# Studies in Computational Intelligence 459

**Editor-in-Chief**

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

For further volumes:
http://www.springer.com/series/7092

Kyandoghere Kyamakya, Wolfgang A. Halang,
Wolfgang Mathis, Jean Camberlain Chedjou,
and Zhong Li (Eds.)

# Selected Topics in Nonlinear Dynamics and Theoretical Electrical Engineering

Springer

*Editors*
Univ. -Prof. Dr.-Ing. Kyandoghere Kyamakya
Alpen-Adria Universität Klagenfurt
Klagenfurt
Österreich

Ass. Prof. Dr.-Ing. Jean Camberlain Chedjou
Alpen-Adria Universität Klagenfurt
Klagenfurt
Österreich

Prof. Dr. Dr. Wolfgang A. Halang
Fernuniversität Hagen
Philipp-Reis-Gebäude
Hagen
Deutschland

apl. Prof. Dr. habil. Zhong Li
Fernuniversität Hagen
Philipp-Reis-Gebäude
Hagen
Deutschland

Prof. Wolfgang Mathis
Institut für Theoretische Elektrotechnik
Leibniz Universität Hannover
Hannover
Deutschland

ISSN 1860-949X
e-ISSN 1860-9503
ISBN 978-3-642-34559-3
e-ISBN 978-3-642-34560-9
DOI 10.1007/978-3-642-34560-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012951168



Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Chapter 22
# Consecutive Repeating State Cycles Determine Periodic Points in a Turing Machine

Michael Stephen Fiske

**Abstract.** The Turing machine is studied with new methods motivated by the notion of recurrence in classical dynamical systems theory. The state cycle of a Turing machine is introduced. It is proven that each consecutive repeating state cycle in a Turing machine determines a unique periodic configuration (point) and vice versa. This characterization is a *periodic point* theorem for Turing machines. A Turing machine is defined to be periodic if it has at least one periodic configuration or it only has halting configurations. Using the notion of a prime directed edge and a mathematical operation called edge pattern substitution, a search procedure finds consecutive repeating state cycles. If the Turing machine is periodic, then this procedure eventually finds each periodic point or this procedure determines that the machine has only halting configurations. New mathematical techniques are demonstrated such as edge pattern substitution and prime directed edge sequences that could be quite useful in the further study of the aperiodic Turing machines. The aperiodicity appears to play an integral role in the undecidability of the Halting problem.

## 22.1 Introduction

New results are achieved here by analyzing the Turing machine from a dynamical systems point of view. Classical dynamical systems theory has been successful by first understanding the periodic behavior and then studying more general recurrent behavior (as in [3], [4], [7], and [11] ). This paper follows the classical approach, by finding a notion of recurrence that characterizes periodic configurations of a Turing machine.

In the next section, the Turing machine is briefly reviewed and then some definitions are presented for studying its recurrent behavior. The Turing machine is represented as $(Q, A, \eta)$ where $Q$ is a finite set of states, $A$ is a finite alphabet and

Michael Stephen Fiske
Aemea Institute, San Francisco, CA, 94129
e-mail: mf@aemea.org

K. Kyamakya et al. (Eds.): Selected Topics in Nonlinear Dynamics, SCI 459, pp. 393–417.
DOI: 10.1007/ 978-3-642-34560-9_22 

the program $\eta$ is a function $\eta : Q \times A \to Q \times A \times \{-1,+1\} \cup \{\mathcal{H}\} \times A \times \{0\}$. A Turing machine configuration $(q,k,T)$ is a triplet, where $q$ is the current state of the machine, the function $T : Z \to A$ represents the tape, where $Z$ is the integers and $k$ is an integer that is the tape head location. A machine starts program execution at configuration $(q,k,T)$ where $T(k) = \alpha$ and $\eta$ determines the next configuration according to three cases:

1. $(r,k-1,S)$ if $\eta(q,\alpha) = (r,\beta,-1)$
2. $(r,k+1,S)$ if $\eta(q,\alpha) = (r,\beta,+1)$
3. $(\mathcal{H},k,S)$ if $\eta(q,\alpha) = (\mathcal{H},\beta,0)$

such that for all three cases the new tape $S(j) = T(j)$ whenever $j \neq k$ and $S(k) = \beta$.

Case 1 means that the machine moves to state $r$, replaces alphabet symbol $\alpha$ with symbol $\beta$ at tape square $k$ and then moves the tape head left $-1$ to tape square $k-1$. Case 2 means the same as Case 1 except the tape head moves right $+1$ to tape square $k+1$. Case 3 means that the machine reaches a unique halting state $\mathcal{H}$ and the program execution halts.

A configuration $(q,k,T)$ is called a halting configuration if machine execution starts at $(q,k,T)$, and after a finite number of execution steps, the machine reaches the halting state. A configuration $(q,k,T)$ is periodic if after the execution of $n$ computational steps of the Turing machine, the new configuration $(r,j,S)$ has the same state and the same tape contents. This means $r = q$ and the same tape contents means $T(x+k) = S(x+j)$ for every integer $x$. In this case, $(q,k,T)$ is a periodic configuration. Thus, all periodic configurations are immortal. A Turing machine is called *periodic* if it has at least one periodic configuration or it only has halting configurations.

The state cycle is a notion of recurrence for the Turing machine. A state cycle is a non-halting execution sequence of input commands $(q_0,a_0) \to (q_1,a_1) \to \cdots \to (q_{N-1},a_{N-1}) \to (q_N,a_N)$, such that $q_0 = q_N$ and where pair $(q,\alpha)$ in $\eta(q,\alpha)$ is called an input command. A state cycle is called a prime state cycle if it contains no proper state subcycles. A consecutive repeating state cycle is a state cycle that repeats itself twice, where the second repeat immediately follows the first: $(q_0,a_0) \to (q_1,a_1) \to \cdots \to (q_{N-1},a_{N-1}) \to (q_0,a_0) \to (q_1,a_1) \to \cdots \to (q_{N-1},a_{N-1})$.

In theorem 22.2, a *periodic point theorem that holds for any Turing machine* is proved: a consecutive repeating state cycle uniquely determines a periodic configuration of the Turing machine; and vice versa, a periodic configuration uniquely determines a consecutive repeating state cycle. Thus, to search for a periodic configuration, a procedure may look for consecutive repeating state cycles.

In definition 22.12, the prime directed edge is defined. The prime directed edge represents, over a window of execution, a Turing machine program visiting only one state twice and visiting the other states one time or not at all. A prime directed edge contains a prime state cycle. Pattern matching determines how two prime directed edges are glued together. When prime directed edges are glued together, this is called link matching. The link matching is used to build prime directed edge sequences. Based on $|Q|$ and $|A|$, an upper bound on the number of prime directed

edges is computed. The prime directed edge sequences cover all Turing program execution possibilities. As a consequence, the number of prime directed edges is a useful measure of the Turing machine complexity. For a given Turing machine, a procedure for finding all prime directed edges is shown. After this, a periodic point search procedure is described whereby prime directed edges are link matched together to form prime directed edge sequences. This procedure searches for consecutive repeating state cycles inside the edge sequences.

In [9], Kurka conjectured that any Turing machine that has no halting configurations has a periodic configuration. In [2], Blondel et al demonstrated that some Turing machines have only aperiodic immortal configurations. Furthermore, Blondel et al. showed in [2] that determining whether a given counter machine has a periodic orbit in configuration space is undecidable.

In this context, there are two main results. First, that each consecutive repeating state cycle determines a unique periodic point in a Turing Machine and vice versa. Second, using this first result, if a Turing machine is periodic, then the search procedure can determine whether the machine only has halting configurations or, if not, then this procedure finds a periodic configuration of this machine. Furthermore, the procedure demonstrated can be used to find periodic configurations of any length when they exist. Finally, the mathematical tools developed here – e.g. the prime directed edge, the edge substitution operator, link matching, and prime directed edge sequences – can be used to study the aperiodic immortal configurations, which could help better understand the dynamics of the Halting problem [12].

## 22.2 Turing Machines & Periodic Configurations

The Turing Machine is defined here so that its program $\eta$ is explicitly represented as a function.

**Definition 22.1.** *Turing Machine*
A Turing machine is a triple $(Q, A, \eta)$ where

1. $Q$ is a finite set of states that does not contain a unique halt state $\mathcal{H}$.
2. The machine execution starts in an initial state $s$ and $s$ lies in $Q$.
3. $A$ is a finite set of alphabet symbols that are read from and written to the tape.
4. $-1$ and $+1$ represent advancing the tape head to the left or right square, respectively.
5. $\eta : Q \times A \to Q \times A \times \{-1, +1\} \cup \{\mathcal{H}\} \times A \times \{0\}$ is a function. $\eta$ acts as the program for the Turing machine. For each $q$ in $Q$ and $\alpha$ in $A$, $\eta(q, \alpha) = (r, \beta, x)$ describes how machine $(Q, A, \eta)$ executes one computational step. When in state $q$ and scanning alphabet symbol $\alpha$ on the tape:
   (a) Machine $(Q, A, \eta)$ changes to state $r$.
   (b) Machine $(Q, A, \eta)$ rewrites alphabet symbol $\alpha$ as symbol $\beta$ on the tape.
   (c) If $x = -1$, then machine $(Q, A, \eta)$ moves its tape head one square to the left on the tape and is subsequently scanning the symbol in this square.

(d) If $x = +1$, then machine $(Q, A, \eta)$ moves its tape head one square to the right on the tape and is subsequently scanning the symbol in this square.

(e) If $r = \mathcal{H}$, machine $(Q, A, \eta)$ reaches halting state $\mathcal{H}$ and halts.

**Definition 22.2.** *Turing Machine tape*
The Turing machine tape $T$ is represented as a function $T : Z \to A$ where $Z$ denotes the integers. The tape $T$ is $M$-bounded if there exists a bound $M > 0$ such that $T(k) = T(j)$ whenever $|k|, |j| \geq M$.

The Turing machine definitions in [5], [12] assume the initial tape, before program execution begins, is $M$-bounded and the tape contains only blank symbols, denoted here as #, outside the bound. In this paper, the tape is not assumed to be $M$-bounded, unless this is explicitly stated for a particular case. The symbol on the $k$th square of the tape is $T(k)$.

**Definition 22.3.** *Turing Machine Configuration with tape head location*
Let $(Q, A, \eta)$ be a Turing machine with tape T. A configuration is an element of the set $C = (Q \cup \{\mathcal{H}\}) \times Z \times \{T : T \text{ is tape with range } A\}$. If $(q, k, T)$ is a configuration, then $k$ is called the tape head location.

Consider the configuration $(p, 2, \ldots \#\#\alpha\underline{\beta}\#\#\ldots)$. The 1st coordinate indicates that the Turing machine is in state $p$. The 2nd coordinate indicates that its tape head is currently scanning tape square 2, denoted as $T(2)$. The 3rd coordinate indicates that tape square 1 contains symbol $\alpha$, tape square 2 contains symbol $\beta$, and all other tape squares contain the # symbol. Sometimes a periodic configuration $p = (q, k, T)$ is called a periodic point or immortal periodic point.

**Definition 22.4.** *Computational Period and Hyperbolic Degree*
Consider immortal periodic point $p$. If the machine starts its execution at point $p$, then the minimal number of computational steps, denoted $C(p)$, for the machine to return to point $p$ is called the computational period of $p$. Observe that $C(p) = R + L$ where $R$ and $L$ denote the number of right and left tape head moves respectively. Define the hyperbolic degree of $p$ as $m(p) = R - L$. If $m \neq 0$, the periodic point is called *hyperbolic*. Otherwise, $p$ is called non-hyperbolic.

The computational period is motivated by the classical dynamical systems definition of the period of a point $p$ in $X$ for an autonomous (e.g. [6]) dynamical system $f : X \to X$ where $X$ is a topological space, $f$ is a function and $m$ is the minimal positive integer such that $f^m(p) = p$ . The hyperbolic degree is analogous to the classical dynamical systems definition of a hyperbolic periodic point of $f : X \to X$ when $X$ is a manifold and $f$ is differentiable.

A *pattern* W is a finite sequence of alphabet symbols chosen from A. In other words, $W : \{0, 1, \ldots, n-1\} \to A$. The length of $W = n$ and is denoted as $|W| = n$. The $k$th element of the pattern $W$ is denoted as $W(k)$ or $w_k$. Thus, pattern $W$ is sometimes explicitly expressed as $w_0 w_1 \ldots w_{n-1}$. $S$ is a *subpattern* of $W$ if $S = w_j w_{j+1} \ldots w_{k-1} w_k$ for some $j$ and $k$ satisfying $0 \leq j \leq k \leq n-1$ and the length of $S = k - j + 1$. A pattern represents a finite sequence of the tape.

The expression $[4,\overline{12}\underline{1}\overline{212}]$ represents the point $p$ where the machine is in state 4; the tape head is located at the underlined 1; the tape to the right of the tape head contains the periodic pattern 212 212… ; and the tape to the left of the tape head contains the periodic pattern …12 12 12.

*Example 22.1. Non-Hyperbolic Periodic Point*

The state set is $Q = \{q,r\}$ and the alphabet set is $A = \{1,2\}$. The halting state is $\mathcal{H}$. $\eta$ is defined below.

$\eta(q,1) = (\mathcal{H},1,0)$ $\eta(q,2) = (r,2,-1)$
$\eta(r,1) = (q,2,+1)$ $\eta(r,2) = (q,1,+1)$

Consider program execution steps: $[r,x\underline{2}2y] \mapsto [q,x1\underline{2}y] \mapsto [r,x\underline{1}2y] \mapsto [q,x2\underline{2}y] \mapsto [r,x\underline{2}2y]$, where $x$ is any infinite left tape sequence chosen from $A$ and $y$ is any infinite right tape sequence. The tape head moves for this non-hyperbolic immortal periodic point are $[+1,-1,+1,-1]$. All points of the form $p = [r,x\underline{2}2y]$ are non-hyperbolic periodic points with period 4.

*Example 22.2. Hyperbolic Periodic Point*

The state set is $Q = \{q,r,s,t,u,v,w,x\}$ and the alphabet set is $A = \{1,2\}$. $\eta$ is defined below.

$\eta(q,1) = (r,1,+1)$ $\eta(q,2) = (\mathcal{H},2,0)$
$\eta(r,1) = (\mathcal{H},1,0)$ $\eta(r,2) = (s,2,+1)$
$\eta(s,1) = (t,1,+1)$ $\eta(s,2) = (\mathcal{H},2,0)$
$\eta(t,1) = (\mathcal{H},1,0)$ $\eta(t,2) = (u,2,+1)$
$\eta(u,1) = (\mathcal{H},1,0)$ $\eta(u,2) = (v,1,+1)$
$\eta(v,1) = (\mathcal{H},1,0)$ $\eta(v,2) = (w,2,+1)$
$\eta(w,1) = (\mathcal{H},1,0)$ $\eta(w,2) = (x,1,-1)$
$\eta(x,1) = (\mathcal{H},1,0)$ $\eta(x,2) = (q,2,+1)$

The point $p = [q,\overline{12}\ \underline{1}\ \overline{212222}]$ is an immortal periodic point with computational period 8 and hyperbolic degree 6.

**Definition 22.5.** *Window of Execution*
Consider the next $N$ computational steps of a Turing machine. The window of execution, denoted as $[\ell,\mu]$ or $[\ell,\ell+1,\ldots,\mu]$, is the sequence of integers representing the tape squares that the tape head visited during these $N$ computational steps. The length of the window of execution is $\mu - \ell + 1$ which is also the number of distinct tape squares visited by the tape head during these $N$ steps. To express the window of execution for the next $n$ computational steps, the lower and upper bounds are expressed as a function of $n$: $[\ell(n),\mu(n)]$. If $j \le k$, then $[\ell(j),\mu(j)] \subseteq [\ell(k),\mu(k)]$ which follows from the definition.

The purpose of the window of execution is to describe only the portion of the tape that the tape head visits during the next $N$ computational steps. This is useful because all tape squares outside the window of execution remain unchanged during those $N$ steps. Since the tape squares may be renumbered without changing the results of the machine execution, for convenience it is often assumed that the machine

starts execution at tape square 0. In example 22.2, during the next 8 computational steps – that is, one cycle of the immortal periodic point – the window of execution is [0, 6] and its length is 7.

## 22.3 State Cycles

This section introduces state cycles and consecutive repeating state cycles. Subsequently, a proof shows that a consecutive repeating state cycle determines a unique periodic point and a periodic point determines a unique consecutive repeating state cycle.

**Definition 22.6.** *State Cycle*
Consider $N$ execution steps of Turing Machine $(Q, A, \eta)$. After each execution step, the machine is in some state $q_k$ and the tape head is pointing to some alphabet symbol $a_k$. Relabel the indices of the states and the alphabet symbols if necessary and assume the machine has not halted after $N$ execution steps. This execution sequence of input commands is $(q_0, a_0) \mapsto (q_1, a_1) \mapsto \ldots \mapsto (q_{N-1}, a_{N-1}) \mapsto (q_N, a_N)$, where each pair $(q_k, a_k)$ executed as $\eta(q_k, a_k)$ is called an input command. A *state cycle* is a non-halting execution sequence of input commands such that the first and last input command in the sequence have the same state: $(q_k, a_k) \mapsto (q_{k+1}, a_{k+1}) \mapsto \ldots \mapsto (q_{N-1}, a_{N-1}) \mapsto (q_k, a_k)$. The length of this state cycle equals the number of input commands minus one. A state cycle is called a prime state cycle if it contains no proper state subcycles. For a prime state cycle, the length of the cycle equals the number of distinct states in the sequence. For example, $(2,0) \mapsto (3,1) \mapsto (4,0) \mapsto (2,1)$ is called a prime 3-state cycle because it has length 3 and also 3 distinct states $\{2,3,4\}$.

*Remark 22.1.* *Any prime state cycle has length* $\leq |Q|$

*Proof.* This follows from the pigeonhole principle and the definition of a prime state cycle.

*Remark 22.2.* *Any state cycle contains a prime state cycle*

*Proof.* Relabeling if necessary let $\zeta(q_1, q_1) = (q_1, a_1) \mapsto \ldots \mapsto (q_n, a_n) \mapsto (q_1, a_{n+1})$ be a state cycle. If $q_1$ is the only state visited twice, then the proof is completed. Otherwise, define $\ell = \min\{|\zeta(q_k, q_k)| : \zeta(q_k, q_k)$ is a subcycle of $\zeta(q_1, q_1)\}$. Then $\ell$ exists because $\zeta(q_1, q_1)$ is a subcycle of $\zeta(q_1, q_1)$. Claim: Any state cycle $\zeta(q_j, q_j)$ with $|\zeta(q_j, q_j)| = \ell$ must be a prime state cycle. Suppose not. Then there is a state $r \neq q_j$ that is visited twice in the state cycle $\zeta(q_j, q_j)$. But then $\zeta(q_r, q_r)$ is a cycle with length less than $\ell$ which contradicts $\ell'$s definition.

**Definition 22.7.** *Consecutive repeating state cycle*
If machine $(Q, A, \eta)$ during program execution repeats a state cycle two consecutive times, $(q_1, b_1) \mapsto \ldots \mapsto (q_n, b_n) \mapsto (q_1, b_1) \mapsto \ldots \mapsto (q_n, b_n) \mapsto (q_1, b_1)$, then $(Q, A, \eta)$ has a consecutive repeating state cycle.

**Theorem 22.1.** *Each periodic point determines a unique consecutive repeating state cycle*

*Proof.* Suppose $p$ is an immortal periodic point with period $n$. Let the input command sequence $(q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_{n+1},b_{n+1})$ denote the first $n$ input commands that are executed. Since $p$ has period $n$, $(q_1,b_1) = (q_{n+1},b_{n+1})$. Thus, the first $n$ steps are a state cycle $(q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_1,b_1)$. Since the $n+1$ computational step corresponds to applying $\eta$ to $p$, the window of execution is identical for the next $n$ steps as it was for the first $n$ steps. Thus, the next $n$ steps have an input command sequence that is identical as the first $n$ steps. Thus, the sequence of input commands for $2n$ steps is $(q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_1,b_1)$.

**Theorem 22.2.** *Each consecutive repeating state cycle determines a unique periodic point*

*Proof.* Suppose Turing machine $(Q,A,\eta)$ begins or resumes execution at some tape square and repeats a state cycle two consecutive times denoted as $(q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_1,b_1) \mapsto \ldots \mapsto (q_n,b_n) \mapsto (q_1,b_1)$. Let $s_k$ denote the tape square just before input command $(q_k,b_k)$ is executed the first time where $1 \leq k \leq n$. Let $t_k$ denote the tape square just before input command $(q_k,b_k)$ is executed the second time where $1 \leq k \leq n$.

Thus, just before input command $(q_1,b_1)$ is executed a second time, the window of execution for the first state cycle is $I_n = \cup_{j=1}^{n+1}\{s_j\}$ where $s_{n+1} = t_1$. The window of execution for the second repetition of the state cycle is $J_n = \cup_{j=1}^{n+1}\{t_j\}$, where $t_{n+1} = t_n + t_1 - s_n$.

Furthermore, observe that the window of execution for the computational steps 1 thru $k$ is $I_k = \cup_{j=1}^{k+1}\{s_j\}$ where the tape head is located at tape square $s_{k+1}$ after input command $(q_k,b_k)$ is executed the first time. Also, observe that the window of execution for the computational steps $n+1$ thru $n+k$ is $J_k = \cup_{j=1}^{k+1}\{t_j\}$ where the tape head is located at tape square $t_{k+1}$ after the input command $(q_k,b_k)$ is executed the second time in the second repeating cycle.

Next a useful notation represents the tape patterns for each computational step. Then the proof is completed using induction. Let $V_1$ denote the tape pattern, which is the sequence of alphabet symbols in the tape squares over the window of execution $I_n$, just before input command $(q_1,b_1)$ is executed the first time. Thus, $V_1(s_1) = b_1$. Let $V_k$ denote the tape pattern, which is the sequence of alphabet symbols in the tape squares over the window of execution $I_n$, just before input command $(q_k,b_k)$ is executed the first time. Thus, $V_k(s_k) = b_k$.

Let $W_1$ denote the tape pattern, which is the sequence of alphabet symbols in the tape squares over the window of execution $J_n$, just before input command $(q_1,b_1)$ is executed the second time. Thus, $W_1(t_1) = b_1$. Let $W_k$ denote the tape pattern, which is the sequence of alphabet symbols in the tape squares over the window of execution $J_n$, just before input command $(q_k,b_k)$ is executed the second time. Thus, $W_k(t_k) = b_k$.

Using induction, it is shown that $V_1$ on window of execution $I_n$ equals $W_1$ on window of execution $J_n$. This completes the proof.

**Base Case.** Since $(q_1,b_1)$ is the input command before computational step 1 and $(q_1,b_1)$ is the input command before computational step $n+1$, then $V_1(s_1) = b_1 = W_1(t_1)$. Thus, $V_1$ restricted to window of execution $I_1$ equals $W_1$ restricted to window of execution $J_1$.

$\eta(q_1,b_1) = (q_2,a_1,x)$ for some $a_1$ in $A$ and where $x = -1$ or $+1$.

Case $x = +1$. **Right Tape Head Move**

$$\begin{array}{cc} & s_1\ s_2 \\ V_1 \ldots & \underline{b_1}\ b_2 \ldots \\ V_2 \ldots & a_1\ \underline{b_2} \ldots \end{array} \qquad \begin{array}{cc} & t_1\ t_2 \\ W_1 \ldots & \underline{b_1}\ b_2 \ldots \\ W_2 \ldots & a_1\ \underline{b_2} \ldots \end{array}$$

Then $s_2 = s_1 + 1$ and $t_2 = t_1 + 1$ and $V_1(s_2) = b_2 = W_1(t_2)$. It has already been observed that $V_1(s_1) = b_1 = W_1(t_1)$. Thus, $V_1$ restricted to the window of execution $I_2$ equals $W_1$ restricted to the window of execution $J_2$ . Furthermore, the tape head is at $s_1$ just before computational step 1 and input command $(q_1,b_1)$ is executed; the tape head is at $t_1$ just before computational step $n+1$ and input command $(q_1,b_1)$ is executed. Also, $V_2(s_1) = a_1 = W_2(t_1)$ and $V_2(s_2) = b_2 = W_2(t_2)$. Thus, $V_2$ restricted to the window of execution $I_2$ equals $W_2$ restricted to the window of execution $J_2$. Furthermore, the tape head is at $s_2$ just before computational step 2 and input command $(q_2,b_2)$ is executed; the tape head is at $t_2$ just before computational step $n+2$ and input command $(q_2,b_2)$ is executed.

Case $x = -1$. **Left Tape Head Move**

$$\begin{array}{cc} & s_2\ s_1 \\ V_1 \ldots & b_2\ \underline{b_1} \ldots \\ V_2 \ldots & \underline{b_2}\ a_1 \ldots \end{array} \qquad \begin{array}{cc} & t_2\ t_1 \\ W_1 \ldots & b_2\ \underline{b_1} \ldots \\ W_2 \ldots & \underline{b_2}\ a_1 \ldots \end{array}$$

Then $s_2 = s_1 - 1$ and $t_2 = t_1 - 1$ and $V_1(s_2) = b_2 = W_1(t_2)$. And $V_1(s_1) = b_1 = W_1(t_1)$. Thus, $V_1$ restricted to the window of execution $I_2$ equals $W_1$ restricted to the window of execution $J_2$. Furthermore, the tape head is at $s_1$ just before computational step 1 and input command $(q_1,b_1)$ is executed; the tape head is at $t_1$ just before computational step $n+1$ and input command $(q_1,b_1)$ is executed. Also, $V_2(s_1) = a_1 = W_2(t_1)$ and $V_2(s_2) = b_2 = W_2(t_2)$. Thus, $V_2$ restricted to the window of execution $I_2$ equals $W_2$ restricted to the window of execution $J_2$. Furthermore, the tape head is at $s_2$ just before computational step 2 and input command $(q_2,b_2)$ is executed; the tape head is at $t_2$ just before computational step $n+2$ and input command $(q_2,b_2)$ is executed. This completes the base case of induction.

**Induction Hypothesis.** Suppose that for the $1,2,\ldots,k-1$ computational steps and the corresponding $n+1,n+2,\ldots,n+k-1$ steps that for every $i$ with $1 \leq i \leq k$:

1. $V_1$ restricted to the window of execution $I_i$ equals $W_1$ restricted to the window of execution $J_i$; and for each remaining $p$ where $p \leq i$, $V_p$ restricted to the window of execution $I_i$ equals $W_p$ restricted to the window of execution $J_i$.

2. Furthermore, the tape head is at $s_i$ just before computational step $i$ and input command $(q_i,b_i)$ is executed; the tape head is at $t_i$ just before step $n+i$ and input command $(q_i,b_i)$ is executed.

**Induction Step.** Since $(q_k,b_k)$ is the input command before computational step $k$ and before computational step $n+k$, then $V_k(s_k) = b_k = W_k(t_k)$.

$\eta(q_k,b_k) = (q_{k+1},a_k,x)$ for some $a_k$ in $A$ and $x = -1$ or $+1$.

Case $x = +1$. **Right Tape Head Move** for computational steps $k$ and $n+k$.

$$\begin{array}{lcl} & s_k\ s_{k+1} & \\ V_k & \ldots\ \underline{b_k}\ b_{k+1}\ \ldots & \\ V_{k+1} & \ldots\ a_k\ \underline{b_{k+1}}\ \ldots & \end{array} \qquad \begin{array}{lcl} & t_k\ t_{k+1} & \\ W_k & \ldots\ \underline{b_k}\ b_{k+1}\ \ldots & \\ W_{k+1} & \ldots\ a_k\ \underline{b_{k+1}}\ \ldots & \end{array}$$

By the inductive hypothesis $V_k$ restricted to window of execution $I_k$ equals $W_k$ restricted to window of execution $J_k$ and the only change to the tape and tape head after executing $\eta(q_k,b_k) = (q_{k+1},a_k,+1)$ for the steps $k$ and $n+k$ is that $V_{k+1}(s_k) = a_k = W_{k+1}(t_k)$ and $V_{k+1}(s_{k+1}) = b_{k+1} = W_{k+1}(t_{k+1})$ and that the tape heads move right to $s_{k+1}$ and $t_{k+1}$ respectively. Thus, $V_{k+1}$ restricted to window of execution $I_{k+1}$ equals $W_{k+1}$ restricted to window of execution $J_{k+1}$. And for each $j$ satisfying $1 \le j \le k$, then $V_j$ restricted to window of execution $I_{k+1}$ equals $W_j$ restricted to window of execution $J_{k+1}$.

Case $x = -1$. **Left Tape Head Move** for computational steps $k$ and $n+k$.

$$\begin{array}{lcl} & s_{k+1}\ s_k & \\ V_k & \ldots\ b_{k+1}\ \underline{b_k}\ \ldots & \\ V_{k+1} & \ldots\ \underline{b_{k+1}}\ a_k\ \ldots & \end{array} \qquad \begin{array}{lcl} & t_{k+1}\ t_k & \\ W_k & \ldots\ b_{k+1}\ \underline{b_k}\ \ldots & \\ W_{k+1} & \ldots\ \underline{b_{k+1}}\ a_k\ \ldots & \end{array}$$

By the inductive hypothesis $V_k$ restricted to window of execution $I_k$ equals $W_k$ restricted to window of execution $J_k$ and the only change to the tape and tape head after executing $\eta(q_k,b_k) = (q_{k+1},a_k,-1)$ for the steps $k$ and $n+k$ is that $V_{k+1}(s_k) = a_k = W_{k+1}(t_k)$ and $V_{k+1}(s_{k+1}) = b_{k+1} = W_{k+1}(t_{k+1})$ and that the tape heads move left to $s_{k+1}$ and $t_{k+1}$ respectively. Thus, $V_{k+1}$ restricted to window of execution $I_{k+1}$ equals $W_{k+1}$ restricted to window of execution $J_{k+1}$. And for each $j$ satisfying $1 \le j \le k$, then $V_j$ restricted to window of execution $I_{k+1}$ equals $W_j$ restricted to window of execution $J_{k+1}$.

## 22.4 Prime Directed Edge Sequences

Edge pattern substitution is the mathematical operation used to link match prime directed edges. Prime directed edges are link matched to construct prime directed edge sequences. The notion of an overlap match expresses how a part or all of one tape pattern may match part or all of another tape pattern.

**Definition 22.8.** *Overlap Matching & Intersection Patterns*
Let $V$ and $W$ be patterns. $(V,s)$ overlap matches $(W,t)$ if and only if $V(s+c) = W(t+c)$ for each integer $c$ satisfying $-\ell \le c \le \mu$ such that $\ell = \min\{s,t\}$ and $\mu = \min\{|V|-1-s,|W|-1-t\}$ where $0 \le s < |V|$ and $0 \le t < |W|$. The index

$s$ is called the head of pattern $V$ and the index $t$ is called the head of pattern $W$. If $V$ is also a subpattern, then $(V,s)$ submatches $(W,t)$. If $(V,s)$ overlap matches $(W,t)$, then define the intersection pattern $I$ with head $u=\ell$ as $(I,u)=(V,s)\cap(W,t)$, where $I(c)=V(c+s-\ell)$ for every integer $c$ satisfying $0\le c\le(\ell+\mu)$ where $\ell=\min\{s,t\}$ and $\mu=\min\{|V|-1-s,|W|-1-t\}$.

**Definition 22.9.** *Edge Pattern Substitution Operator*
Given patterns $V=v_0v_1\ldots v_n$ and $W=w_0w_1\ldots w_n$ with heads $s,t$ satisfying $0\le s,t\le n$ and pattern $P=p_0p_1\ldots p_m$ with head $u$ satisfying $0\le u\le m$. If $(P,u)$ overlap matches $(V,s)$, define the edge pattern substitution operator $\oplus$ as $E=(P,u)\oplus[(V,s)\Rightarrow(W,t)]$ according to the four different cases **A**, **B**, **C** and **D**.

**Case A** $u>s$ and $m-u>n-s$

$$E(k)=\begin{matrix}W(k+s-u) & \text{if } u-s\le k\le u+n-s\\ P(k) & \text{if } 0\le k<u-s \text{ or } u+n-s<k\le m\end{matrix}$$

The head of $E$ is $u+t-s$ and $|E|=m+1$.

$$\begin{matrix}p_0\ p_1\ \ldots\ p_{u-s}\ \ldots\ \underline{p_u}\ \ldots\ p_{u+n-s}\ \ldots\ p_m\\ \qquad v_0\quad \ldots\ \underline{v_s}\ \ldots\ v_n\\ \qquad w_0\quad \ldots\ w_s\ \ldots\ w_n\end{matrix}$$

**Case B** $u>s$ and $m-u\le n-s$

$$E(k)=\begin{matrix}W(k+s-u) & \text{if } u-s\le k\le n+u-s\\ P(k) & \text{if } 0\le k<u-s\end{matrix}$$

The head of $E$ is $u+t-s$ and $|E|=n+u-s+1$.

$$\begin{matrix}p_0\ p_1\ \ldots\ p_{u-s}\ \ldots\ \underline{p_u}\ \ldots\ p_m\\ \qquad v_0\quad \ldots\ \underline{v_s}\ \ldots\ v_{s+m-u}\ \ldots\ v_n\\ \qquad w_0\quad \ldots\ w_s\ \ldots\ w_{s+m-u}\ \ldots\ w_n\end{matrix}$$

**Case C** $u\le s$ and $m-u\le n-s$

$E(k)=W(k)$ when $0\le k\le n$. The head of $E$ is $t$ and $|E|=|W|=n+1$.

$$\begin{matrix}\qquad p_0\quad \ldots\ \underline{p_u}\ \ldots\ p_m\\ v_0\ \ldots\ v_{s-u}\ \ldots\ \underline{v_s}\ \ldots\ v_{s+m-u}\ \ldots\ v_n\\ w_0\ \ldots\ w_{s-u}\ \ldots\ w_s\ \ldots\ w_{s+m-u}\ \ldots\ w_n\end{matrix}$$

**Case D** $u\le s$ and $m-u>n-s$

$$E(k)=\begin{matrix}P(k+u-s) & \text{if } n<k\le m+s-u\\ W(k) & \text{if } 0\le k\le n\end{matrix}$$

The head of $E$ is $t$ and $|E|=m+s-u+1$.

$$\begin{matrix}\qquad p_0\quad \ldots\ \underline{p_u}\ \ldots\ p_{u+n-s}\ \ldots\ p_m\\ v_0\ \ldots\ v_{s-u}\ \ldots\ \underline{v_s}\ \ldots\ v_n\\ w_0\ \ldots\ w_{s-u}\ \ldots\ w_s\ \ldots\ w_n\end{matrix}$$

Set pattern $P=0101\ 110$. Set pattern $V=11\ \underline{0}101$. Set pattern $W=01\ 00\underline{1}0$. Then $(P,0)$ *overlap matches* $(V,2)$. Thus, the *edge pattern substitution operation*

is well-defined so $E = (P,0) \oplus [(V,2) \Rightarrow (W,4)] = 01\ 00\underline{1}0\ 110$. The head or index of pattern $E = 4$. Furthermore, $(P,4)$ overlap matches $(V,0)$. Thus, $F = (P,4) \oplus [(V,0) \Rightarrow (W,4)] = 0101\ 0100\underline{1}0$. The index of pattern $F = u + t - s = 4 + 4 - 0 = 8$.

**Definition 22.10.** *Execution node for* $(Q,A,\eta)$

An execution node is a triplet $\mathfrak{N} = [q,w_0w_1\ldots w_n,t]$ for some state $q$ in $Q$ where $w_0w_1\ldots w_n$ is a pattern of $n+1$ alphabet symbols each in $A$ such that $t$ is a non-negative integer satisfying $0 \le t \le n$. Intuitively, $w_0w_1\ldots w_n$ is the pattern of alphabet symbols on $n+1$ consecutive tape squares and $t$ represents the location of the tape head.

**Definition 22.11.** *Halting Execution Node*

Suppose $[q,v_0v_1\ldots v_n,s]$ is an execution node and over the next $|Q|$ computational steps a prime state cycle is not found. In other words, a prime directed edge is not generated. Then the Turing machine execution halted in $|Q|$ or less steps. Let $W$ be a pattern such that $(W,t)$ submatches $(V,s)$ and $W$ spans the window of execution until execution halts. Define the halting node as $H = [q,W,t]$.

**Definition 22.12.** *Prime directed edge*

A prime head execution node $\mathfrak{H} = [q,v_0v_1\ldots v_n,s]$ and prime tail execution node $\mathfrak{T} = [r,w_0w_1\ldots w_n,t]$ are called a prime directed edge iff all of the following hold:

1. When Turing machine $(Q,A,\eta)$ starts execution, it is in state $q$ and the tape head is located at tape square $s$. For each $j$ satisfying $0 \le j \le n$ tape square $j$ contains symbol $v_j$. In other words, the initial tape pattern is $v_0v_1\ldots\underline{v_s}\ldots v_n$.
2. During the next $N$ computational steps, state $r$ is visited twice and all other states in $Q$ are visited at most once. In other words, the corresponding sequence of input commands during the $N$ computational steps executed contains only one prime state cycle.
3. After $N$ computational steps, where $1 \le N \le |Q|$, the machine is in state $r$. The tape head is located at tape square $t$. For each $j$ satisfying $0 \le j \le n$ tape square $j$ contains symbol $w_j$. The tape pattern after the $N$ computational steps is $w_0w_1\ldots\underline{w_t}\ldots w_n$.
4. The window of execution for these $N$ computational steps is $[0,n]$.

A prime directed edge is denoted as $[q,v_0v_1\ldots v_n,s] \Rightarrow [r,w_0w_1\ldots w_n,t]$ or $\mathfrak{H} \Rightarrow \mathfrak{T}$. The number of computational steps $N$ is denoted as $|\mathfrak{H} \Rightarrow \mathfrak{T}|$.

**Definition 22.13.** *Overlap matching of a node to a prime head node*

Execution node $\mathfrak{N}$ overlap matches head execution node $\mathfrak{H}$ iff the following hold:

1. $\mathfrak{N} = [r,w_0w_1\ldots w_n,t]$ is an execution node satisfying $0 \le t \le n$.
2. $\mathfrak{H} = [q,v_0v_1\ldots v_m,s]$ is a prime head node satisfying $0 \le s \le m$.
3. State $q =$ State $r$.
4. Pattern $(W,t)$ overlap matches $(V,s)$, where $W = w_0w_1\ldots w_n$ and $V = v_0v_1\ldots v_m$.

**Lemma 22.1.** *Overlap matching prime head nodes are equal*

If $\mathfrak{H}_j = [q,P,u]$ and $\mathfrak{H}_k = [q,V,s]$ are prime head nodes and they overlap match, then they are equal. Distinct prime directed edges have prime head nodes that do not overlap match.

*Proof.* From the definition of prime edge, $0 \leq u \leq |P|$ and $0 \leq s \leq |V|$. Let $(I,m) = (P,u) \cap (V,s)$ where $m = \min\{s,u\}$. Suppose the same machine begins execution on tape $I$ with tape head at $m$ in state $q$. If $s = u$ and $|P| = |V|$, then the proof is complete. Otherwise, $s \neq u$ or $|P| \neq |V|$ or both. $\mathfrak{H}_j$ has a window of execution $[0,|P|-1]$ and $\mathfrak{H}_k$ has window of execution $[0,|V|-1]$. Let the $i$th step be the first time that the tape head exits finite tape $I$. This means the machine would execute the same machine instructions with respect to $\mathfrak{H}_j$ and $\mathfrak{H}_k$ up to the $i$th step, so on the $i$th step, $\mathfrak{H}_j$ and $\mathfrak{H}_k$ must execute the same instruction. Since it exits tape I at the $i$th step, this would imply that either pattern $P$ or $V$ are exited at the $i$th step. This contradicts either that $[0,|P|-1]$ is the window of execution for $\mathfrak{H}_j$ or $[0,|V|-1]$ is the window of execution for $\mathfrak{H}_k$.

**Theorem 22.3.** *The number of prime directed edges is* $\leq |Q|^2|A|^{|Q|+1}$

*Proof.* From lemma 22.1, each prime head node determines a unique prime directed edge so an upper bound for head nodes provides an upper bound for the prime directed edges. Consider prime head node $[q,V,s]$. There are $|Q|$ choices for the state $q$. Any pattern that represents the window of execution has length $\leq |Q|+1$.

Furthermore, lemma 22.1 implies, for any pattern $P$ where $(V,s)$ submatches $(P,t)$, then the resultant pattern is the same since $V$ spans the window of execution. Thus, $|A|^{|Q|+1}$ is an upper bound for the number of different patterns $V$.

There are two choices for $s$ in a $|Q|+1$ length pattern because the maximum number of execution steps is $|Q|$ (i.e., the tape head move sequence is either $|Q|$ consecutive left tape head moves or $|Q|$ right tape head moves). Thus, $|Q|$ is an upper bound for the number of choices for $s$ unless $|Q| = 1$. The bound holds in the trivial case that $|Q| = 1$. Thus, there are at most $|Q|^2|A|^{|Q|+1}$ prime directed edges.

$|Q|^2|A|^{|Q|+1}$ is not a strict upper bound. Procedure 2 describes an algorithm for finding all the prime directed edges of a Turing machine, which also provides the number of prime directed edges.

**Definition 22.14.** *Edge Node Substitution Operator*
Let $\mathfrak{H} \Rightarrow \mathfrak{T}$ be a prime directed edge with prime head node $\mathfrak{H} = [q,v_0v_1\ldots v_n,s]$ and tail node $\mathfrak{T} = [r,w_0w_1\ldots w_n,t]$. If execution node $\mathfrak{N} = [q,p_0p_1\ldots p_m,u]$ overlap matches $\mathfrak{H}$, then the edge pattern substitution operator in definition 22.9 induces a new execution node $\mathfrak{N} \oplus (\mathfrak{H} \Rightarrow \mathfrak{T}) = [r,(P,u) \oplus [(V,s) \Rightarrow (W,t)],k]$ with head $k = u+t-s$ if $u > s$ and head $k = t$ if $u \leq s$ such that $0 \leq s,t \leq n$ and $0 \leq u \leq m$ and patterns $V = v_0v_1\ldots v_n$ and $W = w_0w_1\ldots w_n$ and $P = p_0p_1\ldots p_m$.

Let $\mathfrak{P} = \{\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1,\ldots,\mathfrak{H}_k \Rightarrow \mathfrak{T}_k,\ldots,\mathfrak{H}_N \Rightarrow \mathfrak{T}_N\}$ denote the finite set of prime directed edges for $(Q,A,\eta)$. The number of prime directed edges in $\mathfrak{P}$ is $|\mathfrak{P}|$ and is

called the *prime directed edge (PE) complexity* of $(Q, A, \eta)$. As defined in 22.14, the link matching step compares two tape patterns, one from execution node $\mathfrak{N}_k$ and the other from head node $\mathfrak{H}_{k+1}$. If the tape patterns overlap match and the state of $\mathfrak{N}_k$ equals the state of $\mathfrak{H}_{k+1}$, then the edge node substitution operation is well-defined and is used to glue prime directed edge $\mathfrak{H}_{k+1} \Rightarrow \mathfrak{T}_{k+1}$ to execution node $\mathfrak{N}_k$. In other words, $\mathfrak{N}_{k+1} = \mathfrak{N}_k \oplus (\mathfrak{H}_{k+1} \Rightarrow \mathfrak{T}_{k+1})$ is computed.

**Definition 22.15.** *Prime directed edge sequence and Link Matching*
A prime directed edge sequence is defined inductively. Each element is a coordinate pair, where the first element is a prime directed edge, the second element is an execution node and each element is expressed as $(\mathfrak{H}_k \Rightarrow \mathfrak{T}_k, \mathfrak{N}_k)$. The first element of a prime directed edge sequence is $(\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1, \mathfrak{N}_1)$ where $\mathfrak{N}_1 = \mathfrak{T}_1$, and $\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1$ is some prime directed edge in $\mathfrak{P}$. For simplicity in this definition, the indices in $\mathfrak{P}$ are relabeled if necessary so the first element has indices equal to 1. If $\mathfrak{N}_1$ overlap matches some non-halting prime head node $\mathfrak{H}_2$, the second element of the prime directed edge sequence is $(\mathfrak{H}_2 \Rightarrow \mathfrak{T}_2, \mathfrak{N}_2)$ where $\mathfrak{N}_2 = \mathfrak{N}_1 \oplus (\mathfrak{H}_2 \Rightarrow \mathfrak{T}_2)$. This is called a link match step. Otherwise, $\mathfrak{N}_1$ overlap matches a halting node, as defined in 22.11. In this case, the prime directed edge sequence terminates and this is called a halting match step. This is expressed as $[(\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1, \mathfrak{T}_1), \mathcal{H}]$.

If the first $k-1$ steps are link match steps, then the edge sequence up to the *kth* element is inductively defined as $[(\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1, \mathfrak{N}_1), (\mathfrak{H}_2 \Rightarrow \mathfrak{T}_2, \mathfrak{N}_2), \ldots,$
$(\mathfrak{H}_k \Rightarrow \mathfrak{T}_k, \mathfrak{N}_k)]$ where $\mathfrak{N}_j$ overlap matches prime head node $\mathfrak{H}_{j+1}$ and $\mathfrak{N}_{j+1} = \mathfrak{N}_j \oplus (\mathfrak{H}_{j+1} \Rightarrow \mathfrak{T}_{j+1})$ for each j satisfying $0 \leq j < k$.

## 22.5 Search Procedure for Periodic Points

This section demonstrates how to search for any periodic point – when they exist – by looking for consecutive repeating state cycles inside a prime directed edge sequence. This procedure is useful because it does not search over a tape pattern that has already been previously examined. This is due to Lemma 22.1 that proves if two head nodes overlap match in their respective prime directed edges, then the prime directed edges are equal.

Although there are other methods to look for periodic points, this search procedure demonstrates a broader approach because each aperiodic immortal configuration corresponds to a unique prime directed edge sequence. This is relevant because the non-trivial recurrence of the $M$-bounded aperiodic immortal configurations is integral to the undecidability of the Halting problem.

Link matching is a computational operation used to construct prime directed edge sequences. The following example link matches two prime directed edges.

*Example 22.3. Link matching prime directed edges*

State set $Q = \{q, r, s, t, u\}$. Alphabet set $A = \{0, 1\}$. Program $\eta$ is defined below.

$$\eta(q,0) = (r,1,+1) \qquad \eta(q,1) = (r,1,-1)$$
$$\eta(r,0) = (\mathcal{H},1,0) \qquad \eta(r,1) = (s,0,+1)$$
$$\eta(s,0) = (t,0,+1) \qquad \eta(s,1) = (\mathcal{H},1,0)$$
$$\eta(t,0) = (u,1,-1) \qquad \eta(t,1) = (q,0,+1)$$
$$\eta(u,0) = (q,1,-1) \qquad \eta(u,1) = (t,0,+1)$$

The execution steps of $[u,0\underline{0}010,1] \Rightarrow [q,1000\underline{0},4]$ are shown in table 1.

**Table 22.1** Prime Directed Edge $[u,0\underline{0}010,1] \Rightarrow [q,1000\underline{0},4]$

| STATE | TAPE | HEAD | COMMAND |
|---|---|---|---|
| $u$ | $0\underline{0}010$ | 1 | $\eta(u,0) = (q,1,-1)$ |
| $q$ | $\underline{0}1010$ | 0 | $\eta(q,0) = (r,1,+1)$ |
| $r$ | $1\underline{1}010$ | 1 | $\eta(r,1) = (s,0,+1)$ |
| $s$ | $10\underline{0}10$ | 2 | $\eta(s,0) = (t,0,+1)$ |
| $t$ | $100\underline{1}0$ | 3 | $\eta(t,1) = (q,0,+1)$ |
| $q$ | $1000\underline{0}$ | 4 | |

The execution steps of $[q,\underline{0}1010,0] \Rightarrow [q,1000\underline{0},4]$ are shown in table 2.

**Table 22.2** Prime Directed Edge $[q,\underline{0}1010,0] \Rightarrow [q,1000\underline{0},4]$

| STATE | TAPE | HEAD | COMMAND |
|---|---|---|---|
| $q$ | $\underline{0}1010$ | 0 | $\eta(q,0) = (r,1,+1)$ |
| $r$ | $1\underline{1}010$ | 1 | $\eta(r,1) = (s,0,+1)$ |
| $s$ | $10\underline{0}10$ | 2 | $\eta(s,0) = (t,0,+1)$ |
| $t$ | $100\underline{1}0$ | 3 | $\eta(t,1) = (q,0,+1)$ |
| $q$ | $1000\underline{0}$ | 4 | |

Prime edge $[u,0\underline{0}010,1] \Rightarrow [q,1000\underline{0},4]$ can be link matched to prime edge $[q,\underline{0}1010,0] \Rightarrow [q,1000\underline{0},4]$. After link matching, the sequence of input commands is $[(u,0),(q,0),\ (r,1),(s,0),(t,1),(q,0),(r,1),(s,0),(t,1)]$. Observe that $[(q,0),(r,1),\ (s,0),(t,1),\ (q,0),(r,1),\ (s,0),(t,1)]$ is a consecutive repeating state cycle, which corresponds to the periodic configuration $[q,\overline{1000\ \underline{0}101\ 0101}]$.

**Definition 22.16.** *Prime Input Command Sequence*

Suppose $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n)$ is an execution sequence of input commands for $(Q,A,\eta)$. Then $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n)$ is a prime input command sequence if $q_n$ is visited twice and all other states in this sequence are visited once. In other words, a prime input command sequence contains exactly one prime state cycle.

**Lemma 22.2.** *Prime Directed Edges $\Leftrightarrow$ Prime Input Command Sequences*

Prime directed edges and prime input command sequences are in one-to-one correspondence.

*Proof.* ($\Rightarrow$) Let $\mathfrak{H} \Rightarrow \mathfrak{T}$ be a prime directed edge where $\mathfrak{H} = [q, v_0 v_1 \ldots v_n, s]$ and $\mathfrak{T} = [r, w_0 w_1 \ldots w_n, t]$. From the definition of a prime directed edge, over the next $N$ computational steps some state $r$ is visited twice, all other states in $Q$ are visited at most once and there is a sequence of input commands $(q, v_s) \mapsto (q_1, a_1) \mapsto \ldots (r, a_k) \ldots \mapsto (r, w_t)$ corresponding to these $N$ steps. This is a prime input command sequence.

($\Leftarrow$) Let $(q_1, a_1) \mapsto \ldots \mapsto (r, a_{N+1})$ be a prime input command sequence with $N$ computational steps. Then $r$ is visited twice and all other states in the sequence are visited only once. Let $v_0 v_1 \ldots v_n$ be the initial tape pattern over the window of execution during the $N$ computational steps. Now $a_1 = v_s$ for some $s$. Let $w_0 w_1 \ldots w_n$ be the final tape pattern over the window of execution as a result of these $N$ steps. From the definition of the window of execution, the tape head is at some $t$ satisfying $0 \leq t \leq n$ after these $N$ steps. Thus, $[q, v_0 v_1 \ldots v_n, s] \Rightarrow [r, w_0 w_1 \ldots w_n, t]$ is a prime directed edge.

**Lemma 3.** *Any consecutive repeating state cycle is contained in a composition of one or more prime input command sequences.*

*Proof.* The proof is in the appendix.

*Example 22.4*

This example illustrates the correspondence between prime directed edges and prime input command sequences for machine $(Q, A, \eta)$ where $Q = \{2,3,4\}$, $\mathcal{H}$ is the halting state, $A = \{0,1\}$ and $\eta$ is specified as $\eta(2,0) = (3,1,-1)$, $\eta(2,1) = (4,0,-1)$, $\eta(3,0) = (4,1,+1)$, $\eta(3,1) = (4,0,+1)$, $\eta(4,0) = (\mathcal{H},0,0)$, and $\eta(4,1) = (2,0,+1)$. The correspondence is shown below.

| **Prime Directed Edges** | **Prime Input Command Sequences** |
|---|---|
| $[2,0\underline{0}0,1] \Rightarrow [2,10\underline{0},2]$ | $(2,0) \mapsto (3,0) \mapsto (4,1) \mapsto (2,0)$ |
| $[2,1\underline{0}0,1] \Rightarrow [2,00\underline{0},2]$ | $(2,0) \mapsto (3,1) \mapsto (4,1) \mapsto (2,0)$ |
| $[2,1\underline{1},1] \Rightarrow [2,0\underline{0},1]$ | $(2,1) \mapsto (4,1) \mapsto (2,0)$ |
| $[2,0\underline{0}1,1] \Rightarrow [2,10\underline{1},2]$ | $(2,0) \mapsto (3,0) \mapsto (4,1) \mapsto (2,1)$ |
| $[2,1\underline{0}1,1] \Rightarrow [2,00\underline{1},2]$ | $(2,0) \mapsto (3,1) \mapsto (4,1) \mapsto (2,1)$ |
| $[3,\underline{0}10,0] \Rightarrow [3,1\underline{0}1,1]$ | $(3,0) \mapsto (4,1) \mapsto (2,0) \mapsto (3,0)$ |
| $[3,\underline{1}10,0] \Rightarrow [3,0\underline{0}1,1]$ | $(3,1) \mapsto (4,1) \mapsto (2,0) \mapsto (3,0)$ |
| $[4,\underline{1}0,0] \Rightarrow [4,1\underline{1},1]$ | $(4,1) \mapsto (2,0) \mapsto (3,0) \mapsto (4,1)$ |
| $[4,\underline{1}1,0] \Rightarrow [4,0\underline{0},1]$ | $(4,1) \mapsto (2,1) \mapsto (4,0)$ |

There are 9 distinct prime directed edges. Observe that $|Q|^2 |A|^{|Q|+1} = 3^2 2^4 = 144$.

**Definition 22.17.** *Edge Sequence contains a consecutive repeating state cycle* Lemma 22.2 implies that an edge sequence corresponds to a sequence of prime input commands. The expression *an edge sequence contains a consecutive repeating state cycle* means that the corresponding sequence of prime input commands contains a consecutive repeating state cycle.

**Theorem 22.4.** *Any consecutive repeating state cycle is contained in a prime directed edge sequence.*

*Proof.* This follows immediately from definitions 22.15, 22.17 and lemmas 22.2 and 22.3.

**Procedure 1.** *Searching for a consecutive repeating state cycle in a prime directed edge sequence*

Given an edge sequence whose corresponding prime input command sequence $(q_0,a_0) \mapsto (q_1,a_1) \mapsto ... \mapsto (q_N,a_N)$ has length $N$.

Set $n = \frac{N}{2}$ if $N$ is even; otherwise, set $n = \frac{N+1}{2}$ if $N$ is odd.

For each $k$ in $\{0,1,2,\ldots,n-1\}$
For each $j$ in $\{0,1,\ldots,N-2k-1\}$
{
If $(q_j,a_j) \mapsto \ldots \mapsto (q_{j+k},a_{j+k})$ equals $(q_{j+k+1},a_{j+k+1}) \mapsto \ldots \mapsto (q_{j+2k+1},a_{j+2k+1})$
return consecutive repeating state cycle
$(q_j,a_j) \mapsto \ldots (q_{j+k},a_{j+k}) \ldots (q_{j+2k+1},a_{j+2k+1})$
}
If the outer for loop is exited without finding a consecutive repeating state cycle return NO consecutive repeating state cycles were found.

**Procedure 2.** *Prime Directed Edge Search Procedure*

Read Turing Machine $(Q,A,\eta)$ as input.
Set $\mathfrak{P} = \emptyset$.
For each non-halting state $q$ in $Q$
For each pattern $a_{-|Q|} \ldots a_{-2}a_{-1}a_0a_1a_2 \ldots a_{|Q|}$ selected from $A^{2|Q|+1}$
{

| Square | $-\lvert Q\rvert$ | $\ldots$ | $-1$ | 0 | 1 | $\ldots$ | $\lvert Q\rvert$ |
|---|---|---|---|---|---|---|---|
| Contents | $a_{-\lvert Q\rvert}$ | $\ldots$ | $a_{-1}$ | $\underline{a}_0$ | $a_1$ | $\ldots$ | $a_{\lvert Q\rvert}$ |

1. In start state $q$, $T(k) = a_k$ where $-|Q| \le k \le |Q|$, and tape head location 0, execute $(Q,A,\eta)$ until one state is visited twice or a halting state is reached. The execution takes $\le |Q|$ steps.
2. If execution does not halt, set $r$ equal to the state that is first visited twice.
3. Over this window of execution, construct a prime directed edge $\mathfrak{H} \Rightarrow \mathfrak{T}$ where $\mathfrak{H} = [q,v_0v_1 \ldots v_n,s]$, $\mathfrak{T} = [r,w_0w_1 \ldots w_n,t]$ and $0 \le s,t \le n \le |Q|$.
4. Set $\mathfrak{P} = \mathfrak{P} \cup \{\mathfrak{H} \Rightarrow \mathfrak{T}\}$.

}

*Remark 22.3. Prime Directed Edge Search Procedure finds all edges*
Procedure 2 finds all prime directed edges of $(Q,A,\eta)$ and all halting nodes.

*Proof.* Let $\mathfrak{H} \Rightarrow \mathfrak{T}$ be a prime directed edge of $(Q, A, \eta)$. Then $\mathfrak{H} \Rightarrow \mathfrak{T}$ has a head node $\mathfrak{H} = [q, v_0 v_1 ... v_n, s]$, for some state $q$ in $Q$, for some tape pattern $v_0 v_1 \ldots v_n$ that lies in $A^{n+1}$, such that $n \leq |Q|$ and $0 \leq s \leq n$. In the outer loop of procedure 2, when $q$ is selected from $Q$ and in the inner loop when the tape pattern $a_{-|Q|} \ldots a_{-2} a_{-1} a_0 a_1 a_2 \ldots a_{|Q|}$ is selected from $A^{2|Q|+1}$ such that $a_{-s} = v_0 \quad \ldots \quad a_{-k} = v_{s-k} \quad \ldots \quad a_0 = v_s \quad \ldots \quad a_k = v_{s+k} \quad \ldots \quad a_{n-s} = v_n$ then the machine execution in procedure 2 will construct prime directed edge $\mathfrak{H} \Rightarrow \mathfrak{T}$.

When the head node is a halting node, the machine execution must halt in at most $|Q|$ steps. Otherwise, it would visit a non-halting state twice and be a non-halting head node. The rest of the argument for this halting node is the same as for the non-halting head node.

To avoid subscripts of a subscript, let $p_j$ and the subscript $p(j)$ represent the same number. $\mathfrak{P} = \{\mathfrak{H}_1 \Rightarrow \mathfrak{T}_1, \ldots, \mathfrak{H}_k \Rightarrow \mathfrak{T}_k, \ldots, \mathfrak{H}_N \Rightarrow \mathfrak{T}_N\}$ is the set of all prime directed edges. $E([p_1], 1)$ is the edge sequence $[(\mathfrak{H}_{p(1)} \Rightarrow \mathfrak{T}_{p(1)}, \mathfrak{N}_{p(1)})]$ of length 1 where $\mathfrak{N}_{p(1)} = \mathfrak{T}_{p(1)}$ and $1 \leq p_1 \leq |\mathfrak{P}|$. Next $E([p_1, p_2], 2)$ is the edge sequence $[(\mathfrak{H}_{p(1)} \Rightarrow \mathfrak{T}_{p(1)}, \mathfrak{N}_{p(1)}), (\mathfrak{H}_{p(2)} \Rightarrow \mathfrak{T}_{p(2)}, \mathfrak{N}_{p(2)})]$ such that $1 \leq p_1, p_2 \leq |\mathfrak{P}|$ and $\mathfrak{N}_{p(2)} = \mathfrak{N}_{p(1)} \oplus (\mathfrak{H}_{p(2)} \Rightarrow \mathfrak{T}_{p(2)})$. In general,
$E([p_1, p_2, \ldots, p_k], k)$ denotes the edge sequence of length $k$ which is $[(\mathfrak{H}_{p(1)} \Rightarrow \mathfrak{T}_{p(1)}, \mathfrak{N}_{p(1)}), (\mathfrak{H}_{p(2)} \Rightarrow \mathfrak{T}_{p(2)}, \mathfrak{N}_{p(2)}), \ldots, (\mathfrak{H}_{p(k)} \Rightarrow \mathfrak{T}_{p(k)}, \mathfrak{N}_{p(k)})]$ where $\mathfrak{N}_{p(j+1)} = \mathfrak{N}_{p(j)} \oplus (\mathfrak{H}_{p(j+1)} \Rightarrow \mathfrak{T}_{p(j+1)})$ for each $j$ satisfying $1 \leq j \leq k-1$ and $1 \leq p(j) \leq |\mathfrak{P}|$.

**Procedure 3.** *Immortal Periodic Point Search Procedure*

Read Turing Machine $(Q, A, \eta)$ as input.
Use procedure 2 to find all prime directed edges $\mathfrak{P}$.
Set $k = 1$. Set $\mathfrak{E}(1) = \{E([1], 1), E([2], 1), \ldots, E([|\mathfrak{P}|], 1)\}$.
While ( $\mathfrak{E}(k) \neq \emptyset$ )
{
  Set $\mathfrak{E}(k+1) = \emptyset$.
  For each $E([p_1, p_2, \ldots, p_k], k)$ in $\mathfrak{E}(k)$
    For each prime directed edge $\mathfrak{H}_j \Rightarrow \mathfrak{T}_j$ in $\mathfrak{P}$
    {
      if $\mathfrak{H}_j \Rightarrow \mathfrak{T}_j$ link matches with $\mathfrak{N}_{p(k)}$ then
      {
        Set $p_{k+1} = j$.
        Set $\mathfrak{E}(k+1) = \mathfrak{E}(k+1) \cup E([p_1, p_2, \ldots, p_k, p_{k+1}], k+1)$.
        If $E([p_1, p_2, \ldots, p_k, p_{k+1}], k+1)$ has a consecutive repeating state cycle
        then return the consecutive repeating state cycle and exit the while loop.
      }
    }
  $k$ is incremented.
}

If the while loop exits because $\mathfrak{E}(m) = \emptyset$ for some $m$, then return $\emptyset$; in other words, $(Q, A, \eta)$ has only halting configurations.

*Remark 22.4.* $|\mathfrak{E}(k)| \leq |\mathfrak{P}|^k$

**Definition 22.18.** *Periodic Turing Machine*

A Turing machine $(Q, A, \eta)$ that has at least one periodic configuration, whenever it has an immortal configuration is said to be a *periodic* Turing machine.

*Remark 22.5.* If $E([p_1, p_2, \ldots, p_r], r)$ contains a consecutive repeating state cycle, then the corresponding periodic point has period $\leq \frac{1}{2} \sum_{k=1}^{r} |\mathfrak{H}_{p(k)} \Rightarrow \mathfrak{T}_{p(k)}|$.

*Proof.* Theorem 22.2 implies that a consecutive repeating state cycle determines an immortal periodic point. The length of the state cycle equals the period of the periodic point. Further, the number of input commands corresponding to the number of computational steps equals $|\mathfrak{H}_{p(k)} \Rightarrow \mathfrak{T}_{p(k)}|$ in directed edge $\mathfrak{H}_{p(k)} \Rightarrow \mathfrak{T}_{p(k)}$.

**Theorem 22.5.** *When machine $(Q, A, \eta)$ is periodic, procedure 3 terminates in a finite number of steps with either a consecutive repeating state cycle or for some positive integer $J$, then $\mathfrak{E}(J) = \emptyset$, which means all of the configurations of $(Q, A, \eta)$ are halting.*

*Proof.* If $(Q, A, \eta)$ has at least one configuration $(q, k, T)$ that is an immortal, then by definition 22.18, this implies the existence of a periodic point $p$ with some finite period $N$. Thus, from Theorem 22.1, there is a consecutive repeating state cycle that corresponds to the immortal periodic point $p$. Since procedure 3 searches through all possible prime edge sequences of length $k$, a consecutive repeating state cycle will be found that is contained in a prime directed edge sequence with length at most $2N$. Thus, periodic point $p$ of period $N$ will be reached before or while computing $\mathfrak{E}(2N)$.

Otherwise, $(Q, A, \eta)$ does not have any immortal configurations; in other words, for every configuration, $(Q, A, \eta)$ halts in a finite number of steps. Claim: There is a positive integer $J$ such that every edge sequence terminates while executing procedure 3. By reductio absurdum, suppose not. Then there is at least one infinite prime directed edge sequence that exists. This infinite edge sequence corresponds to an immortal configuration, which contradicts that $(Q, A, \eta)$ is a periodic machine.

*Example 22.5. A Turing Machine with only aperiodic immortal configurations*

This example is based on work in [2]. The state set $Q = \{a, b, c, d, e, f\}$ and the alphabet set $A = \{0, 1, 2, 3\}$. The halting state is $\mathcal{H}$. In the following table, the Turing program is presented as quintuples $(q, a, r, b, m)$ where $\eta(q, a) = (r, b, m)$.

$(a,0,d,1,+1)$ $(a,1,f,1,+1)$ $(a,2,f,2,+1)$ $(a,3,f,3,+1)$
$(b,0,c,1,-1)$ $(b,1,e,1,-1)$ $(b,2,e,2,-1)$ $(b,3,e,3,-1)$
$(c,0,a,2,-1)$ $(c,1,f,1,+1)$ $(c,2,f,2,+1)$ $(c,3,f,3,+1)$

$(d,0,b,2,+1)$ $(d,1,e,1,-1)$ $(d,2,e,2,-1)$ $(d,3,e,3,-1)$
$(e,0,a,3,-1)$ $(e,1,a,0,-1)$ $(e,2,d,0,+1)$ $(e,3,f,0,+1)$
$(f,0,b,3,+1)$ $(f,1,b,0,+1)$ $(f,2,c,0,-1)$ $(f,3,e,0,-1)$

Observe that any periodic tape pattern with any non-halting state is immortal. However, none of these configurations are periodic because none are returned to in a finite number of execution steps.

*Remark 22.6. Procedure 3 does not halt on aperiodic Turing machines.*

## 22.6 Discussion and Further Work

In [12], Turing presented the Halting problem and proved that the Halting problem is undecidable with a Turing Machine. In [8], Hooper proved that the Turing Immortality problem is undecidable. Both papers assume that every initial machine configuration is $M$-bounded for some finite $M$ (i.e., the tape is bounded by blank symbols before the Turing machine program begins executing). In [1], Berger demonstrated an aperiodic tiling that proved that the tiling problem was undecidable. In light of [2] and the results presented here, the aperiodicity appears to be an integral part of the undecidability.

Procedure 2 finds all the prime directed edges and works for any Turing machine. Furthermore, the construction of the edge sequences via link matching inside Procedure 3 works on any Turing machine; and at the $k$th pass through the outer loop, this construction explores all possible immortal configurations up to an edge sequence length of $k$ prime directed edges. The limitation of procedure 3 is on the aperiodic Turing machines and is due to the exit condition of finding a consecutive repeating state cycle.

Although the consecutive repeating state cycle characterizes any periodic configuration in the Turing machine, a broader notion of recurrence is needed to address the more complex behavior of aperiodic immortal configurations that are initially $M$-bounded. Lemma 22.1 implies that every immortal configuration is contained by a unique prime directed edge sequence, so prime directed edge sequences cover all possible Turing program behaviors. Research that further develops the mathematical notions described here could help better understand the aperiodic immortal configurations, aperiodic Turing machines and perhaps undecidability.

**Acknowledgements.** I would like to thank Tony Lauck, Michael Jones, Kyandoghere Kyamakya, Don Saari and Sandy Zabell for their helpful advice and discussions. I would like to thank Lutz Mueller for creating newLISP, which has been extremely useful in my research. I am extremely grateful for support from my wonderful wife, Joanne Gomez, without whom this work would not have been possible and Haley Arielle Fiske for her inspiration.

## References


1. Berger, R.: The undecidability of the domino problem. Memoirs of the American Mathematical Society (66) (1966)
2. Blondel, V., Cassaigne, J., Nichitiu, C.: On the presence of periodic configurations in Turing machines and in counter machines. Theoretical Computer Science 289(1), 573–590 (2002)
3. Bowen, R.: Periodic points and measures for Axiom A diffeomorphisms. Transactions for American Mathematical Society 154, 377–397 (1971)
4. Brouwer, L.E.J.: Uber Abbildung von Mannigfaltigkeiten. Mathematische Annalen 71, 97–115 (1912)
5. Davis, M.: Computability and Unsolvability. Dover Publications, New York (1982)
6. Fiske, M.S.: Non-autonomous dynamical systems applied to neural computation. PhD thesis, Northwestern University. UMI Microform 9714584 (1996)
7. Hopf, H.: Abbildungsklassen n-dimensionaler Mannigfaltigkeiten. Mathematische Annalen 96, 209–224 (1926)
8. Hooper, P.K.: The Undecidability of the Turing Machine Immortality Problem. Journal of Symbolic Logic 31(2) (1966)
9. Kürka, P.: On Topological Dynamics of Turing Machines. Theoretical Computer Science 174, 203–216 (1997)
10. Mueller, L.: newLISP Language (1999-2012), `http://www.newlisp.org/`
11. Poincare, H.: Sur les courbes definies par une equation differentielle. Oeuvres 1 (1892)
12. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. London Math. Soc. ser. 2 42(Parts 3 and 4), 230–265 (1936); [Turing, 1937a] A correction, ibid. 43, 544–546 (1937)


## Appendix

Using the same notation as Theorem 22.2, let $V_1$ denote the initial tape pattern, which is the sequence of alphabet symbols in the tape squares over the window of execution of the prime input command sequence, just before the first input command $(q_1,a_1)$ in the sequence is executed. Let $s_1$ denote the location of the tape head, $V_1(s_1) = a_1$. Let $V_k$ denote the tape pattern just before the kth input command $(q_k,a_k)$ in the sequence is executed and let $s_k$ denote the location of the tape head, $V_k(s_k) = a_k$.

**Definition 22.19.** *Composition of Prime Input Command Sequences*

Let $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n)$ and $(r_1,b_1) \mapsto \ldots \mapsto (r_m,b_m)$ be prime input command sequences where $V_k$ denotes the tape pattern just before the $k$th input command $(q_k,a_k)$ with tape head at $s_k$ with respect to $V_k$. $W_k$ denotes the tape pattern just before the $k$th input command $(r_k,b_k)$ with tape head at $t_k$ with respect to $W_k$. Suppose $(V_n,s_n)$ overlap matches with $(W_1,t_1)$ and $q_n = r_1$. Then $(q_n,a_n) = (r_1,b_1)$. The composition of these two prime input command sequences is defined as $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n) \mapsto (r_2,b_2) \mapsto \ldots \mapsto (r_m,b_m)$. The composition is undefined if $(V_n,s_n)$ and $(W_1,t_1)$ do not overlap match or $q_n \neq r_1$. If $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n) \mapsto (q_1,b_1)$ is a prime state cycle, then it is also prime input command sequence.

*Example 22.6.* Directed Partition procedure

Start with finite sequence ( 0 4 2 3 4 1 3 0 1 2 0 4 2 3 4 1 3 0 1 2 ).

The partition steps are shown below.

1. ( (0 4 2 3) 4 1 3 0 1 2 0 4 2 3 4 1 3 0 1 2). 4 lies in (0 4 2 3).
2. ( (0 4 2 3) (4 1 3 0) 1 2 0 4 2 3 4 1 3 0 1 2). 1 lies in (4 1 3 0).
3. ( (0 4 2 3) (4 1 3 0) (1 2 0 4) 2 3 4 1 3 0 1 2). 2 lies in (1 2 0 4).
4. ( (0 4 2 3) (4 1 3 0) (1 2 0 4) (2 3 4 1) 3 0 1 2). 3 lies in (2 3 4 1).
5. ( (0 4 2 3) (4 1 3 0) (1 2 0 4) (2 3 4 1) (3 0 1 2) ). 0 lies in (3 0 1 2).

**Definition 22.20.** *Tuples*

A tuple is a finite sequence of objects denoted as $(\sigma_1,\sigma_2,\ldots,\sigma_m)$. The length of the tuple is the number of objects in the sequence denoted as $|(\sigma_1,\sigma_2,\ldots,\sigma_m)| = m$. For our purposes, the objects of the tuple may be states, input commands or natural numbers. (3) is a tuple of length one. (1, 4, 5, 6) is a tuple of length four. Sometimes the commas will be omitted as in the previous example. (4 6 0 1 2 3) is a tuple of length six. The 4 is called the first object in tuple (4 6 0 1 2 3). 1 is called a member of tuple (4 6 0 1 2 3).

**Definition 22.21.** *Tuple of Tuples*

A tuple of tuples is of the form $(w_1,w_2,\ldots,w_n)$ where each $w_k$ may have a different length. An example of a tuple of tuples is ( (3), (1, 4, 5, 6), (4, 5, 6) ). The commas are omitted in this example ( (0 8 2 3) (1 7 5 7) (5 5 6) ).

**Definition 22.22.** *Directed Partition of a Sequence*

A directed partition is a tuple of tuples $(w_1,w_2,\ldots,w_n)$ satisfying Rule A and B.

Rule A. No object $\sigma$ occurs in any element tuple $w_k$ more than once.

Rule B. If $w_k$ and $w_{k+1}$ are consecutive tuples, then the first object in tuple $w_{k+1}$ is a member of tuple $w_k$.

*Example 22.7.* Directed Partition Examples

The five examples illustrate element and partition tuples and Rule A and Rule B.

( (0 8 2 3) (8 7 5 4) (5 0 6) ) is an example of a directed partition.

( (0 8 2 3) (8 7 5 4) (5 0 6) ) is sometimes called a partition tuple.

(0 8 2 3) is the first element tuple. The first object in this element tuple is 0.

Element tuple (8 0 5 7 0 3) violates Rule A because object 0 occurs twice.

( (0 8 2 3) (1 7 5 4) (5 0 6) ) violates Rule B since 1 does not lie in tuple (0 8 2 3).

**Definition 22.23.** *Consecutive Repeating Sequence and Extensions*

A consecutive repeating sequence is a sequence $(x_1, x_2, \ldots, x_n, \ldots, x_{2n})$ of length $2n$ for some positive integer $n$ such that $x_k = x_{n+k}$ for each $k$ satisfying $1 \leq k \leq n$. An extension sequence is the same consecutive repeating sequence for the first $2n$ elements $(x_1, x_2, \ldots, x_n, \ldots, x_{2n} \ldots x_{2n+m})$ such that $x_k = x_{2n+k}$ for each $k$ satisfying $1 \leq k \leq m$. A minimal extension sequence is an extension sequence $(x_1, \ldots, x_n, \ldots x_{2n+m})$ where $m$ is the minimum positive number such that there is one element in $x_{2n}, x_{2n+1}, \ldots, x_{2n+m}$ that occurs more than once. Thus, $x_{2n+k} = x_{2n+m}$ for some $k$ satisfying $0 \leq k < m$.

For example, the sequence $S = (4234130120\,4234130120)$ is a consecutive repeating sequence and $\overline{S} = (4234130120\,4234130120\,42341)$ is an extension sequence. $\overline{S}$ contains consecutive repeating sequence $S$ .

**Definition 22.24.** *Directed partition extension with last tuple satisfying Rule B*

Suppose $(x_1 \ldots x_n \ldots x_{2n}, x_{2n+1}, \ldots x_{2n+m})$ is an extension of consecutive repeating sequence $(x_1 \ldots\ x_n \ldots x_{2n})$. Then $(w_1, w_2, \ldots, w_r)$ is a directed partition extension if it is a directed partition of the extension: The last tuple $w_r$ satisfies Rule B if $x_{2n+m}$ is the last object in tuple $w_r$ and $x_{m+1}$ lies in tuple $w_r$.

For example, the extension $(4\,2\,3\,4\,1\,3\,0\,1\,2\,0\,4\,2\,3\,4\,1\,3\,0\,1\,2\,0\,4\,2\,3)$ has directed partition extension $((4\,2\,3)\ (4\,1\,3\,0)\ (1\,2\,0\,4)\ (2\,3\,4\,1)\ (3\,0\,1\,2)\ (0\,4\,2\,3))$ and the last tuple satisfies Rule B since 4 lies in $(0\,4\,2\,3)$.

**Procedure 4.** *Directed Partition procedure*

*This procedure converts a finite sequence into a directed partition.*

Given a finite sequence $(x_1 \ldots x_n)$ of objects.

Initialize element tuple $w_1$ to the empty tuple ( ).

Initialize partition tuple $P$ to the empty tuple ( ).

For each element $x_k$ in sequence $(x_1\ \ldots\ x_n)$

{

  if $x_k$ is a member of the current element tuple $w_r$

  {

    Append element tuple $w_r$ to the end of partition tuple $P$ so that $P = (w_1 \ldots w_r)$.

    Initialize current element tuple $w_{r+1} = (x_k)$.

  }

  else update $w_r$ by appending $x_k$ to end of element tuple $w_r$.

}

The result is the current partition tuple $P$ after element $x_n$ is examined in the loop. Observe that the tail of elements from $(x_1 \ldots x_n)$ with no repeated elements will not lie in the last element tuple of the final result $P$.

**Procedure 5.** *Directed Partition Procedure implemented in newLISP* <www.newlisp.org>

*The function findpartition converts any finite sequence represented as a list into a directed partition.*

```
(define (addobject  etuple object)
 (if (member object etuple)
      nil
      (append etuple  (list  object))
))

(define (findpartition seq)
  (let(
         (partition  '())
         (etuple  '())
         (testadd nil)
       )
    (dolist (object seq)
      (set 'testadd (addobject etuple object))
      (if testadd
         (set 'etuple testadd)
         (begin
           (set 'partition (append partition (list etuple)))
           (set 'etuple (list object))
         )
      ))
  partition
))
```

```
> (set 'seq '(4 2 3 4 1 3 0 1 2 0 4 2 3 4 1 3 0 1 2 0 4 2 3 4))
> (findpartition  seq)
((4 2 3) (4 1 3 0) (1 2 0 4) (2 3 4 1) (3 0 1 2) (0 4 2 3))
```

4 lies in the last tuple `(0 4 2 3)`.

*Remark 22.7.* Every Consecutive Repeating Sequence has an extension sequence with a directed partition such that the last tuple satisfies the Rule B property.

*Proof.* As defined in 22.23, extend consecutive repeating sequence $(x_1 \dots x_n \dots x_{2n})$ to the extension sequence $(x_1 \dots x_n \dots x_{2n}, x_{2n+1}, \dots x_{2n+m})$ such that $m$ is the minimum positive number such that there is one element in $x_{2n}, x_{2n+1}, \dots x_{2n+m}$ that occurs more than once. Thus, $x_{2n+k} = x_{2n+m}$ for some $k$ satisfying $0 \leq k < m$.

Apply procedure 4 to $\overline{S} = (x_1 \dots x_n \dots x_{2n}, x_{2n+1}, \dots x_{2n+m})$. Then the resulting partition tuple $P$ extends at least until element $x_{2n}$ and the last tuple in $P$ satisfies rule B. If the partition tuple $P$ is mapped back to the underlying sequence of elements, then it is an extension sequence since it reaches element $x_{2n}$.

**Lemma 22.3.** *Any consecutive repeating state cycle is contained in a composition of one or more prime input command sequences.*

*Proof.* Let $\sigma = [(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n)(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n)]$ be a consecutive repeating cycle. Procedure 4 and remark 22.7 show that this sequence of consecutive repeating input commands may be extended to a minimal extension sequence $[(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n) \mapsto (q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n) \mapsto (q_1,a_1) \mapsto \ldots \mapsto (q_m,a_m)]$.

For simplicity, let $\nu_k$ denote input command $(q_k,a_k)$. Apply procedure 4 to $(\nu_1 \ldots \nu_n \nu_1 \ldots \nu_n \nu_1 \ldots \nu_m)$ so that the result is the partition tuple $P = (w_1 \ldots w_r)$. Then the sequence of element tuples in $P$ represent a composition of one or more prime input command sequences.

Rules A and B imply that for consecutive tuples $w_k = (\nu_{k(1)} \nu_{k(2)} \ldots \nu_{k(m)})$ and $w_{k+1} = (\nu_{(k+1)(1)} \nu_{(k+1)(2)} \ldots \nu_{(k+1)(m)})$ , then $(q_{k(1)},a_{k(1)}) \mapsto \ldots \mapsto (q_{k(m)},a_{k(m)}) \mapsto (q_{(k+1)(1)},a_{(k+1)(1)})$ is a prime input command sequence. Remark 22.7 implies that the last tuple $w_r$ corresponds to a prime input command sequence and that the consecutive repeating state cycle is contained in the partition $P$ mapped back to the sequence of input commands.

**Definition 22.25.** *Finite sequence rotation*

Let $(x_0 x_1 \ldots x_n)$ be a finite sequence. A $k$-rotation is the resulting sequence $(x_k x_{k+1} \ldots\ x_n x_0\ x_1 \ldots x_{k-1})$. The 3-rotation of (8 7 3 4 5) is (3 4 5 8 7).

**Definition 22.26.** *Rotating a state-symbol cycle*

Let $(q_1,a_1) \mapsto \ldots \mapsto (q_n,a_n) \mapsto (q_1,b_1)$ be a state cycle. This state cycle is called a state-symbol cycle if $a_1 = b_1$. A rotation of this state-symbol cycle is the state cycle $(q_k,a_k) \mapsto \ldots, (q_n,a_n) \mapsto (q_1,a_1) \mapsto \ldots (q_k,b_k)$ for some $k$ satisfying $0 \leq k \leq n$. In this case, the state-symbol cycle has been rotated by $k-1$ steps.

**Lemma 22.4.** *Any consecutive repeating rotated state cycle generated from a consecutive repeating state cycle induces the same immortal periodic orbit.*

*Proof.* Let $p$ be the immortal periodic point induced by this consecutive repeating state cycle. Rotating this state cycle by $k$ steps corresponds to starting at periodic machine configuration $p$ and executing the Turing machine $k$ steps.

**Procedure 6.** *A newLISP [10] function that searches for a consecutive repeating sequence.*

```
(define  (findpatternrepeats   plength   seq)
  (let (
         (k 0)
         (maxk (- (length seq) (+ plength plength)) )
         (pattern nil)
         (repeatpair nil)
         (norepeats true)
       )

    (while (and (<= k maxk) norepeats)
       (set 'pattern (slice seq k plength))
```

```
          (if (= pattern (slice seq (+ k plength) plength))
              (begin
                (set 'repeatpair (list pattern k))
                (set 'norepeats false) )
          )
          (set 'k (+ k 1))
        )
        repeatpair
   ))

   (define (findrepeats seq)
     (let (
            (plength 1)
            (maxplength (/ (length seq) 2) )
            (repeatpair nil)
           )
        (while (and (<= plength maxplength) (not repeatpair))
            (set 'repeatpair (findpatternrepeats plength seq))
            (set 'plength (+ plength 1))
        )
        repeatpair
   ))

(set 'seq1 '(3 5 7 2 3 5 7 11 5 7 ) )
(set 'seq2 '(3 5 7 2 3 5 7 11 5 7 11 2 4 6 8 ) )
(set 'seq3 '(1 2 0 2 1 0 2 0 1 2 0 2 1 0 1 2 1 0 2 1 2 0
             2 1 0 1 2 0 2 1 2 0 1 2 1 0 1 2 0 1 0 1) )

>  (findrepeats seq1)
nil

>  (findrepeats seq2)
((5 7 11) 5)

>  (findrepeats seq3)
((0 1) 38)
```

# Author Index