Dynamical systems with double recursion are undecidable

Mihai Prunescu

Abstract

A primitive type of dynamical system is introduced. It is shown that there is no decision procedure able to answer if any such dynamical system is ultimately zero.

A.M.S.-Classification: 12L05.

1 Introduction

Definition 1.1 A dynamical system with double recursion is a tuple \( \mathcal{A} = (A, f, 0, 1) \) consisting of: a finite set \( A \), a function \( f : A \times A \to A \), an element \( 1 \in A \) called start symbol and an element \( 0 \in A \) called white colour. By system or dynamical system we shall understand a dynamical system with double recursion, if not otherwise is explicitly stated. If a dynamical system \( \mathcal{S} \models \forall x, y \ f(x, y) = f(y, x) \) we say that the system \( \mathcal{S} \) is symmetric.

Definition 1.2 The development of \( \mathcal{A} \) is a function \( a : \mathbb{N} \times \mathbb{N} \to A \) defined as follows:

\[
a(i, j) = \begin{cases} 
1 & \text{if } i = 0 \lor j = 0, \\
f(a(i - 1, j), a(i, j - 1)) & \text{if } i > 0 \land j > 0.
\end{cases}
\]

If \( \mathcal{A} = \mathcal{S} \) is symmetric, the development is always a symmetric matrix: \( a(i, j) = a(j, i) \).

Definition 1.3 A dynamical system with double recursion \( \mathcal{A} \) is called ultimately zero if

\( \mathcal{A} \models \exists N \in \mathbb{N} \forall i, j \in \mathbb{N} \ i > 0 \land j > 0 \land i + j > N \to a(i, j) = 0. \)

The problem UW: given a dynamical system \( \mathcal{A} \), tell if it is ultimately zero.

The problem UWS: given a symmetric dynamical system \( \mathcal{S} \), tell if it is ultimately zero.

Theorem 1.4 The problems UW and SUW are not algorithmically solvable.

Of course, if SUW is not algorithmically solvable then neither shall be UW. However, I believe that starting with an easier sketch of proof for UW is helpful for the natural introduction in the matter.

For encoding the evolution of a Turing machine in the development of a dynamical system, we shall need the following description of the development:

Definition 1.5 The development \( a : \mathbb{N} \times \mathbb{N} \to A \) of \( \mathcal{A} \) shall be seen as an infinite matrix with \( a(0, 0) = a_{0,0} \) in the left upper corner. Denote by:

\[ D_n = \{a_{i,j} \mid i + j = n\} \]

the \( n \)-th diagonal. Then \( a = \bigcup D_n \) is a partition of the development of \( \mathcal{A} \).

*Brain Products, Freiburg, Germany, and Institute of Mathematics of the Romanian Academy, Bucharest, Romania. mihai.prunescu@math.uni-freiburg.de.
Before starting with the proof, we observe that dynamical systems can be completely algebraically modeled over finite fields. Just take some prime-power \( q > |A| \), fix an embedding \( A \subseteq \mathbb{F}_q \) such that 0 and 1 are now the corresponding constants in \( \mathbb{F}_q \), and find some polynomial \( F \in \mathbb{F}_q[x,y] \) whose associated polynomial function is an extension of \( f \). For \( f \) symmetric one can always find a symmetric polynomial \( F \). In this way the Theorem 1.4 gets a number-theoretic meaning:

**Corollary 1.6** Consider dynamical systems of the form \( \mathcal{A} = (\mathbb{F}_q, F, 0, 1) \) where \( \mathbb{F}_q \) are finite fields and \( F \in \mathbb{F}_q[x,y] \) are polynomials. Then it is undecidable if dynamical systems \( \mathcal{A} \) are ultimately zero. Moreover, this question remains undecidable if simultaneously restricted to the prime fields \( \mathbb{F}_p \) and to symmetric polynomials.

**Proof:** For a proof one should recall at most the interpolation with more than one variables. Take some arbitrary function \( f : \mathbb{F}_q^n \to \mathbb{F}_q \). To show that \( f \) is always polynomial, we will write it in the form:

\[
f(\vec{x}) = \sum_{\vec{a} \in \mathbb{F}_q^n} d_{\vec{a}}(\vec{x}) f(\vec{a}),
\]

where \( d_{\text{vec}(\vec{x})} = 0 \) if \( \vec{x} \neq \vec{a} \) and \( d_{\vec{a}}(\vec{a}) = 1 \). For \( n = 1 \) take:

\[
d_{\vec{a}}(x) = \prod_{b \in \mathbb{F}_q \setminus \{a\}} \frac{(x-b)}{(a-b)}.
\]

Take \( d_{\vec{x}}(\vec{x}) = d_{a_1}(x_1)d_{a_2}(x_2) \ldots d_{a_n}(x_n) \).

Some historical words. One can see a dynamical system with double recursion as the tiling of the quarter of plane with tiles obtained by modifying squares such that the only relevant compatibilities to solve for putting down a new stone are the neighbors from North and West. Seen in this way, these results are related with the classical results concerning undecidability of questions about finite sets of tiles, see [1], [6], [4]. Aperiodic tilings related to some linear dynamical systems over finite fields have been described by the author in [2], another recent preprint.

The unique classical ingredient used here is the Theorem of Rice, see [3], in its modern formulation concerning sets of (codes of) Turing machines, as stated for example in [5].

## 2 The problem UW

In order to prove that the problem UW is not algorithmically solvable we will interpret instances of the Halting Problem in instances of UW.

**Definition 2.1** An instance of the Halting Problem is a pair \((M, w)\) where \( M = (\Sigma, Q, q_0, q_s, \bar{b}, \delta) \) is a Turing machine and \( w \in \Sigma^* \) is an input for \( M \). Here the tape of \( M \) is infinite in both directions, \( \Sigma \) is the alphabet of \( M \), \( Q \) is \( M \)'s set of states, \( q_0 \) and \( q_s \) are the start state and respectively the stop state, \( \bar{b} \in \Sigma \) is the blank symbol, and \( \delta : \Sigma \times Q \to \Sigma \times Q \times \{R, L, S\} \) is the transition function.

**Lemma 2.2** To every instance \((M, w)\) of the Halting Problem one can algorithmically associate a dynamical system \( \mathcal{A} = (A, f, 0, 1) \) such that \( \mathcal{A} \in \text{UW} \) if and only if for input \( w \) the machine \( M \) stops and after stopping the tape contains only blank symbols.

**Proof:** If the Lemma [2.2] is true then UW is not algorithmically solvable. This is true because to stop with a clear tape is an undecidable property, according to the Theorem of Rice.

Let \((M, w)\) be an instance of the Halting Problem. The dynamical system \( \mathcal{A} \) shall be constructed step by step following its development:
• The start symbol of $A$ is a new letter $1$ that has nothing to do with the Turing machine. On the other hand the colour white $0$ of $A$ and the blanc symbol $\bar{b}$ of $M$ are the same.

• Let the input $w \in \Sigma^*$ be the word $w_1 \ldots w_n$. Using a set $U$ of new letters one defines $f$ such that the content of a diagonal $D_w$ of the development is exactly:

$$1 \ 0 \ 0 \ \delta_0 \ w_1 \ \ldots \ w_n \ 0 \ 0 \ 1$$

The letters of $U$ shall be used only for this goal, and then never again. The simulation of the Turing machine starts with this diagonal.

• The just constructed diagonal $D_w$ is said to be a diagonal of type 0. Starting with $D_w$ diagonals are alternatively of types 0, 1, 0, 1, and so on. Successive diagonals of type 0 simulates successive configurations of the Turing machine. The diagonals of type 1 between them are used to transfer the information.

• The alphabet used for diagonals of type 0 contains $\Sigma \cup (\Sigma \times \mathbb{Q})$. We denote letters from $\Sigma \times \mathbb{Q}$ with $\delta_i$. The meaning of the letter "$(a, q)$" is that the head of $M$ reads $a$ with $M$ being in the state $q$.

• If $\Gamma_0$ is the alphabet used for diagonals of type 0, the alphabet used for diagonals of type 1 will be $\Gamma_1 = (\Gamma_0 \times \Gamma_0 \setminus \{(0, 0)\}) \cup \{0\}$.

• The function $f$ is defined on diagonals of type 0 in the obvious way $f(a, b) = (a, b)$ if at least one of $a$ and $b$ are not 0 or 1, and $f(0, 0) = f(1, 0) = f(0, 1) = 0$.

• The function $f$ is defined on diagonals of type 1 such that: if the element $a_{i,j}$ of the last diagonal of type 0 simulated a certain cell of the tape of $M$ at a given time $k$, then the cell $a_{i+1,j+1}$ simulates the same cell at the time $k+1$. The following example shows how the diagonal of type 1 in between makes possible that the element $a_{i+1,j+1}$ gets information from three successive cells: $a_{i,j}$ and the elements simulating their neighbors.

$$\begin{array}{ccc}
\delta & b \\
\delta_0 & (\delta, b) \\
a & (a, \delta) \\
c
\end{array}$$

• Every diagonal of type 0 is with two cells longer then the precedent one and the head makes one step per time, so there is no danger that the simulation leaves the carpet or even that the simulation meets the wall of ones.

• For the special letter $\delta = (0, q_s)$ we define $f$ such that $f(\delta, 0) = f(0, \delta) = 0$. This makes the development ultimately zero if and only if the machine stops with clear tape.

• Now take $A$ to be $\{1\} \cup U \cup \Gamma_0 \cup \Gamma_1$ and $f : A \times A \to A$ to respect all the conditions given above.

\[\square\]

3 The problem SUW

Definition 3.1 Let $\Gamma \neq \emptyset$ be a set and $\equiv$ be the partition of $\Gamma \times \Gamma$ consisting of the following sets: for all $a \in \Gamma$ the singleton sets $\{(a, a)\}$ and for all $a, b \in \Gamma$ with $a \neq b$ the two-element sets $\{(a, b), (b, a)\}$. Then $\equiv$ is an equivalence relation over $\Gamma \times \Gamma$. Consider the set of equivalence classes:

$$\Gamma \cdot \Gamma = (\Gamma \times \Gamma)/\equiv$$

which is the set of unordered pairs of elements of $\Gamma$. We denote the equivalence class of $(a, b)$ with $[a, b]$ and call this the unordered pair of $a$ and $b$. 
Lemma 3.2. To every instance \((M, w)\) of the Halting Problem one can algorithmically associate a symmetric dynamical system \(\mathcal{S} = (S, f, 0, 1)\) such that \(\mathcal{S} \in SUW\) if and only if for input \(w\): (the machine \(M\) stops with white tape without having done any step in the negative side of the tape) or (the machine \(M\) shall make a step in the negative side of its tape and the first time when \(M\) makes such a step the tape of \(M\) is clear).

Proof: If the Lemma 2.2 is true then SUW is not algorithmically solvable. This is true because the given condition is an undecidable property, according to the Theorem of Rice.

Before starting the construction, I shall shortly explain the arising difficulties. We construct again the function \(f\) together with its development. The function \(f\) and the development are both symmetric, so we will only consider the right half of the development. It shall be again so, that on some special (half of) diagonals one simulates successive configurations of the Turing machine on input \(w\). The function \(f\) being symmetric, one cannot directly make the difference between Left and Right. To overcome this difficulty one can try to double the number of letters of \(\Gamma\) and write every letter \(c\) as \(cc'\). The function \(f\) should now act symmetrically on diagonals of type 0, so we define \(f(a, b)\) to be the unordered pair \([a, b]\). This strategy is not sophisticated enough: if we look at words \(aba\) and \(bab\) on a diagonal of type 0, they both produce a word \(xx\) on the following diagonal of type 1, where \(x = [a, b]\). This means that this encoding may lose essential information.

The solution shall be to triple the number of letters and to encode every letter \(c\) with a sequence \(cc'c''\), where \(c'\) and \(c''\) are used only for this goal.

- The start symbol of \(\mathcal{S}\) is a new letter 1 that has nothing to do with the Turing machine. Also the the colour white 0 of \(\mathcal{S}\) is now a new letter.
- Let the input \(w \in \Sigma^*\) be the word \(w_1 \ldots w_n\). Using a set \(U\) of new letters one defines \(f\) symmetrically such that the content of a diagonal \(D_w\) of the development is exactly:

\[
10^9 w_n''w_n'w_n \ldots w_1''w_1'w_1 000 \delta_{0}^{0} \delta_{0}^{0} w_1 w_1' w_1'' \ldots w_n w_n'' 0^9 1
\]

Here means \(0^9\) a word built up by 9 zeros. The letters of \(U\) shall be used only for this goal, and then never again. The simulation of the Turing machine starts with this diagonal.

- The just constructed diagonal \(D_w\) is again a diagonal of type 0. This time there are 8 types of diagonals: types 0, 1, \ldots, 7. Starting with \(D_w\) diagonals are of types 0, 1, \ldots, 7, 0, 1, \ldots, 7, and so on. Successive diagonals of type 0 simulates successive configurations of the Turing machine. The diagonals of types 1, \ldots, 7 between them are used to transfer the information from a simulation to the next.

- The alphabet \(\Gamma_0\) used for diagonals of type 0 contains three disjoint copies of the set \(\Sigma \cup (\Sigma \times Q) \setminus \{\overline{b}\}\). The letter \(c \in \Sigma \setminus \{\overline{b}\}\) is called \(c'\) in \(\Sigma'\) and \(c''\) in \(\Sigma''\), the same for the letters \(\delta \in \Sigma \times Q\). The fact that \(c \in \Sigma\) is contained in a cell of the Turing machine is encoded by the word \(cc'c''\) on a diagonal of type 0. On the right side of the development: The fact that the head of the Turing machine in state \(q\) reads a cell of content \(a\) is encoded by the word \(\delta \delta' \delta''\) on a diagonal of type 0, where \(\delta = (a, q) \in \Sigma \times Q\). The blanc symbol \(\overline{b}\) as content of a cell of the Turing machine is always encoded by the word \(000\) in the simulation by the word \(000\) on a diagonal of type 0. In the left side of the development the codes are \(c''c'c\), \(\delta'' \delta' \delta\) and \(000\) respectively.

- Let \(\Gamma_0\) be the alphabet used for diagonals of type 0. For \(i = 1, 2, \ldots, 7\) the alphabet used for diagonals of type \(i\) will be \(\Gamma_i = (\Gamma_{i-1} \setminus \{(0, 0)\}) \cup \{0\}\).

- The function \(f\) is defined on diagonals of type \(i = 0, 1, \ldots, 6\) in the obvious way \(f(a, b) = [a, b]\) if at least one of \(a\) and \(b\) are not 0 or 1, and \(f(0, 0) = f(1, 0) = f(0, 1) = 0\).

- The function \(f\) is defined on diagonals of type 7 such that: if the element \(a_{i,j}\) of the last diagonal of type 0 contains a letter \(c, c', c'', \delta, \delta', \delta''\) or 0 that appears in a subword of length
3 simulating a cell of the tape of \( M \) at a given time \( k \), then the element \( a_{i+4,j+4} \) of the development shall be the corresponding letter of the diagonal word simulating the configuration of \( M \) at time \( k + 1 \). This is done like in the following example. Let \( a_{i,j} = \delta' \in \Gamma_0 \) be a part of the following segment of simulation in a diagonal of type 0: \( \ldots c''c'\delta\delta'\delta''dd'd'' \ldots \), and suppose that in the next configuration the tape-cell containing \( \delta \) shall contain \( e \in \Sigma \). As proved in the postponed Lemmas \[ \text{4.4} \] one has on the first coming diagonal of type 7: \( a_{i+3,j+4} = \alpha \) and \( a_{i+4,j+3} = \beta \) with \( \alpha, \beta \in \Gamma_7 \), such that \( \alpha \) encodes the word \( c'c''\delta\delta'\delta''dd'^3 \) or its reverse and \( \beta \) encodes the word \( c'c''\delta\delta'\delta''dd'^3 \) or its reverse. One has either the words themselves (if we look to the left-hand side of the development), or the reversed words (if we look to the left-hand side of the development). If we are in the right-hand side the matching of the encoded words looks like:

\[
\begin{array}{cccccccc}
c & c' & c'' & \delta & \delta' & \delta'' & d & d' \\
c' & c'' & \delta & \delta' & \delta'' & d & d' & d''
\end{array}
\]

If we are in the left-hand side of the development, the matching is:

\[
\begin{array}{cccccccc}
d'' & d' & d & d'' & \delta'' & \delta' & \delta & \delta''
\end{array}
\]

\[
\begin{array}{cccccccc}
d' & d & \delta'' & \delta' & \delta & \delta'' & c'
\end{array}
\]

In both cases the matching is a word of length 7 centered in \( \delta' \), so the value \( f(\alpha, \beta) \) is uniquely determined to be \( e' \), where \( e \in \Sigma \) is the letter that shall replace \( \delta \) in that tape-cell in the next configuration. The same arguments work for every connected subword of length 8 that is disjoint from the central 000 word, like \( c'c''\delta\delta'\delta''dd'd''e \), and so on.

- Every diagonal of type 0 is with eight elements longer then its predecessor of type 0 (four elements in the left-hand side and four elements in the right-hand side) and the simulation needs at most three elements more per step, so there is no danger that the simulation leaves the carpet or even that the simulation meets the wall of ones.

- If the connected subword \( \delta'' \delta'' \) with the special letter \( \delta = (0, q_s) \) arrises, then we define \( f \) such that the corresponding development elements in the next diagonal of type 0 are 0.

- For the connected subwords of length 8 containing the central 000 the function \( f \) is defined such that: Words of the type \( c''c'000cc'c'' \) are preserved in the next configuration. Words \( \delta''\delta000\delta'' \) are replaced with \( c''c'000cc'c'' \) if \( \delta = (a, q) \rightarrow (e, q', R) \). Words \( \delta''\delta000\delta'' \) are replaced with \( a''a'000aa'a'' \) if \( \delta = (a, q) \rightarrow (e, q', L) \). As asked in the statement of this Lemma the computation dies by the first movement in the left-hand side of the tape.

- Now take \( A \) to be \( \{1\} \cup U \cup_{i=0}^r \Gamma_i \) and \( f : A \times A \rightarrow A \) to respect all the conditions given above.

\[ \square \]

### 4 Symmetric codes

**Definition 4.1** Let \( \Gamma_0 \) be a finite alphabet with \( \geq 2 \) letters and \( 0 \in \Gamma_0 \) a special letter. We define the sequence of alphabets \( \Gamma_i \), such that \( \Gamma_{i+1} = (\Gamma_i \setminus \{0, 0\}) \cup \{0\} \) and \( f : \Gamma_i \times \Gamma_i \rightarrow \Gamma_{i+1} \) such that \( f(a, b) = [a, b] \) if at least one of the arguments is not 0 and \( f(0, 0) = 0 \). For an alphabet \( \Gamma \) let \( \Gamma^* \) be the set of words over \( \Gamma \) and \( \Gamma^{\geq k} \) the set of words of length \( \geq k \) over \( \Gamma \). The set of words of length \( k \) shall be simply denoted \( \Gamma^k \).

**Definition 4.2** Let \( \pi_i : \Gamma_{i+1}^{\geq 2} \rightarrow \Gamma_{i+1} \) given as \( \pi_i(w_1 \ldots w_n) = f(w_1, w_2)f(w_2, w_3) \ldots f(w_{n-1}, w_n) \). Let \( \pi : \Gamma_0^{\geq 2} \rightarrow \Gamma_7 \) given as \( \pi(w) = \pi_0\pi_5\pi_4\pi_3\pi_2\pi_1\pi_0(w) \). We call the words \( w, \pi_0(w), \pi_1\pi_0(w), \ldots, \pi_5\pi_4\pi_3\pi_2\pi_1\pi_0(w) \) the levels of the code.
Definition 4.3. Now let \( \Gamma_0 \) be the alphabet defined in the section \( \Xi \). Let \( E \) be the set of all words in \( \Gamma_0^8 \) that can possible arise during a simulation. They are exactly the connected subwords of length 8 in all words \( aa'a'a''b'b''c'c''d'd'' \) where \( a, b, c, d \in \Sigma \cup \Sigma \times Q \) are not necessarily different, and if some \( e \in \{a, b, c, d\} \) are 0 then the corresponding \( e' = e'' = 0 \). The restriction of \( \pi \) to \( E \to \Gamma_7 \) shall be simply called \( \pi \).

Definition 4.4. Let \( S \) be the set of connected subwords of length 8 in all words \( ba''a'a000aa'a''b \) where \( a, b \in \Sigma \cup \Sigma \times Q \) are not necessarily different, and if \( a = 0 \) then the corresponding \( a' = a'' = 0 \). Again the restriction of \( \pi \) to \( S \to \Gamma_7 \) shall be simply called \( \pi \).

Definition 4.5. For a word \( w \in \Gamma^* \), \( w = w_1 \ldots w_n \), call \( \sigma(w) \) the reversed word \( w_n \ldots w_1 \).

Lemma 4.6. For all words \( v \in E \cup S \) and \( w \in (\Sigma \cup (\Sigma \times Q))^8 \), if \( \pi(w) = \pi(v) \) then \( w = v \) or \( w = \sigma(v) \).

Proof. The proof works as follows: We check for all types of words in \( E \cup S \) that only they and their reverses lead to their symmetrically iterated codes.

Words in \( E \):

It is enough to check the worst cases with letter repetitions. Start with the word \( ccc'e'cc'e'c'c' \). The levels of encoding are as follows: \( x_1x_2x_3x_1x_2x_3x_1x_2, y_1y_2y_3y_1y_2y_3y_1, z_1z_2z_3z_1z_2z_3, v_1v_2v_3v_1v_2, t_1t_2t_3t_1, u_1u_2u_3, s_1s_2, \alpha \), where \( \alpha \in \Gamma_7 \). Starting with \( \alpha \), one gets \( \alpha = [s_1, s_2] \) so the two possibilities are \( s_1s_2 \) and \( s_2s_1 \). The first one leads directly to \( w \), the other one directly to \( \sigma(w) \); there are not other possibilities to reconstruct the word.

For a complete proof of the Lemma, one has to check the following worst cases: (a) All the connected subwords of length 8 in \( ccc'e'000ccc'e'(000000) \) and (b) All the connected subwords of length 8 in \( ccc'e'cc'e'c'e'000 \). All this cases have the following common property: at all levels of coding, including the level 0, two successive letters are equal if and only if they are 0.

Words in \( S \):

The words occuring here are the exceptions in our symmetric encoding: They do not enjoy the property that successive letters at every level are equal if and only if they are both 0, but they are however well behaving even by being the only words that don’t enjoy this property. Look at \( ccc'e'0000cc'e' \). The levels of its symmetric code are: \( x_1x_2x_3000x_2x_3, y_1y_2y_3y_1y_2y_3y_1, z_1z_2z_3z_1z_2z_3, v_1v_2v_3v_1v_2, t_1t_2t_3t_1, u_1u_2u_3, s_1s_2, \alpha \). In decoding we have again the choice \( s_1s_2 \) or \( s_2s_1 \). If we choose \( s_1s_2 \), that can backwards develop only in \( t_1t_2t_2 \), and so on. One easily checks all other words in question. ☐

Just some commentaries at the end. Letters have been encoded by directed words of the form \( ccc'e'' \) or \( \delta \delta' \delta'' \) in order to make the difference between the left and the right neighbor in a symmetric dynamical system. To symmetrically encode words of length 8 one needs 7 supplementary types of diagonals. A letter of type 7 encodes a word of length 8 and so has information from three successive simulated Turing cells. The common part of two successive subwords of length 8 has length 7 and so always has a central letter: this is the letter to copy or replace on the next coming diagonal of type 0.

Both constructions done here work in polynomial time.

One can do very spectacular experiments with dynamical systems in three variables, developed with the rules \( a_{i,j} = a_{0,i} = 1 \) and \( a_{i,j} = f(a_{i-1,j}, a_{i-1,j-1}, a_{i,j-1}) \) and using for example functions \( f(x, y, z) \) which are symmetric in \( x \) and \( z \). This class of dynamical systems is also undecidable because it trivially contains the problem SUW given above.
References

[1] Egon Börger, Erich Grädel, Yuri Gurevich: The classical decision problem. Perspectives in Mathematical Logic, Springer Verlag, 1997.

[2] Mihai Prunescu: Self-similar carpets over finite fields. Talk at CiE 2007, Siena, Italy. Submitted.

[3] Gordon H. Rice: Classes of recursively enumerable sets and their decision problems. Transactions of the American Mathematical Society, 74, 358 - 366, 1953.

[4] Raphael Robinson: Undecidability and non-periodicity for tilings in the plane. Inventiones Mathematicae 12, 177 - 209, 1971.

[5] Uwe Schöning: Theoretische Informatik - kurz gefaßt. Spektrum Akademischer Verlag, Heidelberg, Berlin, 1997.

[6] Hao Wang: Dominoes and the $\forall \exists \forall$-case of the decision problem. Proceedings Symposia on Mathematical Theory of Automata, 23 - 55, Brooklin Polytechnic Institute, New York, 1962.