The computation of the 768-th Laver table

Joseph Van Name

6/13/2021

Preliminaries and summary

Base 10_d integers will be denoted using a d subscript while hexadecimal numbers will be denoted without any subscripts. For example, $32_d = 20, 256_d = 100, 47_d = 2F$.

The largest Laver table computed before 2019_d was the 48_d-th Laver table (by Randall Dougherty). We compute the 768_d-th Laver table, but we have no proof that our computation is actually correct. Our results are obtained simply through experimental calculation.

The computation of A_{768_d} was done in GAP on a Toshiba laptop with a 4 core CPU and 8 GB of memory over the process of about a month. We can get further with better hardware and programming.

Laver tables

Laver tables

The *n-th Laver table* is the unique algebraic structure

$$A_n = (\{1, 2, \ldots, 2^n - 1, 2^n\}, *_n)$$

such that

$$x *_n (y *_n z) = (x *_n y) *_n (x *_n z)$$

and

$$x *_n 1 = x + 1 \mod 2^n$$

for all $x, y, z \in \{1, 2, \ldots, 2^n - 1, 2^n\}$.

Laver tables

Laver tables

The *n-th Laver table* is the unique algebraic structure

$$A_n = (\{1, 2, \ldots, 2^n - 1, 2^n\}, *_n)$$

such that

$$x *_n (y *_n z) = (x *_n y) *_n (x *_n z)$$

and

$$x *_n 1 = x + 1 \mod 2^n$$

for all $x, y, z \in \{1, 2, \ldots, 2^n - 1, 2^n\}$.

For $n > 0$, the operation $*_n$ is non-commutative and non-associative. The Laver tables arise from set theory from rank-into-rank embeddings. If there exists a rank-into-rank cardinal, then the Laver tables are precisely the nilpotent self-distributive algebras generated by a single element.

Examples of Laver tables

The 2×2-Laver table is isomorphic to reduct $(\{0,1\}, \to)$ of the Heyting algebra $(\{0,1\}, \wedge, \vee, \to)$.

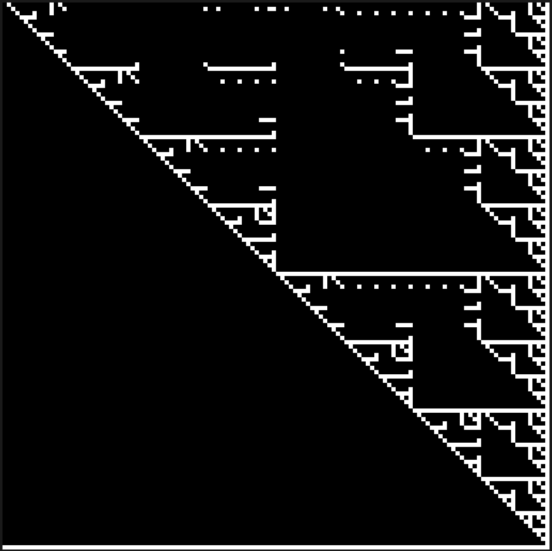
$*_2$	1	2	3	4
1	2	4	2	4
2	3	4	3	4
3	4	4	4	4
4	1	2	3	4

$*_3$	1	2	3	4	5	6	7	8
1	2	4	6	8	2	4	6	8
2	3	4	7	8	3	4	7	8
3	4	8	4	8	4	8	4	8
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

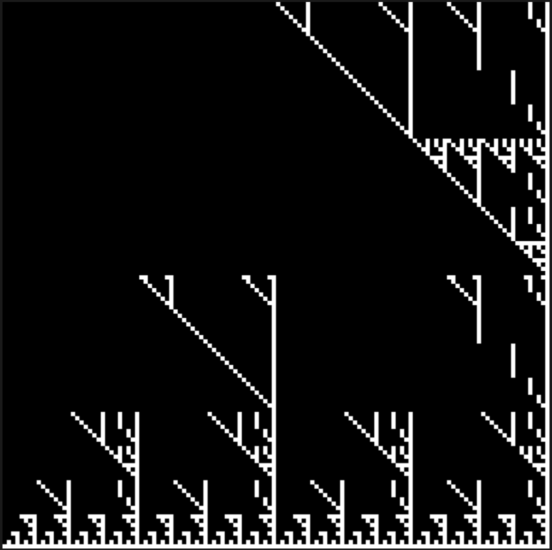
The Laver table A_4

$*_4$	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
1	2	C	E	10	2	C	E	10	2	C	E	10	2	C	E	10
2	3	C	F	10	3	C	F	10	3	C	F	10	3	C	F	10
3	4	8	C	10	4	8	C	10	4	8	C	10	4	8	C	10
4	5	6	7	8	D	E	F	10	5	6	7	8	D	E	F	10
5	6	8	E	10	6	8	E	10	6	8	E	10	6	8	E	10
6	7	8	F	10	7	8	F	10	7	8	F	10	7	8	F	10
7	8	10	8	10	8	10	8	10	8	10	8	10	8	10	8	10
8	9	A	B	C	D	E	F	10	9	A	B	C	D	E	F	10
9	A	C	E	10	A	C	E	10	A	C	E	10	A	C	E	10
A	B	C	F	10	B	C	F	10	B	C	F	10	B	C	F	10
B	C	10	C	10	C	10	C	10	C	10	C	10	C	10	C	10
C	D	E	F	10	D	E	F	10	D	E	F	10	D	E	F	10
D	E	10	E	10	E	10	E	10	E	10	E	10	E	10	E	10
F	F	10	F	10	F	10	F	10	F	10	F	10	F	10	F	10
10	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10

Fractal

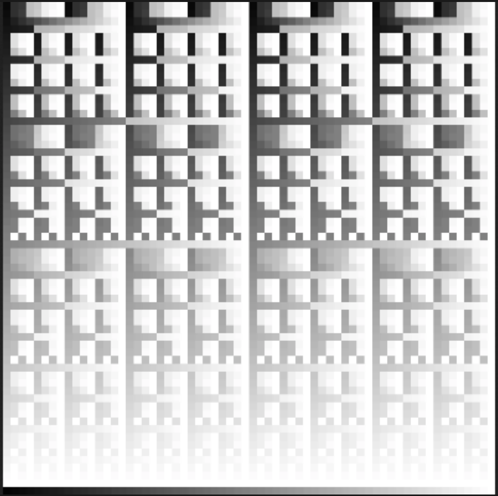


Fractal by inverting binary expansion

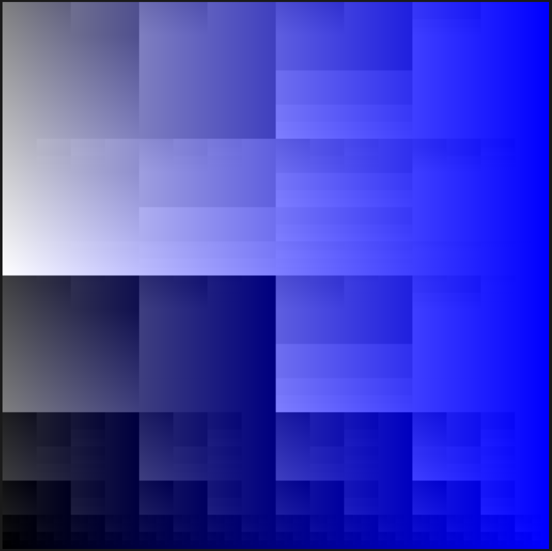


$64_d \times 64_d$-Heat map

The image is taken from googology.wikia.org



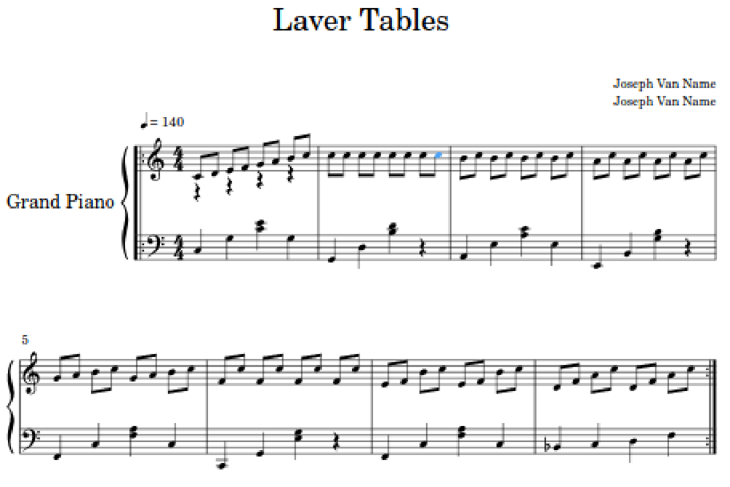
Heat map from inverting binary expansion



IKEA Laver table

8 × 8-Laver tables as music

This is musical evidence for the existence of rank-into-rank cardinals.



Laver tables are challenging to compute.

Proposition

Computing A_n gets more difficult as n gets larger since there is an injective homomorphism $i : A_n \to A_{n+1}$ and a surjective homomorphism $j : A_{n+1} \to A_n$ defined by letting $i(x) = x + 2^n, j(x) = x \mod 2^n$ for all x.

Laver tables are challenging to compute.

Proposition

Computing A_n gets more difficult as n gets larger since there is an injective homomorphism $i: A_n \to A_{n+1}$ and a surjective homomorphism $j: A_{n+1} \to A_n$ defined by letting $i(x) = x + 2^n, j(x) = x \mod 2^n$ for all x.

Theorem (R. Dougherty)

Define a function $f: \omega \to \omega$ by letting $f(n)$ be the least natural number such that $1 *_{f(n)} 2^n = 2^{f(n)}$. If there exists a rank-into-rank cardinal, then the function f is well-defined and the function $n \mapsto f^n(0)$ grows faster than the Ackermann function.

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1							

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2						

We calculate
$8 *_3 2 = 8 *_3 (1 *_3 1) = (8 *_3 1) *_3 (8 *_3 1) = 1 *_3 1 = 2.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3					

We calculate

$8 *_3 3 = 8 *_3 (2 *_3 1) = (8 *_3 2) *_3 (8 *_3 1) = 2 *_3 1 = 3.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3	4				

We calculate
$8 *_3 4 = 8 *_3 (3 *_3 1) = (8 *_3 3) *_3 (8 *_3 1) = 3 *_3 1 = 4.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3	4	5			

We calculate

$8 *_3 5 = 8 *_3 (4 *_3 1) = (8 *_3 4) *_3 (8 *_3 1) = 4 *_3 1 = 5.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3	4	5	6		

We calculate

$8 *_3 6 = 8 *_3 (5 *_3 1) = (8 *_3 5) *_3 (8 *_3 1) = 5 *_3 1 = 6.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3	4	5	6	7	

We calculate

$8 *_3 7 = 8 *_3 (6 *_3 1) = (8 *_3 6) *_3 (8 *_3 1) = 6 *_3 1 = 7.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8							
8	1	2	3	4	5	6	7	8

We calculate
$8 *_3 8 = 8 *_3 (7 *_3 1) = (8 *_3 7) *_3 (8 *_3 1) = 7 *_3 1 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8						
8	1	2	3	4	5	6	7	8

We calculate
$7 *_3 2 = 7 *_3 (1 *_3 1) = (7 *_3 1) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8					
8	1	2	3	4	5	6	7	8

We calculate
$7 *_3 3 = 7 *_3 (2 *_3 1) = (7 *_3 2) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8	8				
8	1	2	3	4	5	6	7	8

We calculate
$7 *_3 4 = 7 *_3 (3 *_3 1) = (7 *_3 3) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8	8	8			
8	1	2	3	4	5	6	7	8

We calculate

$7 *_3 5 = 7 *_3 (4 *_3 1) = (7 *_3 4) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8	8	8	8		
8	1	2	3	4	5	6	7	8

We calculate
$7 *_3 6 = 7 *_3 (5 *_3 1) = (7 *_3 5) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8	8	8	8	8	
8	1	2	3	4	5	6	7	8

We calculate

$7 *_3 7 = 7 *_3 (6 *_3 1) = (7 *_3 6) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7							
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

We calculate
$7 *_3 8 = 7 *_3 (7 *_3 1) = (7 *_3 7) *_3 (7 *_3 1) = 8 *_3 8 = 8.$

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8						
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7					
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7	8				
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7	8	7			
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7	8	7	8		
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7	8	7	8	7	
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6							
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5							
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4							
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3							
3	4	8	4	8	4	8	4	8
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2							
2	3	4	7	8	3	4	7	8
3	4	8	4	8	4	8	4	8
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Computing the Laver table A_3.

Let us compute the multiplication table for A_3.

$*_3$	1	2	3	4	5	6	7	8
1	2	4	6	8	2	4	6	8
2	3	4	7	8	3	4	7	8
3	4	8	4	8	4	8	4	8
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

One can compress the multiplication table for A_n since the rows are periodic. One only needs to compute the white entries in the Laver table since the gray entries give redundant information.

$*_3$	1	2	3	4	5	6	7	8
1	2	4	6	8	2	4	6	8
2	3	4	7	8	3	4	7	8
3	4	8	4	8	4	8	4	8
4	5	6	7	8	5	6	7	8
5	6	8	6	8	6	8	6	8
6	7	8	7	8	7	8	7	8
7	8	8	8	8	8	8	8	8
8	1	2	3	4	5	6	7	8

Dougherty's algorithm for A_{48_d}

Suppose that $2^N \leq n \leq 3 \cdot 2^N$. Then Dougherty has shown that one can easily compute the A_n from the restriction

$$L_{N,n} : \{1, \ldots 2^{2^N}\} \times \{1, \ldots, 2^n\} \to \{1, \ldots, 2^n\}$$

where $L_{N,n}(x, y) = x *_n y$ for all x, y.

Dougherty used this algorithm in the 1990_d's to compute A_{48_d} from the function

$$L_{4,48_d} : \{1, \ldots 65536_d\} \times \{1, \ldots, 2^{48_d}\} \to \{1, \ldots, 2^{48_d}\}.$$

Thresholds

Suppose that $1 \leq x \leq 2^n$. Then let $o_n(x)$ be the least natural number m such that $x *_n 2^m = 2^n$. There is a number $\theta_{n+1}(x)$ called the threshold of x at A_{n+1} such that $0 \leq \theta_{n+1}(x) \leq 2^{o_n(x)}$ and $\theta_{n+1}(2^n) = 0$ such that for every y with $1 \leq y \leq 2^{o_n(x)}$, we have

$$x *_{n+1} y = x *_n y$$

whenever $1 \leq y \leq \theta_{n+1}(x)$ and

$$x *_{n+1} y = x *_n y + 2^n$$

whenever $\theta_{n+1}(x) < y \leq o_n(x)$.

Observations that result in better computation

In most cases, we have $\theta_{n+1}(x) = \theta_{n+2}(x)$.

Observations that result in better computation

In most cases, we have $\theta_{n+1}(x) = \theta_{n+2}(x)$.

For most n, x, there are i, j with $\theta_{n+1}(x) = 2^i - 2^j$.

Observations that result in better computation

In most cases, we have $\theta_{n+1}(x) = \theta_{n+2}(x)$.

For most n, x, there are i, j with $\theta_{n+1}(x) = 2^i - 2^j$.

Exceptional sets $E(n), E^\sharp(n)$, T_n

Let $E(n) = \{x \in \{1, \dots, 2^{n-2}\} \mid \theta_{n-1}(x) \neq \theta_n(x)\}$, and let $E^\sharp(n) = E(n) \cap \{1, \dots, 2^N - 2\}$ where N is the least natural number such that $n \leq 3 \cdot 2^N$, and $T_n = \{(x, \theta_n(x)) \mid x \in E^\sharp(n)\}$.

Observations that result in better computation

In most cases, we have $\theta_{n+1}(x) = \theta_{n+2}(x)$.

For most n, x, there are i, j with $\theta_{n+1}(x) = 2^i - 2^j$.

Exceptional sets $E(n), E^\sharp(n), T_n$

Let $E(n) = \{x \in \{1, \ldots, 2^{n-2}\} \mid \theta_{n-1}(x) \neq \theta_n(x)\}$, and let $E^\sharp(n) = E(n) \cap \{1, \ldots, 2^N - 2\}$ where N is the least natural number such that $n \leq 3 \cdot 2^N$, and $T_n = \{(x, \theta_n(x)) \mid x \in E^\sharp(n)\}$.

One can easily compute A_n from A_{n-1} and the data T_n using Dougherty's algorithm. One computes T_n by initially setting $T_n^{\text{temp}} = \emptyset$ and then by repeatedly modifying T_n^{temp} to correct instances of non-distributivity and other errors until $T_n^{\text{temp}} = T_n$.

Observations that result in better computation

In most cases, we have $\theta_{n+1}(x) = \theta_{n+2}(x)$.

For most n, x, there are i, j with $\theta_{n+1}(x) = 2^i - 2^j$.

Exceptional sets $E(n), E^\sharp(n), T_n$

Let $E(n) = \{x \in \{1, \ldots, 2^{n-2}\} \mid \theta_{n-1}(x) \neq \theta_n(x)\}$, and let $E^\sharp(n) = E(n) \cap \{1, \ldots, 2^N - 2\}$ where N is the least natural number such that $n \leq 3 \cdot 2^N$, and $T_n = \{(x, \theta_n(x)) \mid x \in E^\sharp(n)\}$.

One can easily compute A_n from A_{n-1} and the data T_n using Dougherty's algorithm. One computes T_n by initially setting $T_n^{\text{temp}} = \emptyset$ and then by repeatedly modifying T_n^{temp} to correct instances of non-distributivity and other errors until $T_n^{\text{temp}} = T_n$.

It turns out that computing T_n is relatively easy except when $n = 2^r + 1$ or $n = 3 \cdot 2^r + 1$ for some r (i.e. for us, this is when n belongs to $\{49_d, 65_d, 97_d, 129_d, 257_d, 385_d, 513_d, 769_d, 1025_d, \ldots\}$).

Some early threshold data

$E^\sharp(25_d) =$
$\{E, F0, FE, FF, E00, E0F, E20, E2F, EF0, EF2, EFF, FF00, FF0E\}$.

$E^\sharp(33_d) = \{E, F, E0, E2, EF, F0, F1, FE, FF00,$
FF01,FF0E,FFF0,FFF1,FFF2,FFFC,FFFE}.

$E^\sharp(49_d) =$
$\{FFFF00FE, FFFF00F0, FFFF000E, FFFF0000, FEFFFF, FEFFF2,$
FEFFF0,FEFF2F,FEFF20,FEFF0F,FEFF00,FEF2FF,FEF2F0,FEF20F,
FEF200,FEF0FF,FEF000,FE2FFF,FE2FF0,FE2F0F,FE2F00,FE20FF,
FE2000,FE0FFF,FE0F00,FE00FF,FE00F0,FE000F,FE0000,F0FFFF,
F000FE,F000F0,F0000E,F00000,EFFFF,EFF0E,EFF00,E0EFF,E0EF0,
E0E0F,E0E00,E00FF,E000E,E0000,FFFF,FFFE,FFF0,E}.

$E(n)$ vs $E^\sharp(n)$

$E^\sharp(n)$ is generally much more well-behaved than $E(n)$. For example, $E(18_d) =$
$\{D, E, F, 10, 1F, E0, E2, EF, F0, F1, FE, FF, 100, 10F, 1F0, 1FF, E00,$ E0F, E20, E2F, EF0, EF2, EFF, F00, F0E, FFF, 1000, 10FF, 1200, 120F, 12F0, 12FF, 1F00, 1F0F, 1F20, 1F2F, 1FF0, 1FF2, 1FFF, 2000, 20FF, 2200, 220F, 22F0, 22FF, 2F00, 2F0F, 2F20, 2F2F, 2FF0, 2FF2, 2FFF, 3000, 30FF, 3200, 320F, 32F0, 32FF, 3F00, 3F0F, 3F20, 3F2F, 3FF0, 3FF2, 3FFF, 4000, 40FF, 4F00, 4F0F, 4FF0, 4FFF, 7000, 70FF, 7300, 730F, 73F0, 73FF, 7C00, 7C0F, 7CF0, 7CFF, 7F00, 7F0F, 7F30, 7F3F, 7FC0, 7FCF, 7FF0, 7FF3, 7FFC, 7FFF, 8000, 80FF, 8F00, 8F0F, 8FF0, 8FFF, B000, B0FF, B300, B30F, B3F0, B3FF, BC00, BC0F, BCF0, BCFF, BF00, BF0F, BF30, BF3F, BFC0, BFCF, BFF0, BFF3, BFFC, BFFF, C000, C0FF, C200, C20F, C2F0, C2FF, CF00, CF0F, CF20, CF2F, CFF0, CFF2, CFFF, D000, D0FF, D300, D30F, D3F0, D3FF, DC00, DC0F, DCF0, DCFF, DF00, DF0F, DF30, DF3F, DFC0, DFCF, DFF0, DFF3, DFFC, DFFF, E000, E0FF, E300, E30F, E3F0, E3FF, EC00, EC0F, ECF0, ECFF, EF00, EF0F, EF30, EF3F, EFC0, EFCF, EFF0, EFF3, EFFC, EFFF, F000, F00E, F0FF, F100, F10F, F120, F12F, F1F0, F1F2, F1FF, F200, F20F, F220, F22F, F2F0, F2F2, F2FF, F300,$\dots\}$.

Later threshold data

- $|E^{\sharp}(49_d)| = 48.$
- $|E^{\sharp}(65_d)| = 390.$
- $|E^{\sharp}(97_d)| = 205.$
- $|E^{\sharp}(129_d)| = 70.$
- $|E^{\sharp}(193_d)| = 2515.$
- $|E^{\sharp}(257_d)| = 64167.$
- $|E^{\sharp}(385_d)| = 38808.$
- $|E^{\sharp}(513_d)| = 2320.$

Finding the initial elements in T_n

The most difficult part in calculating T_n is finding the elements in $E^\sharp(n)$. One begins to calculate A_n first by finding a few element in $E^\sharp(n)$ by searching through the following possibilities:

1. Search through integers whose hexadecimal representation contains nothing but $E, F, 0, 2$'s.
2. Search through all elements in $\bigcup\{E^\sharp(m) \mid m < n\}$ which have already been computed.
3. Let $U_{m,0} = \{1\}$, $V_{m,0} = \{1\}$, and let

$$U_{m,r+1} = U_{m,r} \cup \{x *_m y \mid x, y \in U_{m,r}\}$$

and

$$V_{m,r+1} = V_{m,r} \cup \{x *_m y \mid i + j \leq r, x \in V_{m,i}, y \in V_{m,j}\}.$$

Search through elements of the form $U_{m,r}$ or $V_{m,r}$ such that $|U_{m,r}|$ or $|V_{m,r}|$ respectively is small enough to calculate.

Finding new elements in T_n

Suppose that $x, y \in E^{\sharp}_{\text{temp}}(n)$, and x, y have binary expansions $x = a_1 \dots a_r, y = b_1 \dots b_r$. Then we test elements of the following forms to determine whether they belong to $E^{\sharp}(n)$.

1. $x \wedge y, x \vee y$ which are the bitwise meet and join of x and y.
2. $a_1 \dots a_p b_{p+1} \dots b_r$
3. $x *_m y$
4. $a_1 \dots a_p(\neg a_{p+1})a_{p+2} \dots a_r$
5. $a_1 \dots a_p 0^{q-p-1} a_q \dots a_r$
6. $a_1 \dots a_p 1^{q-p-1} a_q \dots a_r$

The Fooooooobonacci sequence

Define the Foobonacci sequence $f_{n,k}$ by letting $f_{n,1} = f_{n,2} = 1$ and $f_{n,i} = f_{n,i-1} *_n f_{n,i-2}$ whenever $i \geq 3$. The Foobonacci terms are used to obtain the composition operation from the application operation $*_n$.

1. $F_{104,1} = 1$.
2. $F_{104,2} = 1$.
3. $F_{104,3} = 2$.
4. $F_{104,4} = 3$.
5. $F_{104,5} = F00000000000000000000000000000000000000000000000000000000000000008$.
6. $F_{104,6} = F0000000000000000000000000000000000000000000000000000000000000000B$.
7. $F_{104,7} = 10000000000000000000000000000000000000000000000000000000000000000$.

Some open directions

1. Recalculate A_{768_d} using similar techniques in order to gain further confidence in the accuracy of the computation of A_{768_d}. It would be great if I could get a co-author to calculate A_{768_d} independently.
2. Compute A_{1024_d} and higher Laver tables.
3. Can one compute A_n for $n >> 768_d$ using a different strategy?
4. Prove that our calculation of A_{768_d} is actually correct. Since the data file of A_{768_d} is several megabytes long, this is actually a very complicated conjecture to state.
5. Compress the data file for calculating A_{768_d}.

References

1. Sheet music for 8×8-Laver table. Composed by Joseph Van Name. https://flat.io/score/606342a5e3c6fc06a2888291-laver-tables
2. Randall Dougherty, Critical points in an algebra of elementary embeddings, Annals of Pure and Applied Logic Volume 65, Issue 3, 15 December 1993, Pages 211-241
3. Randall Dougherty, Critical points in an algebra of elementary embeddings, II Logic: from foundations to applications: European logic colloquium, August 1996 Pages 103–136

Thanks for attending.