

# New Higher-Order Conservation Functions for 1-D Cellular Automata

Barry Fagin (barry.fagin@usafa.af.mil)  
Leemon Baird (leemon.baird@usafa.af.mil)  
Department of Computer Science  
US Air Force Academy  
Colorado Springs, CO 80840  
719-333-3590

**Abstract:** We present several new conservation functions of order 8 and higher for cellular automata. We introduce a new classification scheme using basis sets and non-trivial conservation function extensions.

## INTRODUCTION

We make use of algorithms that permit increased efficiency in the calculation of conservation functions for cellular automata to report conservation laws for 1-D cellular automata of higher order than any previously known. We introduce the notion of *trivial* and *core* conservation functions to distinguish truly new conservation functions from simple extensions of lower-order ones. We give new theorems related to these concepts, and show our use of them to derive more efficient algorithms for finding conservation functions. We then present the complete list of conservation functions up to order 16 for the 256 elementary 1-d binary cellular automata. These include CAs that were not previously known to have nontrivial conservation functions.

Our classification scheme permits finer distinctions between CAs than merely their lowest order of conservation function by grouping functions together with identical sets of basis vectors and by distinguishing *core* functions from *simple extensions*. Of the 88 equivalent 1-dimensional cellular automata, 8 have provably trivial conservation functions only, 47 have conservation functions of various kinds, and 33 remain unclassified. Those that we have classified include CAs with newly discovered conservation functions of order 8, 9, 12, 13 and 14. Our calculations also show there are no CAs with conservation functions of order 15 or 16. We therefore present the complete classification scheme for 1-dimensional cellular automata based on conservation functions up through order 16.

## TERMINOLOGY

A one-dimensional *cellular automaton* is an array of finite size  $W$  of cells containing  $\{0, 1, \dots, s-1\}$ . The array is called the *universe*, and the number of cells it contains is the size of the universe. The universe is considered to have no boundaries, which means in one dimension the end cells are considered adjacent to one another.

The current set of values in each cell is the *state* of the universe. This state changes over time based on a *characteristic function*, parameterized by a neighborhood size  $n$ . To determine the next state of any cell, the state of the  $(n-1)/2$  cells on each side of the current cell, along with the state of the current cell itself is examined, and rules applied that uniquely determine the new state. (Since we are only interested in symmetric neighborhoods,  $n$  will always be odd). The new state of the universe as determined by the parallel application of the characteristic function is called the *successor state*. A 1-dimensional CA with  $s$  states and neighborhood of size  $n$  can be configured in  $s^n$  ways, not all of which are unique.

Let  $\mathbf{x}$  be the state of the universe at a given time, and  $\mathbf{x}_i$  be the state of cell  $i$ . An energy function  $E(\mathbf{x})$  is defined as:

$$E(\mathbf{x}) = \sum_{i=0}^{n-1} f(\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{i+m-1})$$

where subscripts outside the universe wrap around. The energy function is a sum of the energy of  $n$  different regions, each of which contains  $m$  cells. We refer to  $m$  as the *order* of the function. Let  $\text{succ}(\mathbf{x})$  be the successor state of the universe reached on the next time step if it starts in state  $\mathbf{x}$ . We say that  $E$  is conserved, or that  $E$  is an energy conservation function, iff, for all finite universes:

Conserved:  $\forall \mathbf{x} \quad E(\mathbf{x}) = E(\text{succ}(\mathbf{x}))$

This is equivalent to the following [1]:

$$\forall \mathbf{x}, \mathbf{y} \quad E(\mathbf{x}) - E(\text{succ}(\mathbf{x})) = E(\mathbf{y}) - E(\text{succ}(\mathbf{y}))$$

where  $\mathbf{x}$  and  $\mathbf{y}$  are states that differ by exactly one cell.

It can be shown that for neighborhood size of  $n=3$ , we need only analyze the energy of a  $2m+3$ -cell array and the  $2m+1$ -cell array of its successor state in order to determine if a conservation law exists, despite the fact that the universe itself may be arbitrarily large. This is what makes the computation of conservation functions possible.

It is a theorem of Hattori and Takesue [1], later restated in simpler form in [2] and [3], that if we zero the block of cells in

$x$  and  $y$  either to the left or the right of the non-identical cell, our energy calculations will be unaffected. Thus by applying this theorem over all possible  $(m+1)$ -cell states of the universe for a given CA gives a system of  $s^{(m+1)}$  equations in  $s^m$  unknowns, with the right hand side equal to zero. If this system of equations has a solution (that is, if the resulting matrix has a non-empty null space), then an energy conservation function of order  $m$  has been found. If it does not, then no such function exists for that CA.

TRIVIAL, NON-TRIVIAL, AND CORE ENERGY CONSERVATION FUNCTIONS

Clearly energy functions that assign the same value to all states are conserved. We call such functions *trivial*. Of greater interest are non-trivial energy conservation functions. Trivial functions can be eliminated from the solution space through the addition of a small number of additional constraints on the system, using the following two theorems we have proved [3].

**Theorem 1:** For a 1D, 2-state, neighborhood 3 cellular automaton, the following set of  $2^{m-1}$  trivial energy functions form a basis set for all possible energy functions over  $m$  bits, where  $1 < S < 2^{m-1}$ , and expressions like  $1S$  represent a 1 followed by the bits of  $S$  in binary:

$$\begin{aligned}
 f_0(x) &= 1 \\
 f_{m-1}(x) &= 1 \text{ if } x = 000\dots0001 \\
 &= -1 \text{ if } x = 1000\dots000 \\
 &= 0 \text{ otherwise} \\
 f_S(x) &= 1 \text{ if } x = 0S \text{ or } x = 1S \\
 &= -1 \text{ if } x = S0 \text{ or } x = S1 \\
 &= 0 \text{ otherwise}
 \end{aligned}$$

**Theorem 2:** For a 1D, 2-state, neighborhood 3 cellular automaton, a basis set for the set of all conserved linear functions over  $m$  bits can consist solely of energy functions satisfying the constraint  $f(0S) = 0$  for all  $m-1$  element strings  $S$ . In other words, the energy of an  $m$ -cell region can be defined to be zero for any region whose leftmost cell is zero.

Once a non-trivial conservation function of order  $m$  has been found, it can be extended in a very simple way to produce a function of order  $m+1$  by ignoring the newly added cell on either side.

The conservation functions reported in [1] do not distinguish between new functions derived in this way: A conservation function of order  $m$  is by definition a conservation function of order  $n \geq m$  for all  $n$ . We believe, however, that it is useful to distinguish between conservation functions derived from existing ones by ignoring newly added cells and conservation functions that are completely different.

We refer to a non-trivial conservation function for a given CA that cannot be derived from non-trivial conservation of lower order as a *core* function. The core function of lowest order for a given CA is the *primary core* function for that CA. We call functions derived from a core by ignoring adjacent cells *simple extensions*. Simple extensions can be eliminated

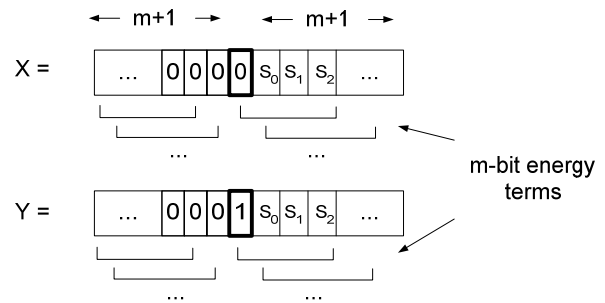
from the solution space through the addition of a small number of constraint equations [3].

Eliminating simple extensions allows for the possibility of a CA having a completely different conservation function that would otherwise not be detected. Notice that through the judicious choice of basis vectors, we are able to reduce the size of the state space matrix by a factor of 2.

AN ALGORITHM FOR CALCULATING CONSERVATION FUNCTIONS OF ORDER  $M$  FOR A GIVEN CA

By examining in detail how each of the energy terms are formed, it is possible to generate the state space matrix without doing explicit lookups and generation of all states and next state vectors. We do this by breaking the state space matrix up into three distinct matrices of different sizes, combining them only at the end when the null space must be calculated.

To see how this is done, consider first the terms in  $E(X) - E(Y)$ , the energy terms in the  $X$  and  $Y$  initial states, and what happens to them as the order  $m$  increases. We have for any  $m$ :



where  $S_0, S_1 \dots$  are state counting bits (we have the leftmost bit change the fastest, for reasons that will become clear shortly). The energy expressions  $E(X)$  and  $E(Y)$  are just the sums of the  $m$ -bit energy terms for the  $X$  and  $Y$  states respectively. It is clear by inspection that all terms in  $E(X) - E(Y)$  cancel except for those involving the center bit. Because we may also assume that  $f(0, \dots) = 0$  for our energy function  $f$ , the matrix for  $E(X) - E(Y)$  becomes

$$-f(1 S_0 S_1 \dots S_{m-2})$$

where the  $S$  bits are as previously described. In other words, each row has a  $-1$  in the column indicated by bits  $1S_0S_1\dots S_{m-2}$ . Making the leftmost 1 implicit in the column numbering, and reversing the column subscripts, we have

$$\text{For } m=1, E(X) - E(Y) = -1$$

$$\text{For } m=2, E(X) - E(Y) =$$

$S_0$	$\frac{10}{-1}$	$\frac{11}{0}$
0	-1	0
1	0	-1

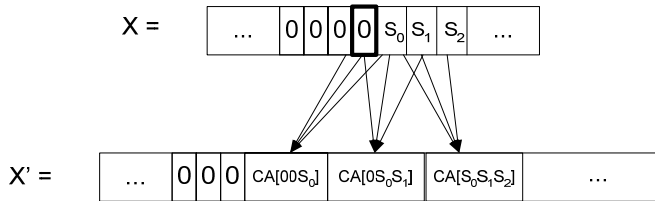
For  $m=3$ ,  $E(X) - E(Y) =$

$S_0S_1$	<u>100</u>	<u>110</u>	<u>101</u>	<u>111</u>
00	-1	0	0	0
10	0	-1	0	0
01	0	0	-1	0
11	0	0	0	-1

and in general  $E(X)-E(Y)$  is given by a negative identity matrix of  $2^{m-1}$  rows and  $2^{m-1}$  columns.

Now consider the energy terms of the successor states,  $E(X')$  and  $E(Y')$ . It is useful to first consider just those CAs that map the neighborhood region 000 to the 0 state. We shall refer to these CAs as zero-preserving automata.

For zero-preserving automata, we have the following relationship between  $X$  and its successor state  $X'$ :



since zero-preserving automata map 000 to 0. (Here  $CA[xyz]$  denotes the appropriate bit of the number of the CA written in binary). A similar analysis holds for states  $Y$  and  $Y'$ .

For  $m = 1$ , we can compute the energy difference of the successor states  $E(X') - E(Y')$  as

$$\{E(X') - E(Y')\}_1 = CA[00S_0] + CA[0S_0S_1] - CA[001] - CA[01S_0] - CA[1S_0S_1]$$

since all other terms cancel, and we assume the energy function  $f$  maps all terms with an MSB of 0 to 0. Let us rewrite the conservation law as

$$E(X) - E(Y) + E(Y') - E(X') = 0$$

Plugging our formula for  $m=1$  above gives

$$-I + CA[001] + CA[01S_0] + CA[1S_0S_1] - CA[00S_0] - CA[0S_0S_1] = 0$$

where  $I$  is the identity matrix. Ignoring signs, we will refer to these terms as  $E_x - E_y$ ,  $D_1$ ,  $D_2$ ,  $D_3$ ,  $R_1$  and  $R_2$  for reasons that will become apparent.

We reorder this expression to group terms with identical numbers of state bits together, giving

$$-I + CA[001] + CA[01S_0] - CA[00S_0] + CA[1S_0S_1] - CA[0S_0S_1] = 0$$

or equivalently

$$E_x - E_y + D_1 + D_2 - R_1 + D_3 - R_2 = 0$$

These six terms, with some bookkeeping, can be represented with three matrices of different sizes that grow in similar ways as  $m$  increases. Their size differences do not matter until the calculation of a null space is required, at which time they can all be made of uniform size and summed.

To see how this process works, first note that as  $m$  increases the energy terms grow to the right. Using the convention of having the leftmost state bit increase first, if we read the CA bit index from right to left, then each increment of  $m$  corresponds to simply copying the existing matrix both above itself and horizontally, and then shifting the individual rows of the new matrix by exactly half the columns if a particular bit of the CA is 1.

For example, consider the  $D_1$  term for  $CA = 50 = 00110010$  in binary. Recall that we read CA bit indices from right to left. For  $m=1$ , we have

$$D_{1_1} = CA[001] = 1$$

For  $m = 2$ , we have

$$D_{1_2} = CA[001] CA[01S_0] = CA[001] CA[010] = 1 \ 0$$

$$CA[001] CA[011] = 1 \ 0$$

$$= \begin{array}{cc} S_0 & \begin{array}{c} 10 \\ 1 \end{array} & \begin{array}{c} 11 \\ 0 \end{array} \\ = & \begin{array}{cc} 0 & 0 \\ 1 & 1 \end{array} & \begin{array}{cc} 0 & 0 \end{array} \end{array}$$

meaning for the state with  $S_0 = 0$  this matrix has an energy term of  $f(10)$ , just as it does for the state with  $S_0 = 1$  (recalling again that the MSB of 1 in the energy term is implicit in the column numbering). Note that we obtained  $D_{1_2}$  from  $D_1$  by copying the rows, doubling the columns, and moving the individual columns over to the right half of the matrix based on the values of the vector  $CA[01S_0] = (CA[010] \ CA[011]) = \{0 \ 0\}$ . We refer to this doubling and shifting operation as *expansion via the vector*  $\{0 \ 0\}$ , and refer to  $\{0 \ 0\}$  as the *expansion vector*. In this case both values in the expansion vector are zero so no shifting of the columns occurs.

For  $m = 3$ , we have

$$D_{1_3} = CA[001] CA[01S_0] CA[1S_0S_1]$$

$$= \begin{array}{ccc} CA[001] CA[010] CA[100] & & 101 \\ = CA[001] CA[011] CA[110] & = & 100 \\ CA[001] CA[010] CA[101] & & 101 \\ CA[001] CA[011] CA[111] & & 100 \end{array}$$

$S_0S_1$	100	110	101	111
00	0	0	1	0
10	1	0	0	0
01	0	0	1	0
11	1	0	0	0

meaning for the state with  $S_0S_1 = 00$  this matrix contributes an energy term of  $f(101)$ , and so forth. This time the expansion vector is  $CA[1S_0S_1] = \{CA[100] CA[110] CA[101] CA[111]\} = \{1 0 1 0\}$ , corresponding to the third non-zero block in the previous figure.

After  $m = 3$ , the expansion vector always  $CA[S_{m-4}S_{m-3}S_{m-2}]$ . So D1 is now completely determined by:

$$\begin{aligned} D1_1 &= CA[001] \\ D1_2 &= \text{expansion\_via}(D1_1, CA[01S_0]) \\ D1_3 &= \text{expansion\_via}(D1_2, CA[1S_0S_1]) \\ D1_m &= \text{expansion\_via}(D1_{m-1}, CA[S_{m-4}S_{m-3}S_{m-2}]) \text{ for } m > 3 \end{aligned}$$

It can be shown that after  $m=1$ , D3 and R2 have identical expansion vectors and may therefore be combined into a single matrix  $C = D3-R2$ . Similarly, after  $m=2$ , D2 and R1 may be combined into a single matrix  $B = D2 - R1$ .  $Ex-Ey$  may be combined immediately with D1 into  $A = Ex-Ey+D1$ . Thus after some preliminary bookkeeping up through  $m=3$ , we need only maintain three energy term matrices A, B and C using only expansion operations with vectors derived from the CA under test. When the desired  $m$  has been reached, we replicate A 4 times and B twice, and then solve the  $2^{m+1} \times 2^{m-1}$  system of equations  $A+B+C = 0$ .

We refer to the matrix  $A+B+C$  as the *state space matrix N*. Finding a conservation function of order  $m$  for a given CA requires calculating the null space of N.

We see from the method of construction that A, B and C are sparse: Each one is the sum of matrices with only one non-zero term per row. This means that N can have at most six non-zero terms per row, out of rows that contain  $2^{m-1}$  entries. In practice, N also contains a large number of redundant rows. Empirical analysis shows that removing all duplicate rows from N reduces its size by about a factor of two.

For an explicit listing of the algorithm using this technique to calculate conservation functions, as well as treatment of non-zero-preserving automata, the reader is referred to [3]. This algorithm has been implemented in MATLAB, and has led to the results we now describe.

#### A TAXONOMY FOR 1-D CELLULAR AUTOMATA BASED ON CONSERVATION LAWS

We now present a complete taxonomy of 1-d CAs based on conservation functions of order  $m \leq 16$ . We use Wolfram's numbering scheme [4] to denote particular CAs, in which the

successive state  $b$  of a cell appears as bit number  $i$  in an 8-bit binary number that designates the CA, where  $i$  is itself a 3-bit binary number determined by the left, center and right cell contents in a CA's next state rules.

When examining all  $s$ -state CAs, symmetry laws can be seen to divide the set into equivalence classes. Viewing the CA through a mirror produces a CA with identical properties, as does replacing state number  $j$  with state symbol  $s-j$ . For binary CAs, repeated application of these laws divides the 256 automata into 88 equivalence classes. For purposes of this discussion, if a single CA is mentioned, it is understood to refer to all CAs in its class. We will normally use the lowest numbered CA of a class to represent it.

Some CAs can be shown to have no non-trivial energy conservation functions. In particular, all CAs with next state functions of  $x0x0x000$  have only trivial energy conservation functions, where  $x$  denotes 0 or 1. For a proof of this, see [3].

#### A. Classification By Lowest Order Conservation Function

We first present a table of CAs grouped by their lowest order conservation function. Table 1 lists all CAs with conservation functions up through order 16, grouped by the order at which their first conservation function emerges. With one exception, this table is identical at lower orders to that in [1], but extends it to significantly higher values of  $m$ .

We note that [1] reports the existence of a CA with a conservation function at  $m=7$ . Wolfram [4] disagrees, and our results support his conclusions. We believe the entry in Table 1 in [1] for CA 19h and its equivalence class at  $m=7$  should be 0.

There are 3 CA equivalence classes with conservation functions of order 1: The shifter (170 decimal/aa hex), the identity (208/cc), and 184/b8. 184/b8 is discrete asymmetric exclusion, and has been extensively studied [5].

At  $m=2$  there are 11 equivalence classes with second order conservation properties, 8 at  $m=3$ , and so forth. The results for  $m \geq 8$  are, to our knowledge, new ([6] reports higher order results for reversible CAs only), as are the basis functions identified in the next section.

#### B. Classification By Conservation Function Basis

We can gain more insight into the structure of CA conservation functions by examining the basis of the associated null space, noting both the number of dimensions and the basis vectors themselves. The tables in this section show the number of dimensions in the null space (which is equivalent to the dimensionality of the basis of the conservation function), and groups the CAs together for which the basis functions are identical. Such CAs should be regarded as equivalent in some natural sense. We can also examine what non-simple core functions emerge after the first one, as  $m$  increases.

Table 2 shows the non-zero half of the conservation function basis vectors for CAs with primary cores at  $m=2$  through 5. (The three CAs with primary cores at  $m=1$



TABLE 2  
 CONSERVATION FUNCTIONS, BASIS VECTORS AND EQUIVALENCE CLASSES FOR CAS WITH FIRST CORES FROM M=2 THROUGH M=5

	m=2	m=3	m=4	m=5
12/c	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
14/e	[1 0]			
15/f	[1 0]	[-1 1 -2 0]	[1 0 -1 0 -1 0 1 0] [2 2 -5 1 -5 1 0 0]	[-20 4 8 0 4 -4 8 0 4 -4 8 0 12 4 -24 0] [11 0 -7 4 -5 4 -7 0 -3 4 -9 0 -11 0 11 0] [-5 4 1 0 -1 4 -3 0 -3 0 -1 4 1 0 -9 0] [-1 4 -3 0 -1 0 -3 4 -3 4 -1 0 -3 0 -5 0]
34/22	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
35/23	[1 0]			
42/2a	[1 0]	[0 0 1 0]	[-1 -1 1 1 1 -1 0 0]	[-3 -3 3 3 0 0 0 0 3 -3 0 0 0 0 0 0] [-2 -2 0 0 2 2 0 0 -1 -3 1 3 0 -6 0 0] [-2 -2 0 0 -1 -1 3 3 -1 -3 4 0 0 -6 0 0]
43/2b	[1 0]			
51/33	[1 0]	[-1 1 -2 0]	same as f	same as f
140/8c	[1 0]		[-1 -1 1 1 0 -2 0 0]	[-5 -5 3 3 1 1 1 1 0 -8 2 2 0 -8 0 0]
142/8e	[1 0]			
136/c8	[0 1]		[0 0 0 0 0 0 -1 1]	[0 0 0 0 0 0 0 -1 -1 -1 3 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 -3 3]
2		[1 0 0 0]		
3		[1 0 0 0]		[-1 -1 1 1 -2 0 0 0 -2 -2 0 0 0 0 0 0]
4		[1 1 -1 0]		[1 1 1 1 -3 -3 1 1 -1 -1 3 -1 0 0 0 0]
10/a		[1 0 1 0]	[1 1 0 2 -1 -1 0 0]	
56/38		[1 1 1 0]		
76/4c		[1 1 1 0]	[1 1 1 1 -2 -2 3 0]	[3 3 3 3 0 0 -12 0 0 0 0 0 0 0 3 0] [0 0 0 0 0 0 0 -3 -3 3 3 3 -3 0 0] [0 0 0 0 3 3 -13 3 2 2 0 0 1 3 -3 0]
138/8a		[1 0 1 1]	[-2 -2 0 -3 1 1 1 1]	[0 0 0 0 0 0 1 -1 -1 -1 -1 -1 1 1 1]
172/ac		[1 0 0 0]		
1			[1 0 0 0 0 0 0 0]	
11/b			[1 1 0 1 0 0 0 0]	
27/1b			[1 1 0 1 0 0 0 0]	
29/1d			[1 1 1 1 1 0 0 0]	[-7 -7 -7 -7 3 3 3 3 3 13 -10 -10 -10 0 0 0]
38/26			[1 1 0 0 0 1 0 0]	
46/2e			[1 1 0 0 0 1 0 0]	
72/48			[0 0 0 0 1 1 -1 0]	
5				[1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0] [2 2 2 1 1 1 1 1 0 -1 0 -1 0 0 0 0] [0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0] [1 1 1 0 0 0 0 0 0 0 1 1 1 0 0 0] [1 1 1 1 -1 0 0 0 -1 -1 0 1 0 0 0 0] [1 1 1 1 1 0 0 1 1 1 0 0 0 1 0 0] [0 0 0 0 1 1 0 0 0 0 -1 0 0 0 0 0]
19/13				
24/18				
36/24				
108/6c				
132/84				

TABLE 3  
EQUIVALENCE CLASSES AND DIMENSIONALITY OF CONSERVATION FUNCTIONS FOR CAS WITH FIRST CORES AT M=2 THROUGH M=6

CA	m=2	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10	m=11
12/c	1		1	1	2	3	5	8	13	21
14/e	1									
15/f	1	1	2	4	8	16	32	64	128	256
34/22	1		1	1	2	3	5	8	13	21
35/23	1									
42/2a	1	1	1	3	5	9	17	31	57	105
43/2b	1									
51/33	1	1	2	4	8	16	32	64	128	256
140/8c	1		1	1	2	3	5	8	13	20
142/8e	1									
136/c8	1	1	1	2	4	7	12	21	37	65
2		1			1	1	1	2	3	4
3		1		1	2	3	5	9	16	28
4		1		1	1	2	3	5	8	13
10/a		1	1		1	3	4	5	9	16
56/38		1								
76/4c		1	1	3	5	8	14	25	45	82
138/8a		1	1	1	2	4	7	12	21	37
172/ac		1			1	1	1	2	3	3
1			1			1	2	3	4	6
11/b			1							
27/1b			1			2	3	5	9	16
29/1d			1	1		2	4	7	13	24
38/26			1			1	2	2	4	8
46/2e			1			1	1	1	2	3
72/48			1			1	1	1	2	3
5				2	2		2	7	8	21
19/13				1			1	2	3	4
24/18				1			1	1	1	2
36/24				1			1	1	1	2
108/6c				1	1	3	5	7	10	18
132/84				1		2		4	1	8
23/17					1		2		4	1
50/32					1		2		4	1
77/4d					2		4		8	2
178/b2					1		2		4	1
248/e8					2		4		8	2

TABLE 4  
EQUIVALENCE CLASSES AND DIMENSIONALITY OF CONSERVATION FUNCTIONS FOR CAS WITH FIRST CORE AT M=7 THROUGH M=16

CA	m=2	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10	m=11	m=12	m=13	m=14	m=15
No CAs with primary cores at m=7														
44/2c							1		1					
73/49							1							
7								1		2				
No CAs with primary cores at m=10														
No CAs with primary cores at m=11														
33/21											1			
164/a4												1		
94/5e													*	
104/68													*	
No CAs with primary cores at m=15														
No CAs with primary cores at m=16														

\*solution known to exist, but not yet calculated explicitly.