Formally published (with corrections) as *Cellular Automata Complexity Trade-Offs*, *Simple Computation-Universal Cellular Spaces*, *Simple Nontrivial Self-Reproducing Machines*, and *Real-Time Language Recognition by One-Dimensional Cellular Automat*

FOCS = *IEEE Foundations of Computer Science Conference Record*

FOCS 9 (1968) 269-277

SIMPLE COMPUTATION-UNIVERSAL CELLULAR SPACES AND SELF-REPRODUCTION

Alvy Ray Smith III
Electrical Engineering Department
Stanford University
Stanford, California

Summary

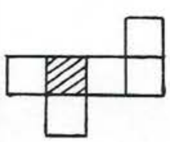
Cellular spaces computationally equivalent to any given Turing machine are exhibited which are simple in the sense that each cell has only a small number of states and a small neighborhood. Neighborhood reduction theorems are derived in this interest, and several simple computation-universal cellular spaces are presented. Conditions for computation-universality of a cellular space are investigated, and, in particular, the conjecture that unbounded but boundable propagation in a space is a sufficient condition is refuted. Finally, the computation-universal spaces derived in the study are used to introduce, via recursive function theory, examples of simple self-reproducing universal Turing machine configurations in one and two dimensions.

Introduction

Von Neumann was the first to use automata theory to study the logical intricacies of biological reproduction. In particular, he[1] detailed a cellular space conceived as an infinite chessboard of identical finite automata--one automaton (cell) per square--which supported an activity interpreted as self-reproduction. In the process he also demonstrated the computational power of his cellular space.

Both the cellular space (29 states per cell) and the self-reproducing, computing construction (40,000 cells or so) of von Neumann are quite complex and have led others (Thatcher[2], Codd[3], and Arbib[4]) to simplify and generalize. This paper is also along these lines; the terminology, reviewed below, is essentially that of Thatcher and Codd.

Although more general definitions are possible, here we will consider only cellular spaces of the infinite chessboard variety--a one-dimensional cellular space corresponding to one row of such a chessboard. Associated with each cell C of a cellular space Z is a <u>neighborhood</u> consisting of C itself and a finite set of cells in fixed positions relative to C. In this paper, all cells in a given cellular space will have neighborhoods of the same shape designated by a subset of chessboard squares called a <u>template</u>, such as



where we hatch the cell whose neighborhood this is. Thus the neighborhood of cell C is determined by translating the template associated with Z until the hatched <u>template origin</u> covers cell C. All cells under the template squares then form the neighborhood of C.

The state of each cell in a given cellular space at time t+1 is uniquely determined by the <u>transition function</u> f of the cell acting on the <u>neighborhood state</u> of the cell at time t. All cells operate synchronously under action of the <u>global transition function</u> F which maps any one <u>configuration</u>--i.e., an allowable assignment of states to all cells--into another. Thus $F(c)(C) = f(N(c,C))$ where C is a cell with neighborhood state $N(c,C)$ in configuration c. Given an initial configuration c_0, F determines a sequence of configurations, the <u>propagation</u> $\langle c_0 \rangle$:

$$c_0, c_1, \ldots, c_t, \ldots$$

where $c_{t+1} = F(c_t)$ for all t.

There is a distinguished state q_0, the <u>quiescent state</u>, in each cell C such that $f(N(c,C)) = q_0$ if q_0 is the state of every cell in the neighborhood of C. A configuration c is restricted to have finite <u>support</u>; i.e., sup(c), the set of nonquiescent cells, is finite. Usually, the term "configuration c" will be used loosely to mean $c|\sup(c)$. A configuration c is <u>passive</u> if $F(c) = c$. A configuration c' is a <u>subconfiguration</u> of c if $c|\sup(c') = c'|\sup(c')$.

For configuration c, the propagation $\langle c \rangle$ is <u>bounded</u> if for all t, $C \in \sup(c)$, and $C' \in \sup(F^t(c))$, it holds that $\max_{C,C'} \rho(C,C') < K$ for Manhattan-city-block metric ρ and integer K. Otherwise, $\langle c \rangle$ is <u>unbounded</u>. $\langle c \rangle$ is <u>boundable</u> if there exists a disjoint configuration d such that $\langle c \cup d \rangle$ is bounded, and <u>unboundable</u> if no such d exists. By <u>disjoint</u> configurations c and d we mean their supports are disjoint. By the notation $c \cup d$ we mean the <u>union</u> of c and d, defined by

$$(c \cup d)(C) = \begin{cases} c(C) & \text{if } C \in \sup(c) \\ d(C) & \text{if } C \in \sup(d) \\ q_0 & \text{else} \end{cases}$$

if c and d are disjoint.

If c and d are disjoint, then c <u>passes information to</u> d if there is a time t such that $F^t(c \cup d)|S \neq F^t(d)|S$ where $S = \sup(F^t(d))$. This definition will be adequate here but it should be noted that it must be modified slightly to handle some of the cellular space phenomena recorded in the literature (notably the "pushing" of one configuration by another in Arbib[4]).

In the sequel, we will be representing symbols of a Turing machine tape by states of cells in a cellular space. Here it will suffice to have only one cell represent each square, but in general a group of cells will represent a single tape square. Hence <u>T</u>, the set of Turing machine tapes on alphabet X corresponds to an effectively defined subset of the set of configurations, the <u>tape configurations</u>, over cellular-space state set $W \subset Q$ where Q is the state set of the space. Let $C_{\underline{T}}$ be the set of tape configurations for a cellular space. Then $\underline{T}$ is obtained from $C_{\underline{T}}$ by effective procedure h.

Let $g:\underline{T} \rightarrow \underline{T}$ be a function mapping tapes into tapes. Then we say c computes g if there exists a configuration c such that, for any tape configuration $d \in C_{\underline{T}}$, $g(h(d))$ is defined iff there exists a

time t such that $h(F^t(c \cup d)|\sup(C_{\underline{T}})) = g(h(d))$

where $\sup(C_{\underline{T}}) = \bigcup_{d \in C_{\underline{T}}} \sup(d)$ and $F^t(c \cup d)|\overline{\sup(C_{\underline{T}})}$ does

not pass information to $F^t(c \cup d)|\sup(C_{\underline{T}})$, and there exists an effective procedure for determining when a computation is completed. We leave this procedure unspecified in the definition since it may differ with the context (e.g., the "computation complete" signal might be a prespecified cell in a prespecified state or a prespecified pattern of states in a prespecified region of the space). Note that this definition does not require that a configuration which computes become passive.

A cellular space Z is computation universal if for any Turing machine computable function g there exists a configuration c in Z such that c computes g.

A Turing machine will be the type that moves right (R) or left (L) at each time step. Its associated state table is assumed to be of the following form (only one typical entry shown):

state

symbol	q_0	q_1	. . .	q_{n-1}
x_0				
x_1		$x_i@/q_j$		
.				
.				
.				
x_{m-1}				

$@ \in \{R,L\}$
$0 \leq i \leq m-1$
$0 \leq j \leq n-1$

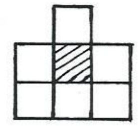
This will be referred to as an (m,n) Turing machine.

The cellular spaces of von Neumann, Thatcher, and Codd as well as those introduced in this paper are what Holland[5] terms Moore-type spaces--i.e., there is a non-zero delay associated with every transition. Mealy-type spaces (zero delay for some states) are used by Wagner[6] and Arbib. Wagner does not treat self-reproduction, but he does study computation by embedding multi-headed machines called "spider automata" in Mealy-type spaces ("modular computers") with cells of considerable complexity. Since a one-legged spider is a Turing machine, his work is related to that below although his aim is not specifically that of cell simplicity.

Simple Computation-Universal Cellular Spaces

Theorem 1. For an arbitrary (m,n) Turing machine, there exists a 2-dimensional, 7-neighbor, max(m+1, n+1)-state cellular space which simulates it in real time.

Proof. Let T be an arbitrary (m,n) Turing machine. A cellular space Z_T with neighborhood template

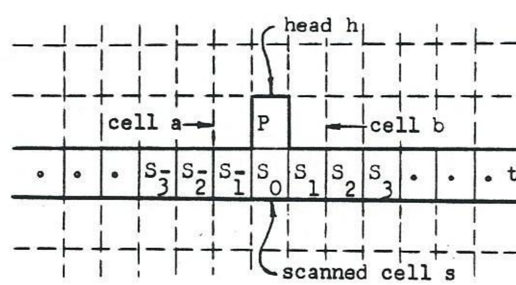


is constructed which simulates T as follows. Each cell of Z_T is provided with a set Q of M = max(m+1,n+1) states. Without loss of generality, let $Q = \{0,1,2,\ldots,M-1\}$ so that (i+1) corresponds to symbol x_i of T for $0 \leq i \leq (m-1)$ and state (j+1) corresponds to Turing machine state q_j for $0 \leq j \leq (n-1)$. 0 is the quiescent state of Z_T and never corresponds to a Turing machine state or symbol. The geometry of Z_T will be utilized to distinguish

a cell whose state $Q_1 \in A = \{1,\ldots,m\}$ corresponds to

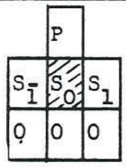
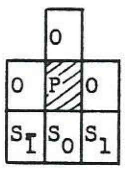
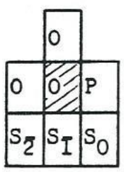
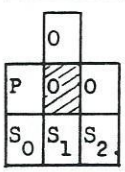
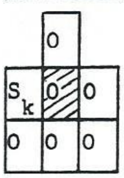
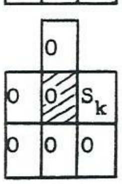
a Turing machine symbol from a cell whose state $Q_2 \in B = \{1,\ldots,n\}$ corresponds to a Turing machine state.

We cause Z_T to simulate T by embedding a configuration in it which "looks like" T. That is, one row of cells in Z_T is the "tape" of the embedded Turing machine--one cell of Z_T per tape square of T--and one cell in an adjacent row is the "head". Thus the embedded Turing machine configuration will have the following form at any one instant of time:



As indicated in the diagram, a and b are always labels for the cells to the left and right, respectively, of the head cell h. All other symbols are state assignments: $S_0 \in A$ is always the state of the scanned cell s; $S_k \in A$ is always the state of the tape cell at distance |k| from the scanned cell in the direction determined by the sign of k as indicated; and $P \in B$ is the state of head cell h. C_y is used to designate the cell immediately to the $y \in \{R,L\}$ of a finite embedded tape. All cells other than the head and tape cells are assumed to be in the quiescent state 0. Thus C_R and C_L are always in state 0.

Head cell h is made to "move" along the tape subconfiguration simulating the head moves of T by suitable specification of the transition function f for a cell in Z_T. This is simply done. Unless the cell is a, b, h, s, C_R, or C_L, then it does not change state. For these six cases, let the Turing machine state-table entry for symbol x_r and state q_t be denoted (x_r,q_t). Then f is given for cells a, b, h, s, C_R, and C_L as indicated below when $(x_r,q_t) = x_p@/q_q$.

cell C	neighborhood state of C	next state of C	conditions
s		p	$S_0 = r+1$, $p \in A$; $P = t+1$
h		0	in all cases
a		0	if @ = R
		(q+1)	if @ = L, $(q+1) \in B$
b		(q+1)	if @ = R
		0	if @ = L
C_R		1	$1 \in A$ is the "blank" symbol
C_L		1	as for C_R

The last two entries are "tape extenders" which convert the quiescent state to the blank symbol at either end of the necessarily finite embedded tape configuration. This is necessary since the simulated Turing machine may need an infinite tape.

Some one state Q* is designated the starting state of T. Corresponding to Q* is a state $w \in B$ in each cell of Z_T. Thus the simulation of T in Z_T is set up as follows: the non-blank portion of the (finite) initial tape of T is embedded in one row of Z_T. The cell above the cell corresponding to the leftmost non-blank square is set to state w and hence represents the initial position of the tape head of T. Then the global transition function F causes the cellular tape subconfiguration to be modified (in a real-time simulation) just as would be the tape of T. ▮

Remark. The 7-neighbor template above suggests the symmetrically more elegant hexagonal tiling of the infinite plane. Indeed such a "beehive" cellular space can be used[7] to construct another proof of Theorem 1.

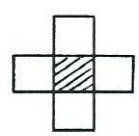
It is of interest to contrast the construction in the proof above, in which the cell design depends on the Turing machine to be embedded, to the cellular spaces of von Neumann, Thatcher, Codd, and Arbib, in which any Turing machine can be simulated once the cell design is set. However the difference between the Turing-machine-dependent cell constructions and the Turing-machine-independent cell constructions is not too important in many cases since the embedded Turing machine of interest is often the universal machine, as, for example, in the corollaries below. In this case the machine-dependent cells are clearly superior in the sense that all simulations are real-time (or "almost" real-time as will be specified in other theorems below), as opposed to the very slow simulations in, say, the von Neumann space. Of course, a real-time simulation of a universal Turing machine is not real-time with respect to the original Turing machine simulated by the universal machine.

Corollary 1.1 There exists a max(m+1,n+1)-state computation-universal cellular space for every (m,n) universal Turing machine.

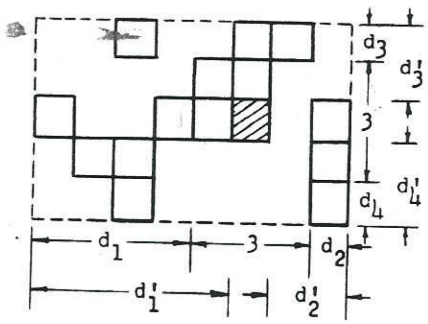
Corollary 1.2 There exists a 7-state computation-universal cellular space in two dimensions.

Proof. Minsky[8] has found a (6,6) universal Turing machine. ▮

Previous work with infinite chessboard varieties of cellular spaces has often used the symmetric, nearest-neighbor, 5-cell neighborhood of von Neumann:



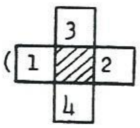
The 7-neighbor spaces of the type used in proving Theorem 1 can be replaced easily with spaces having the von Neumann neighborhood. In fact, it will be shown that a chessboard space with any neighborhood template can be simulated by another space with at most five neighbors per cell (or, at most three in one dimension). Any neighborhood template has a minimal circumscribing rectangle with nomenclature as indicated in this example:



(Here $d_1 = 4$ and $d_2 = d_3 = d_4 = 1$.)

In the theorem below, the notation (K,M) will be used to represent the M-state cellular space with template diagram K from the set of template diagrams with hatched template origins.

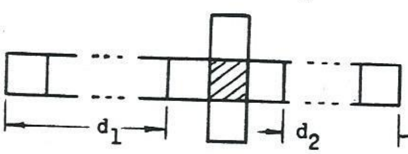
Theorem 2. Any (K,M) cellular space Z can be simulated by a

( $, [M(M+1)^{d_3'+d_4'}][M(M+1)^{d_3'+d_4'+1}]^{d_1+d_2})$

cellular space Z', where d_i or d_i' is set to zero and cell i is not included in the template if $d_i' = 0$, $i = 1,2,3,4$. The simulation requires $\max(d_1',d_2')+\max(d_3',d_4')$ times real time.

Proof. Let Z have transition function f and state set Q. The proof is in two steps, one for reduction in each dimension:

Step 1. Reduce Z to Z_1 =

( $, M(M+1)^{d_3'+d_4'})$

by supplying each cell in Z_1 with states of the set $Q_1' = \underbrace{Q_1 x Q_1 x \ldots x Q_1}_{d_3'} x Q x \underbrace{Q_1 x Q_1 x \ldots x Q_1}_{d_4'}$ where

$Q_1 = Q+\{b\}$ and b is a specially designated state. For each cell in Z in state $q \in Q$, there is a corresponding cell in Z_1 set to state $q_{11}' \in Q_1'$ where $q_{11}' = (b,\ldots,b,q,b,\ldots,b)$. Z_1 is supplied with a transition function f_1 which "fills in the blanks" b of q_{11}' as follows:

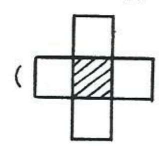
Number the positions of the $(d_3'+d_4'+1)$-tuple q_{11}' from 0 to $(d_3'+d_4')$, from left to right (then q is in position d_3'). Let cell C be in state q_{11}'.

Then f_1 changes q_{11}' to $q_{12}' = (b,\ldots,b,\overline{d}_3',q,\underline{d}_3',b,\ldots,b)$ where $\overline{d}_3'$ is the value of position d_3' in the cell above C and $\underline{d}_3'$ is the value of position d_3' in the cell below C. Similarly, f_1 changes q_{12}' to $q_{13}' = (b,\ldots,b,\overline{d_3'-1},\overline{d}_3',q,\underline{d}_3',\underline{d_3'+1},b,\ldots,b)$, and so forth until q_{1y}' contains no symbols b, where $y = \max(d_3',d_4')$. Note that this process requires $\max(d_3',d_4')$ steps.

Then q_{1y}' contains the value of position d_3' of every cell up to and including the cell at d_3' cells above C and of every cell up to and including the cell at d_4' cells below C. That is, the neighborhood state of a cell in Z_1 in state q_{1y}' contains at least the information contained in a neighborhood state of a cell in state q in Z. f_1 selects this information, acts on it as would f, and resets the b's all in one step.

Thus Z_1 simulates Z with the given number of states. Clearly, if $d_3' = 0$, then Z_1 need only have states of the form $Q_1' = Qx\underbrace{Q_1 x Q_1 x \ldots x Q_1}_{d_4'}$ and Z_1 simulates Z with $M(M+1)^{d_4'}$ states. A similar result holds if $d_4' = 0$.

Step 2. Reduce Z_1 to Z_2 =

( $, N(N+1)^{d_1+d_2})$

where $N = M(M+1)^{d_3'+d_4'}$. Then identify $Z' = Z_2$. This can be accomplished in a manner exactly analogous to that used for reducing Z to Z_1 in Step 1. Each state in Z_2 is of the set

$$Q_2' = \underbrace{Q_2 x \ldots x Q_2}_{d_1} x Q_1' x \underbrace{Q_2 x \ldots x Q_2}_{d_2}$$

where $Q_2 = Q_1'+\{B\}$, with $B \neq b$ a specially designated state. For each cell in Z_1 in state $q_{1y}' \in Q_1'$ there is a cell in Z_2 in state $q_{21}' = (B,\ldots,B,q_{1y}',B,\ldots,B) \in Q_2'$ and f_2 "fills in the blanks" B, just as did f_1 for Z_1, to obtain q_{2z}' where $z = \max(d_1,d_2)$. The Z_2 neighborhood thus contains the information in a Z_1 neighborhood, and f_2 is designed to act on this information to simulate Z_1 and hence Z. The blank-filling process requires z steps and one step is needed to reset the B's.

Clearly, Step 2 can follow Step 1 immediately if the final reset operation of Step 1 is omitted. Then filling all blanks b and B requires $\max(d_1,d_2)+\max(d_3',d_4')$ steps and resetting them requires one step. The simulation of one step of f occurs during the reset step. Hence Z is simulated with a time slowdown of $1+\max(d_1,d_2)+\max(d_3',d_4') = \max(d_1',d_2')+\max(d_3',d_4')$. ▮

Theorem 2 applied to the 7-state space of Corollary 1.2 yields a 5-neighbor space with $7(8)^2$ states (or, with only slight revision of the theorem in this special case, 7(8) states). In this space, and, in fact, in all spaces of the type introduced in the proof of Theorem 1, we can do much better in the sense of fewer states as the following theorem purports.

<u>Theorem 3.</u> For an arbitrary (m,n) Turing machine, there exists a 2-dimensional, 5-neighbor, M-state cellular space which simulates it (in four times real time), where $M = \max(3m+1,n+1)$.

<u>Proof.</u> Each symbol x_i of an arbitrary Turing machine T will be represented in 5-neighbor cellular space Z by a set of states $\{(s_i,0),(s_i,1),(s_i,-1)\}$.

The configuration which simulates T occupies two rows as in the proof of Theorem 1, but here the information stored in that larger neighborhood about right (R) or left (L) head moves is coded into the enlarged state set $\{(s_i,b)\}$. Roughly, $b = 1$ corresponds to R and $b = -1$ to L.

Suppose T is in state q_i reading symbol x_j at time t; and changes x_j to x_j', moves R (or L), and goes into state q_i' at t+1. Then Z will have a state $(s_i,0)$ corresponding to q_i and state $(s_j,0)$ corresponding to x_j. Thus for some time t' and quiescent state 0 there will be a configuration in Z of the form (parentheses denote individual cells):

$$t': \quad \ldots(\ 0\ )\ (\ 0\ )\ (s_i,0)\ (\ 0\ )\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j,0)\ (s_k,0)\ (s_r,0)\ldots$$

The top row is the "head" row and the bottom row is the "tape" row. The simulation of the R move proceeds as follows (the L move proceeds analogously):

$$t'+1: \ldots(\ 0\ )\ (\ 0\ )\ (s_i',0)\ (\ 0\ )\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j',1)\ (s_k,0)\ (s_r,0)\ldots$$

$$t'+2: \ldots(\ 0\ )\ (\ 0\ )\ (s_i',0)\ (\ 0\ )\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j',1)\ (s_k,1)\ (s_r,0)\ldots$$

$$t'+3: \ldots(\ 0\ )\ (\ 0\ )\ (s_i',0)\ (s_i',0)\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j',1)\ (s_k,1)\ (s_r,0)\ldots$$

$$t'+4: \ldots(\ 0\ )\ (\ 0\ )\ (\ 0\ )\ (s_i',0)\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j',1)\ (s_k,0)\ (s_r,0)\ldots$$

$$t'+5: \ldots(\ 0\ )\ (\ 0\ )\ (\ 0\ )\ (s_i'',0)\ (\ 0\ )\ldots$$
$$\ldots(s_u,0)\ (s_v,0)\ (s_j',0)\ (s_k',b)\ (s_r,0)\ldots$$

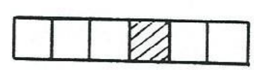
Thus at t'+5 the space is already in the first step of its four-step simulation of the next step of T. It is a simple matter to specify a transition function which accomplishes the simulation outlined above. ▮

A consequence of Theorem 3 is a 5-neighbor, 13-state computation-universal cellular space obtained by applying the theorem to the (4,7) universal Turing machine exhibited by Minsky[8]. This might be compared for simplicity to the 5-neighbor, 8-state space of Codd. Such a comparison is difficult since the 13-state computation-universal configuration occupies two "rows" of the space whereas the 8-state configuration covers very many more and computes very slowly--i.e., in much more than four times real time.

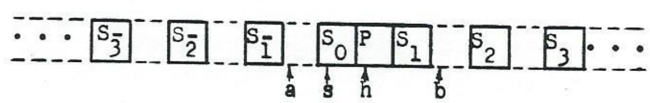
Utilization of only two rows of a 2-dimensional space implies immediately the existence of a 169-state, 1-dimensional computation-universal cellular space, but Theorem 4 does better:

<u>Theorem 4.</u> For an arbitrary (m,n) Turing machine, there exists a 1-dimensional, 6-neighbor, max(m+1, n+1)-state cellular space which simulates it in real time.

<u>Proof.</u> Let T be an arbitrary (m,n) Turing machine. Then a cellular space Z_T is designed to simulate T by providing it with the following 6-cell neighborhood template:



The embedded Turing machine configuration is as illustrated below; the tape squares occupy every other cell in the space.



The transition function f leaves the state of a cell C unchanged except in the six cases a, s, h, b, C_R, and C_L (as in Theorem 1). The function f is easily specified and is omitted here. It should be noted that tape extenders are required here as in Theorem 1 to convert quiescent cells to blank cells at the tape "ends". ▮

Corollary 4.1 There exists a 1-dimensional, max(m+1,n+1)-state computation-universal cellular space for every (m,n) universal Turing machine T.

Corollary 4.2 There exists a 1-dimensional, 7-state computation-universal cellular space.

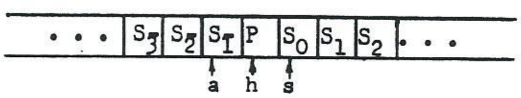
Proof. Let T be Minsky's (6,6) universal Turing machine. ▮

Similar to Theorem 3 for two dimensions is Theorem 5 for one dimension.

Theorem 5. For an arbitrary (m,n) Turing machine, there exists a 1-dimensional, (m+2n)-state, 3-neighbor cellular space which simulates it (in at most twice real time).

Proof. Let T be an arbitrary (m,n) Turing machine. Then a cellular space Z_T which simulates T is designed as follows: Provide Z_T with this template:

and embed a Turing machine configuration in Z_T as indicated below:



The transition function f leaves the state of all cells unchanged except in the cases a, h, and s. It is a simple matter to fill in the details of this function such that tape configurations like $\cdots x_0 x_1 q x_2 x_3 \cdots$ simulating a right move into new state q' after changing symbol x_2 to x_2' appear in time as

$$\cdots x_0 x_1 q\ x_2 x_3 \cdots$$

$$\cdots x_0 x_1 x_2' q' x_3 \cdots \quad .$$

Similarly, a left move looks like this:

$$\cdots x_0 x_1 q\ x_2 x_3 \cdots$$

$$\cdots x_0 x_1 q_L' x_2' x_3 \cdots$$

$$\cdots x_0 q' x_1 x_2' x_3 \cdots \quad .$$

Thus two states, q and q_L, are needed to represent each Turing machine state. ▮

Conditions for Computation-Universality

Theorems 1-5 have provided several designs for simple computation-universal cellular spaces. The obvious question then is, what is the simplest such space? That is, what are necessary and sufficient conditions for computation-universality of a space? This question is of special interest in research on the origin-of-life problem, an area in which cellular automata approaches hold some promise[9,10].

In particular, Codd has suggested that a necessary condition for computation-universality in a cellular space is the existence of unbounded but boundable propagation in the space. This condition is proved to be not sufficient below, however, after the presentation of two necessary conditions. First we extend the definition of boundedness.

Definition. A propagation ⟨c⟩ is k-bounded effectively if ⟨c⟩ is bounded by some integer K (as in the definition of "bounded") and K can be effectively determined.

Definition. The k-bounding problem of a configuration c is to determine, in an effective manner, (1) an integer K, or an algorithm for finding K, such that ⟨c⟩ is k-bounded effectively, or (2) that ⟨c⟩ is unbounded. The bounding problem for a cellular space Z is solvable if, for all configurations in Z, the k-bounding problem of the configuration is solvable. The bounding problem for Z is unsolvable if there exists a configuration in Z with an unsolvable k-bounding problem.

Definition. The propagation problem of a configuration c is to determine, in an effective manner, whether there exists a time t at which the propagation ⟨c⟩ becomes passive--i.e., if there exists t such that $F^t(c) = F^{t-1}(c)$. The propagation problem for a cellular space Z is solvable if, for all configurations in Z, the propagation problem of the configuration is solvable. The propagation problem for Z is unsolvable if there exists a configuration in Z with an unsolvable propagation problem.

Theorem 6. If the bounding problem of a cellular space Z is solvable, then the propagation problem of Z is solvable.

Proof. If a configuration c in Z has unbounded propagation, then ⟨c⟩ never becomes passive. If a configuration d has ⟨d⟩ k-bounded effectively by K, and if F is the global transition function of Z, then let R be a region in Z to which ⟨d⟩ is confined for all time. Such an R can be effectively determined since d and K are known. Suppose R contains N cells and Z is an A-state cellular space. Then R has A^N possible states, and an effective procedure for solving the propagation problem of d is as follows:

Observe R containing d at time 0 for at most A^N+1 time steps. If at any time t, $0 \le t \le A^N$, $F^t(d) = F^{t+1}(d)$, then ⟨d⟩ becomes passive. If this is not the case, then there must exist times

t, t', with $t' > t+1$, such that $F^t(d) = F^{t'}(d)$ and $F^t(d) \neq F^{t+1}(d) \neq \ldots \neq F^{t'-1}(d)$. That is, since A^N is finite then $\langle d \rangle$ must become cyclic and hence active for all time. ▮

Theorem 7. A computation-universal cellular space has an unsolvable propagation problem.

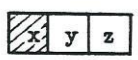
Proof. Let Z_U be a computation-universal cellular space. Then there must exist a configuration d in Z_U which computes f_H, the function computed by H, a Turing machine with an unsolvable halting problem. Hence it is impossible to determine whether there is a passive configuration in $\langle d \rangle$ for some time t. ▮

Apply Theorem 6 to Theorem 7 to obtain

Corollary 7.1 A computation-universal cellular space has an unsolvable bounding problem.

Theorem 8. The existence of unbounded but boundable propagation in a cellular space is not sufficient for computation-universality of the space.

Proof. Consider the 2-state, 3-neighbor cellular space Z which has the template



and the transition function f which leaves the state of a cell unchanged except in these cases: $f(100) = 0$, $f(110) = 0$, and $f(010) = 1$, where $f(xyz) = x'$ gives the next state x' of the cells with states as shown above. Define configurations c and d on integer coordinates by

$$c(x) = \begin{cases} 1, & x = 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{and } d(x) = \begin{cases} 1, & x = +1 \\ 0, & \text{otherwise} \end{cases}.$$

Then $\langle c \rangle$ is unbounded but $\langle c \cup d \rangle$ is not. However, we now show that the bounding problem is solvable.

Let e be an initial configuration in the space Z. Let K be the number of cells between the leftmost 1 and the rightmost 1 in e, inclusive. Look at the leftmost block of one or more 1's in e and let q be the state of the cell two cells left (thus $q = 0$ at $t = 0$). Then q_i is the state of the cell i cells to the right of the cell whose state is q (thus $q_1 = 0$ at $t = 0$).

Lemma: If $q = 1$ at time $t' > 0$ where $q = 0$ for $0 \leq t < t'$, then $\langle e \rangle$ is unbounded.

Proof: There are two possibilities at time t': $qq_1 = 10$ or $qq_1 = 11$. If $qq_1 = 10$ at t', then $qq_1q_2q_3 = 0100$ at $t'-1$. The subconfiguration 100... (all 0's to the left and don't cares to the right) has unbounded propagation hence $\langle e \rangle$ is unbounded. If $qq_1 = 11$ at t', then $qq_1q_2q_3 = 0101$ at $t'-1$ and $qq_1q_2q_3q_4q_5 = 011101$ at $t'-2$. Thus, in general, if at t', $qq_1 = 11$ then $qq_1q_2q_3q_4 \cdots q_{n-1}q_n q_{n+1}q_{n+2} = 01111\ldots1101$ at $t'-((n+1)/2)$, n odd, as can be readily checked. (The other possible candidate at each step is 00101...100- which does not qualify since $f(00-) \neq 1$.) Thus $qq_1 = 11$ at t' implies $q_1 = 1$ for all preceding times which contradicts the assumption that $q_1 = 0$ at $t = 0$. Hence $qq_1 = 11$ cannot occur at t', and it is sufficient to check only q to determine if $\langle e \rangle$ is unbounded or not.

Since $f(100) = 0$, the rightmost 1 of e is erased ($1 \to 0$) at the first step. In fact, at least one such erasure occurs at each succeeding step. Hence, if $q = 0$ for K+2 time steps from $t = 0$, then the initial configuration e has been completely erased and $\langle e \rangle$ is k-bounded effectively by K+1. Else, by the lemma, $\langle e \rangle$ is unbounded. ▮

Self-Reproducing Machines

In the preceding sections only the computational abilities of cellular spaces have been investigated. It is of interest to study also the "constructional" abilities--the construction of one configuration by another. It is not at all clear what "construction" should be defined to be, but it is probably agreed that (non-trivial) self-reproduction should be considered an example of construction. Moore's definition[11] of "self-reproduction" will be used, although, as he points out, it does not exclude trivial self-reproduction. That is, care will be exercised to insure the non-triviality of any self-reproducing configuration in this section.

In fact, a set of one-dimensional self-reproducing Turing machine configurations will be exhibited in this section. Let c be one of these configurations. Then c self-reproduces in the (Moore) sense that if c is embedded in cellular space Z at time $T = 0$, then at some time $T' > 0$ there will exist two disjoint copies of c in Z; at $T'' > T'$ there will exist three copies, etc. Furthermore c will be required to compute any given ~~total~~ recursive function g.

The environment of one self-reproducing configuration in the set will be one of the type of spaces introduced above. Let Z_M be such a space; that is, the transition function and neighborhood of each cell in Z_M is such that M, an (m,n) Turing machine, can be simply embedded as in, say, the proofs of Theorem 1 and Theorem 4. It is said then that M is wired in Z_M or that Z_M has M wired in.

By the notation $x \xrightarrow[P]{} y$ will be meant that Turing machine program P acting on initial tape x halts with y on the tape as its final result.

Note that the following two extremes follow from work relating Turing machines to computing spaces: (1) the wired-in Turing machine "does everything", or (2) the self-reproducing configuration "does everything".

Self-Reproduction: Extreme 1. The wired-in Turing machine M does everything.

Example 1: Trivial Self-Reproduction. Wire in machine M which duplicates its input tape, repositions its head, duplicates all tape to the right of this position, etc. This can be represented as follows:

$$\underset{\uparrow}{x} \xrightarrow[M]{} (\underset{\uparrow}{x},x) \xrightarrow[M]{} (x,x,\underset{\uparrow}{x}) \xrightarrow[M]{} \cdots$$

where the vertical arrow indicates the position of the head after the computation indicated by the horizontal arrow (i.e., $\underset{\uparrow}{x}$ means the head is at the leftmost symbol in the tape string x).

Note that any one-dimensional configuration can be made to self-reproduce in Moore's sense in this scheme. Just embed a "head" cell at the left end of a given "tape" configuration; program the head cell to its initial state.

Example 2. Non-trivial Case. Wire in universal Turing machine U' such that

$$\underset{\uparrow}{f'}(P,x) \xrightarrow[U']{} (f'(P,x),y,\underset{\uparrow}{f'}(P,x)) \xrightarrow[U']{} \cdots$$

where f' is the encoding of programs and tapes required by U' and $x \xrightarrow[P]{} y$. Note that P could be universal if desired.

By the notation h.s is meant that "tape" configuration s has been augmented by a "head" cell in state h in initial position (i.e., in a position corresponding to the leftmost square of tape string s). Hence Examples 1 and 2 above can be summarized as theorems.

Theorem 9. Let S be the set of finite 1-dimensional configurations on finite state set A. Then there exists cellular space Z_M with (head) cell state h in which h.s is a self-reproducing configuration for all s∈S. (That is, h.s is the initial configuration and subconfiguration s self-reproduces.)

Theorem 10. For any Turing machine P, there exists a cellular space $Z_{U'}$, (head) cell state h, and configuration c such that h.c self-reproduces and simulates P.

Corollary 10.1 There exist $Z_{U'}$, h, and c such that h.c self-reproduces and is universal.

Self-Reproduction: Extreme 2. The self-reproducing configuration does everything. That is, for simplicity of the embedding space, a solution is desired so that the wired-in computer U does as little as possible--i.e., nothing more is required of U than it be universal (as opposed to U' in Theorem 10 above, of which several other duties are also required). In fact, since the simplest U does just the following:

$$f(P',x') \xrightarrow[U]{} y'$$

where f is the encoding function required by U and $x' \xrightarrow[P']{} y'$, then a program P is desired such that

$$\underset{\uparrow}{x} \xrightarrow[P]{} (f(P,x),y,\underset{\uparrow}{f}(P,x)) \xrightarrow[P]{} \cdots$$

where y is the result of some computation on x (e.g., y might be a string of d background (blank) symbols, b^d, so that f(P,x) is "separated" from the second f(P,x).

The following theorem should be compared to a similar result on self-describing machines obtained by Lee[12]. In it we will use terminology due to Michael Arbib: A bi-recursive function f from set R to set S is such that given R we can effectively find f(R)∈S and given s∈S we can effectively tell whether or not it is of the form f(R), and if so for which (unique) R.

Theorem 11. For an arbitrary bi-recursive function f from (program, tape) pairs to tapes and for an arbitrary total recursive function g ~~such that~~ g(x) = y, there exists a self-describing machine with program P such that

$$\underset{\uparrow}{x} \xrightarrow[P]{} (f(P,x),y,\underset{\uparrow}{f}(P,x))$$

Proof. Define the function h from programs to programs such that h(Q) is the first program which reads arbitrary input tape x, encodes program Q and tape x by given function f to get f(Q,x), prints f(Q,x), computes g(x) to get ~~y, prints y~~, prints f(Q,x) again (either by re-encoding f(Q,x) or by copying the result of the first encoding), and finally moves the head to the leftmost symbol in the rightmost encoding f(Q,x). That is,

$$\underset{\uparrow}{x} \xrightarrow[h(Q)]{} (f(Q,x),y,\underset{\uparrow}{f}(Q,x))$$

Clearly h can be chosen total recursive. Thus, by the recursion theorem, there exists P which is a computational fixed point of h such that

$$\underset{\uparrow}{x} \xrightarrow[P]{} (f(P,x),y,\underset{\uparrow}{f}(P,x))$$

Remark 1. In particular, g(x) could be a universal Turing machine function designed to be total by defining it to leave an empty tape in case x is not an encoded Turing machine and tape.

Remark 2. Clearly, only slight modifications of the proof above are required so that P satisfies

$$\underset{\uparrow}{x} \xrightarrow[P]{} (f(P,x),y,\underset{\uparrow}{f}(P,x)) \xrightarrow[P]{} \cdots$$

That is, h(Q)--hence P--is chosen so that it ignores all tape to the left of the head position when the head is in its initial state, and the final state of the h(Q) in the proof is identified with the initial state.

Thus in cellular space Z_U with U wired in, the following situation can hold:

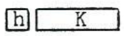
$$f(P,x) \xrightarrow[U]{} (f(P,x),y,f(P,x))$$

at some time $T > 0$ and

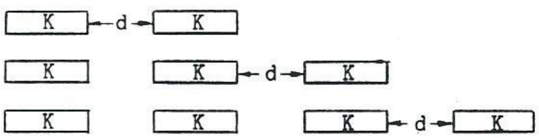
$$f(P,x) \xrightarrow[U]{} (f(P,x),y,f(P,x),y,f(P,x))$$

at some later time $T_1 > T$ and so forth. That is, if the "tape" configuration K for f(P,x) is embedded in Z_U and a "head" cell is programmed in the proper place into the configuration to form the configuration h.K, where h is the head cell state corresponding to the initial state of U, then later (assuming P is a halting program) the tape configuration for (f(P,x),y,f(P,x)) will exist in Z_U, etc.

Suppose $g(x) = b^d$ = string of d background symbols for all x. Then, in pictures,

h.K:  $T = 0$

provides the following time sequence:



$T_1 > T$

$T_2 > T_1$

$T_3 > T_2$

Hence we have shown the following:

Theorem 12. Let Z_U be a computation-universal cellular space with universal Turing machine U wired in. Then there exists a configuration c in Z_U which is self-reproducing and computes any given ~~total~~ recursive function g.

Remark. Theorem 12 gives a sufficient condition for a space Z to support (Moore) self-reproduction. The condition is not that Z is computation-universal but that a universal Turing machine is wired in (implies computation-universality). This is a condition on just one module and its template neighbors and is therefore simple to apply.

Here is an interesting result:

Corollary 12.1 There exists a 7-state, 1-dimensional self-reproducing universal Turing machine configuration.

Proof. Embed the configuration h.K from the proof of the theorem in the cellular space of Corollary 4.2. Choose g to be a universal Turing machine function. ∎

Conclusions

Von Neumann exhibited a 29-state cellular space capable of supporting self-reproduction and computation-universality. His construction is very lengthy and complex. Codd was able to reduce the state count to 8 states per cell but his construction is also long and complicated. By greatly increasing cell complexity, Arbib was able to describe the processes simply. Here we have demonstrated both simple spaces and simple descriptions by deriving cellular spaces with low state count (e.g., 7 states per cell) and using recursive function theory for compact constructions in them. Whereas all previously published work in this area has been confined to two dimensions, the results here are most elegant in one dimension although applicable to two or more dimensions. It is also striking that nowhere have we had to define "construction" in a cellular space to obtain self-reproduction. In fact, computation-universality has been shown sufficient. However, a complete set of conditions insuring computation-universality has yet to be shown. Here we were able to refute one conjectured sufficient condition and offer two necessary conditions. Of course, we have not answered what is probably the most vexing question: Have we actually constructed machines, or are the procedures introduced here just fancy copying routines somehow distinct from construction?

References

1. von Neumann, J.: The Theory of Self-Reproducing Automata, edited by A.W.Burks, University of Illinois Press, Urbana and London (1966).

2. Thatcher, J.W.: "Universality in the von Neumann Cellular Model", to appear in a book edited by A.W.Burks on Cellular Automata.

3. Codd, E.F.: "Propagation, Computation and Construction in 2-dimensional Cellular Spaces", Technical Report, University of Michigan, March 1965.

4. Arbib, M.A.: "Simple Self-Reproducing Universal Automata", Information and Control 9 (1966) 177-189.

5. Holland, J.H.: "Universal Embedding Spaces for Automata", in Progress in Brain Research: Volume 17: Cybernetics of the Nervous System, (N. Wiener and J.P.Schadé, Eds.) Elsevier Publishing Company, New York (1965).

6. Wagner, E.G.: "An Approach to Modular Computers, I: Spider Automata and Embedded Automata", IBM Research Report RC-1107, January 28, 1964.

7. Smith, A.R., III: dissertation to be submitted to Stanford University.

8. Minsky, M.L.: "Size and Structure of Universal Turing Machines Using Tag Systems", Recursive Function Theory, Symposia in Pure Mathematics 5, Amer. Math. Soc. (1962).

9. Pattee, H.H.: "The Physical Bases of Coding and Reliability in Biological Evolution", in Towards a Theoretical Biology: I, Prolegomena, (C.H.Waddington, Ed.) Edinburgh University Press (1968).

10. Arbib, M.A.: "Self-Reproducing Automata--Some Implications for Theoretical Biology", in Towards a Theoretical Biology: II, Sketches, (see reference 9).

11. Moore, E.F.: "Machine Models of Self-Reproduction", Proc. Symp. Appl. Math., XIV, 17-32.

12. Lee, C.Y.: "A Turing Machine Which Prints Its Own Code Script", in Proc. Symp. Math. Theory of Automata, 1963, 155-164.