, 58, p_x is the ordinal position of a given partition of *p*, and *s* is a successful integer composition.

Questions for Future Research

- Are there additional constraints in the pitch-class structure that will limit the number of calculations required in finding successful integer compositions?

– Yes, there must be. Pitch-class repetition? Type inform sequence of compositions?

- Is it even possible to generate *all* distinct all-partitions that exist?

– Probably not. Greedy algorithm and heuristics?

Acknowledgements

This talk was prepared while working as a Ph.D. fellow on a collaborative EU project entitled “Learning to Create” (Lrn2Cre8). The Lrn2Cre8 project acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 610859