

CHURCH-TURING THESIS: THE TURING IMMORTALITY PROBLEM
SOLVED WITH A DYNAMIC REGISTER MACHINE

DEFINITION 8.1 *Prime directed edge from head and tail execution nodes*

A *prime head execution node* $\mathcal{H} = [q, v_0 v_1 \dots v_n, s]$ and *prime tail execution node*

$\mathcal{T} = [r, w_0 w_1 \dots w_n, t]$ are called a *prime directed edge* iff all of the following hold:

- When Turing machine (Q, A, η) starts execution, it is in state q ; the tape head is located at tape square s . For each j satisfying $0 \leq j \leq n$ tape square j contains symbol v_j . In other words, the initial tape pattern is $v_0 v_1 \dots v_s \dots v_n$.
- 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.
- After N computational steps, where $1 \leq N \leq |Q|$, the machine is in state r . The tape head is located at tape square t . For each j satisfying $0 \leq j \leq n$ tape square j contains symbol w_j . The tape pattern after the N computational steps is $w_0 w_1 \dots w_t \dots w_n$.
- The window of execution for these N computational steps is $[0, n]$.

A prime directed edge is denoted as $\mathcal{H} \Rightarrow \mathcal{T}$ or $[q, v_0 v_1 \dots v_n, s] \Rightarrow [r, w_0 w_1 \dots w_n, t]$. The number of computational steps N is denoted as $|\mathcal{H} \Rightarrow \mathcal{T}|$.

DEFINITION 8.2 *Prime Input Command Sequence*

7.4 introduced input commands. If $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n)$ is an execution sequence of input commands for (Q, A, η) , then $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n)$ is a prime input command sequence if q_n is visited twice and all other states in the sequence are visited once. In other words, a *prime input command sequence* contains exactly one prime state cycle.

NOTATION 8.3 *Prime Input Command Sequence Notation*

Using the same notation as lemma 7.11, 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 – right before the first input command (q_1, a_1) in the sequence is executed. And let s_1 denote the location of the tape head i.e. $V_1(s_1) = a_1$. Let V_k denote the tape pattern right before the k th input command (q_k, a_k) in the sequence is executed and let s_k denote the location of the tape head i.e. $V_k(s_k) = a_k$.

DEFINITION 8.4 *Composition of Prime Input Command Sequences*

Let $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n)$ and $(r_1, b_1) \mapsto \dots \mapsto (r_m, b_m)$ be prime input command sequences where V_k denotes the tape pattern right before the k th input command (q_k, a_k) with tape head at s_k with respect to V_k and W_k denotes the tape pattern right 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 $(q_n, a_n) = (r_1, b_1)$.

And the composition of these two prime input command sequences is defined as

$$(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (r_2, b_2) \mapsto \dots \mapsto (r_m, b_m)$$

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 $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, b_1)$ is a prime state cycle, then it is also prime input command sequence. For simplicity in upcoming lemma 8.15, it is called a composition of one prime input command sequence.

The purpose of these next group of definitions is to show that any *consecutive repeating state cycle* is contained inside a *composition of prime input command sequences*. From lemmas 7.10 and 7.11, there is a one to one correspondence between a *consecutive repeating state cycle* and an *immortal periodic point*. If this *consecutive repeating state cycle* is rotated, then it is still part of the same periodic orbit of the original periodic point. Next it is shown that there is a one to one correspondence between *prime input command sequences* and *prime directed edges*. Subsequently, it is explained how to *link match prime directed edges*. Then it is demonstrated how to find all *prime directed edges* for a particular Turing machine. If a particular Turing machine has any immortal periodic points, then it will have corresponding *consecutive repeating state cycles* which will be contained in an *edge sequence of prime directed edges that are link matched*. Now for the details . . .

EXAMPLE 8.5 *Directed Partition Method*

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

Partition Steps

(0 4 2 3 4 1 3 0 1 2 0 4 2 3 4 1 3 0 1 2)	
((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). 1st element found.
(((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). 2nd element found.
(((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). 3rd element found.
(((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). 4th element found.
(((0 4 2 3) (4 1 3 0) (1 2 0 4) (2 3 4 1) (3 0 1 2))	0 lies in (0 4 2 3). 5th element found.

DEFINITION 8.6 *Tuples*

A *tuple* is a finite sequence of objects denoted as $(\sigma_1, \sigma_2, \dots, \sigma_m)$. The *length* of the tuple is the number of objects in the sequence denoted as $|(\sigma_1, \sigma_2, \dots, \sigma_m)| = m$. 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 8.7 *Tuple of Tuples*

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

DEFINITION 8.8 *Directed Partition of a Sequence*

A *directed partition* is a tuple of tuples (w_1, w_2, \dots, w_n) that satisfies Rules A and B.

Rule A. No object σ 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 8.9 *Directed Partition Examples*

$((0\ 8\ 2\ 3)\ (8\ 7\ 5\ 4)\ (5\ 0\ 6))$ 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. And 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 is not a member of *element tuple* $(0\ 8\ 2\ 3)$.

DEFINITION 8.10 *Consecutive Repeating Sequence and Extensions*

A *consecutive repeating sequence* is a sequence $(x_1, x_2, \dots, x_n, \dots, x_{2n})$ 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 $(x_1 \dots x_n \dots x_{2n} \dots x_{2n+m})$ 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, \dots, x_{2n+m}) where 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 $0 \leq k < m$.

For example, the sequence $S = (4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0\ 4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0)$ is a *consecutive repeating* sequence and $\bar{S} = (4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0\ 4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0\ 4\ 2\ 3\ 4\ 1)$ is an *extension* sequence. \bar{S} contains consecutive repeating sequence S .

DEFINITION 8.11 *Directed partition extension with last tuple satisfying Rule B*

Suppose $(x_1 \dots x_n \dots x_{2n}, x_{2n+1}, \dots, x_{2n+m})$ is an extension of *consecutive repeating* sequence $(x_1 \dots, x_n \dots x_{2n})$. Then (w_1, w_2, \dots, 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 $\bar{S} = (4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0\ 4\ 2\ 3\ 4\ 1\ 3\ 0\ 1\ 2\ 0\ 4\ 2\ 3)$ 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 $(0\ 4\ 2\ 3)$

METHOD 8.12 *Directed Partition Method*

Given a finite sequence $(x_1 \dots x_n)$ 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 $(x_1 \dots x_n)$
 {
 if x_k is a member of the current element tuple w_r
 {
 Append element tuple w_r to the end of partition tuple so that $P = (w_1 \dots w_r)$
 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 final result is the current partition tuple P after element x_n is examined in the loop.

Observe that the tail of elements from $(x_1 \dots x_n)$ with no repeated elements will not lie in the last element tuple of the final result P .

EXAMPLE 8.13 *Directed Partition Method implemented in newLISP, www.newlisp.org.*

```
(define (add_object element_tuple object)
  (if (member object element_tuple) nil
      (append element_tuple (list object)) ))

(define (find_partition seq)
  (let
    ( (partition_tuple '() )
      (element_tuple '() )
      (test_add nil)
    )
    (dolist (object seq)
      (set 'test_add (add_object element_tuple object) )
      (if test_add
          (set 'element_tuple test_add)
          (begin
              (set 'partition_tuple (append partition_tuple (list element_tuple) ) )
              (set 'element_tuple (list object) )
            )
          )
    )
    partition_tuple
  ))

(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 ) )

> (find_partition 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)
```

LEMMA 8.14 *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 **8.10**, extend consecutive repeating sequence $(x_1, x_2 \dots x_{2n})$ to the extension sequence $(x_1, x_2 \dots x_{2n}, x_{2n+1} \dots x_{2n+m})$ 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 method **8.12** to $\bar{S} = (x_1, x_2 \dots x_{2n}, x_{2n+1} \dots x_{2n+m})$. 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 8.15 *Any consecutive repeating state cycle is contained in a composition of one or more prime input command sequences.*

PROOF. Let $\sigma = [(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_n, a_n)]$ be a consecutive repeating cycle. Method **8.12** and lemma **8.14** show that this sequence of consecutive repeating input commands may be extended to a minimal extension sequence :

$$[(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_m, a_m)]$$

For simplicity, let v_k denote input command (q_k, a_k) .

Apply method **8.12** to $(v_1, \dots v_n v_1 \dots v_n v_1 \dots v_m)$ so that the result is the partition tuple $P = (w_1, \dots w_r)$. 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

$w_k = (v_{k(1)} v_{k(2)} \dots v_{k(m)})$ and $w_{k+1} = (v_{(k+1)(1)} v_{(k+1)(2)} \dots v_{(k+1)(m)})$, then $(q_{k(1)}, a_{k(1)}) \mapsto (q_{k(2)}, a_{k(2)}) \mapsto \dots \mapsto (q_{k(m)}, a_{k(m)}) \mapsto (q_{(k+1)(1)}, a_{(k+1)(1)})$ is a prime input command sequence. And 8.14 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 8.16 *Finite sequence rotation*

Let $(x_0 x_1 \dots x_n)$ be a finite sequence. A k -rotation is the resulting sequence $(x_k x_{k+1} \dots x_n x_0 x_1 \dots x_{k-1})$. The 3-rotation of $(8 7 3 4 5)$ is $(3 4 5 8 7)$. When it does matter how many elements it has been rotated, it is called a sequence rotation.

DEFINITION 8.17 *Rotating a state-symbol cycle*

Let $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, b_1)$ 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 $(q_k, a_k) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_k, a_k)$ for some k satisfying $0 \leq k \leq n$.

In this case, the *state-symbol cycle* has been *rotated* by $k-1$ steps.

LEMMA 8.18 *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 iterating p by the next k corresponding affine functions.

LEMMA 8.19 *Prime Directed Edges \Leftrightarrow Prime Input Command Sequences*

Prime directed edges and prime input command sequences are in 1 to 1 correspondence.

PROOF. (\Rightarrow) Let $\mathcal{A} \Rightarrow \mathcal{B}$ be a prime directed edge where $\mathcal{A} = [q, v_0 v_1 \dots v_n, s]$ and $\mathcal{B} = [r, w_0 w_1 \dots w_n, t]$. 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 $(q, v_s) \mapsto (q_1, a_1) \mapsto \dots (r, a_k) \dots \mapsto (r, w_t)$ corresponding to these N steps. This is a prime input command sequence.

(\Leftarrow) Let $(q_1, a_1) \mapsto \dots \mapsto (q_n, a_n)$ be a prime input command sequence with N computational steps. Then q_n is visited twice and all other states in the sequence are visited only once. Let $v_0 v_1 \dots v_n$ be the initial tape pattern over the window of execution during the N computational steps. Now $a_1 = v_s$ for some s . Let $w_0 w_1 \dots w_n$ be the final tape pattern over the window of execution as a result of these N steps. Then $a_n = v_t$ for some t . Thus, $[q, v_0 v_1 \dots v_n, s] \Rightarrow [r, w_0 w_1 \dots w_n, t]$ is a prime directed edge.

REMARK 8.20 *Upper bound for the number of prime directed edges*

Each prime head node determines a unique prime directed edge so an upper bound for head nodes provides an upper bound for the number of distinct 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, by the previous remark any pattern P such that (V, s) submatches (P, t) for some 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 .

Lastly, 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 $L^{|Q|}$ or $R^{|Q|}$. Thus, $|Q|$ is an upper bound for the number of choices for s unless $|Q|=1$. The following bound works in the trivial case that $|Q|=1$. Thus, there are at most $|Q|^2 |A|^{|Q|+1}$ prime directed edges.

EXAMPLE 8.21 *3-state machine prime directed edges and prime input command sequences*

Consider Turing Machine (Q, A, η) . $Q = \{2, 3, 4\}$ and 1 is the halting state. $A = \{0, 1\}$ and η is specified in the following table.

q	T_k	$\eta(q, T_k)$
2	0	(3, 1, L)
2	1	(4, 0, L)
3	0	(4, 1, R)
3	1	(4, 0, R)
4	0	(1, 0, R)
4	1	(2, 0, R)

Prime Directed Edges

- [2, 000, 1] \Rightarrow [2, 100, 2]
- [2, 100, 1] \Rightarrow [2, 000, 2]
- [2, 11, 1] \Rightarrow [2, 00, 1]
- [2, 001, 1] \Rightarrow [2, 101, 2]
- [2, 101, 1] \Rightarrow [2, 001, 2]

- [3, 010, 0] \Rightarrow [3, 101, 1]
- [3, 110, 0] \Rightarrow [3, 001, 1]

- [4, 10, 0] \Rightarrow [4, 11, 1]
- [4, 11, 0] \Rightarrow [4, 00, 1]

Prime Input Command Sequences

- (2, 0) \mapsto (3, 0) \mapsto (4, 1) \mapsto (2, 0)
- (2, 0) \mapsto (3, 1) \mapsto (4, 1) \mapsto (2, 0)
- (2, 1) \mapsto (4, 1) \mapsto (2, 0)
- (2, 0) \mapsto (3, 0) \mapsto (4, 1) \mapsto (2, 1)
- (2, 0) \mapsto (3, 1) \mapsto (4, 1) \mapsto (2, 1)

- (3, 0) \mapsto (4, 1) \mapsto (2, 0) \mapsto (3, 0)
- (3, 1) \mapsto (4, 1) \mapsto (2, 0) \mapsto (3, 0)

- (4, 1) \mapsto (2, 0) \mapsto (3, 0) \mapsto (4, 1)
- (4, 1) \mapsto (2, 1) \mapsto (4, 0)

There are 9 distinct prime state cycles. Observe that $|Q|^2 |A|^{|Q|+1} = 3^2(4^2) = 144$.

Observe that $|Q|(|A| + |A|^2) = 2(2+4) = 12$.

The upper bound in **8.20** appears to not be sharp. Although sharp upper bounds for the number of prime directed edges are important, these types of results are not addressed here.

In what follows *prime directed edges* are *link matched* so that for a given Turing Machine a *method for finding* consecutive repeating state cycles is demonstrated. It is proved that this method will find immortal periodic points if they exist. The expression *demonstrate a method for finding* is used instead of *describe an algorithm* in order to not create confusion with the current notion of a Turing algorithm computed by a fixed Turing machine. In section 9, an *dynamic register machine implements this new method of computing*. Then an *active element machine* is defined which can execute a *dynamic register machine*.

DEFINITION 8.22 *Halting Execution Node*

Suppose $[q, v_0 v_1 \dots v_n, s]$ 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 $\mathcal{H} = [q, W, t]$.

NOTATION 8.23 *Set of all prime directed edges*

Remark **8.20** provides an upper bound on the number of prime directed edges. Let

$\mathcal{P} = \{\mathcal{H}_1 \Rightarrow \mathcal{T}_1, \dots, \mathcal{H}_k \Rightarrow \mathcal{T}_k, \dots, \mathcal{H}_N \Rightarrow \mathcal{T}_N\}$ denote the finite set of prime directed edges for machine (Q, A, η) .

DEFINITION 8.24 *Overlap matching of a node to a prime head node*

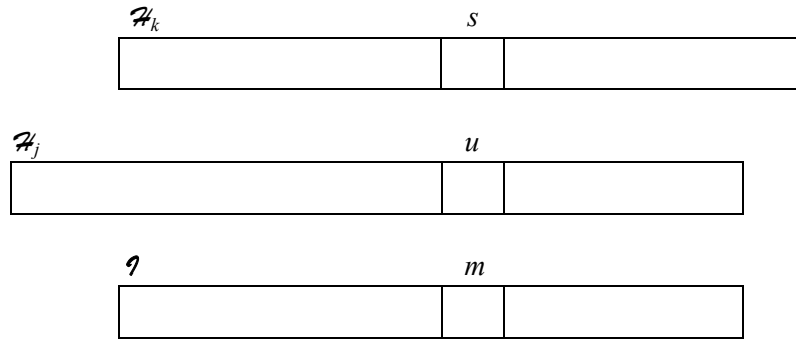
Execution node \mathcal{N} overlap matches head execution node \mathcal{H} iff the following hold:

- $\mathcal{N} = [r, w_0 w_1 \dots w_n, t]$ is an execution node satisfying $0 \leq t \leq n$
- $\mathcal{H} = [q, v_0 v_1 \dots v_m, s]$ is a prime head node satisfying $0 \leq s \leq m$
- State $q = \text{State } r$.
- W denotes pattern $w_0 w_1 \dots w_n$ and V denotes pattern $v_0 v_1 \dots v_m$
- Pattern (W, t) overlap matches (V, s) as defined in definition 7.1.

LEMMA 8.25 *Overlap matching prime head nodes are equal*

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

PROOF.



$$0 \leq u \leq |\mathcal{H}_j| \quad \text{and} \quad 0 \leq s \leq |\mathcal{H}_k|.$$

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 $|\mathcal{H}_j| = |\mathcal{H}_k|$, then the proof is complete.

Otherwise, $s \neq u$ or $|\mathcal{H}_j| \neq |\mathcal{H}_k|$ or both. \mathcal{H}_j has a window of execution $[0, |\mathcal{H}_j| - 1]$ and \mathcal{H}_k has window of execution $[0, |\mathcal{H}_k| - 1]$. 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 \mathcal{H}_j

and \mathcal{N}_k up to the i th step, so on the i th step, \mathcal{N}_j and \mathcal{N}_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 $[0, |\mathcal{N}_j| - 1]$ is the window of execution for \mathcal{N}_j or $[0, |\mathcal{N}_k| - 1]$ is the window of execution for \mathcal{N}_k .

DEFINITION 8.26 *Edge Node Substitution Operator* $\mathcal{N} \oplus (\mathcal{H} \Rightarrow \mathcal{T})$

Let $\mathcal{H} \Rightarrow \mathcal{T}$ be a prime directed edge with prime head node $\mathcal{H} = [q, v_0 v_1 \dots v_n, s]$ and tail node $\mathcal{T} = [r, w_0 w_1 \dots w_n, t]$. If execution node $\mathcal{N} = [q, p_0 p_1 \dots p_m, u]$ overlap matches \mathcal{H} , then the edge pattern substitution operator from 7.2 induces a new execution node $\mathcal{N} \oplus (\mathcal{H} \Rightarrow \mathcal{T}) = [r, (P, u) \oplus [(V, s) \Rightarrow (W, t)], k]$ 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 $V = v_0 v_1 \dots v_n$ and $W = w_0 w_1 \dots w_n$ and $P = p_0 p_1 \dots p_m$.

DEFINITION 8.27 *Prime directed edge sequence and Link Matching*

A *prime directed edge sequence* is defined inductively. Each element is a coordinate pair with the first element being a prime directed edge and the second element is an execution node. Each element is abstractly expressed as $(\mathcal{H}_k \Rightarrow \mathcal{T}_k, \mathcal{N}_k)$.

The first element of a *prime directed edge sequence* is $(\mathcal{H}_1 \Rightarrow \mathcal{T}_1, \mathcal{N}_1)$ where $\mathcal{N}_1 = \mathcal{T}_1$, and $\mathcal{H}_1 \Rightarrow \mathcal{T}_1$ is some prime directed edge in \mathcal{P} . For simplicity in this definition, the indices in \mathcal{P} are relabeled if necessary so the first element has indices equal to 1.

If \mathcal{N}_1 overlap matches some non-halting prime head node \mathcal{H}_2 , the second element of the prime directed edge sequence is $(\mathcal{H}_2 \Rightarrow \mathcal{T}_2, \mathcal{N}_2)$ where $\mathcal{N}_2 = \mathcal{N}_1 \oplus (\mathcal{H}_2 \Rightarrow \mathcal{T}_2)$. This is called a *link match* step.

Otherwise, \mathcal{N}_1 overlap matches a halting node, then the prime directed edge sequence terminates.

This is expressed as $[(\mathcal{A}_1 \Rightarrow \mathcal{T}_1, \mathcal{T}_1), \text{HALT}]$. In this case it is called a *halting match* step.

If the first $k - 1$ steps are *link match* steps, then the prime directed edge sequence is denoted as $[(\mathcal{A}_1 \Rightarrow \mathcal{T}_1, \mathcal{N}_1), (\mathcal{A}_2 \Rightarrow \mathcal{T}_2, \mathcal{N}_2), \dots, (\mathcal{A}_k \Rightarrow \mathcal{T}_k, \mathcal{N}_k)]$ where \mathcal{N}_j overlap matches prime head node \mathcal{A}_{j+1} and $\mathcal{N}_{j+1} = \mathcal{N}_j \oplus (\mathcal{A}_{j+1} \Rightarrow \mathcal{T}_{j+1})$ for each j satisfying $0 \leq j < k$.

NOTATION 8.28 *Edge Sequence Notation* $E([\rho_1, \rho_2, \dots, \rho_k], k)$

To avoid subscripts of a subscript, ρ_j and the subscript $_{p(j)}$ represent the same number. As defined in 8.27, $\mathcal{P} = \{\mathcal{A}_1 \Rightarrow \mathcal{T}_1, \dots, \mathcal{A}_k \Rightarrow \mathcal{T}_k, \dots, \mathcal{A}_N \Rightarrow \mathcal{T}_N\}$ denotes the set of all prime directed edges. $E([\rho_1], 1)$ denotes the edge sequence $[(\mathcal{A}_{p(1)} \Rightarrow \mathcal{T}_{p(1)}, \mathcal{N}_{p(1)})]$ of length 1 where $\mathcal{N}_{p(1)} = \mathcal{T}_{p(1)}$ and $1 \leq \rho_1 \leq |\mathcal{P}|$. Next $E([\rho_1, \rho_2], 2)$ denotes the edge sequence $[(\mathcal{A}_{p(1)} \Rightarrow \mathcal{T}_{p(1)}, \mathcal{N}_{p(1)}), (\mathcal{A}_{p(2)} \Rightarrow \mathcal{T}_{p(2)}, \mathcal{N}_{p(2)})]$ of length 2 where $\mathcal{N}_{p(2)} = \mathcal{N}_{p(1)} \oplus (\mathcal{A}_{p(2)} \Rightarrow \mathcal{T}_{p(2)})$ and $1 \leq \rho_1, \rho_2 \leq |\mathcal{P}|$.

In general, $E([\rho_1, \rho_2, \dots, \rho_k], k)$ denotes the edge sequence of length k which is explicitly $[(\mathcal{A}_{p(1)} \Rightarrow \mathcal{T}_{p(1)}, \mathcal{N}_{p(1)}), (\mathcal{A}_{p(2)} \Rightarrow \mathcal{T}_{p(2)}, \mathcal{N}_{p(2)}), \dots, (\mathcal{A}_{p(k)} \Rightarrow \mathcal{T}_{p(k)}, \mathcal{N}_{p(k)})]$ where $\mathcal{N}_{p(j+1)} = \mathcal{N}_{p(j)} \oplus (\mathcal{A}_{p(j+1)} \Rightarrow \mathcal{T}_{p(j+1)})$ for each j satisfying $1 \leq j \leq k - 1$ and $1 \leq p(j) \leq |\mathcal{P}|$.

DEFINITION 8.29 *Edge Sequence contains a consecutive repeating state cycle*

Lemma 8.19 implies that an edge sequence corresponds to a composition of prime input commands. The expression *an edge sequence contains a consecutive repeating state cycle* is used if the corresponding sequence of prime input commands contains a consecutive repeating state cycle.

THEOREM 8.30 Any consecutive repeating state cycle of (Q, A, η) is contained in an edge sequence of (Q, A, η) .

PROOF. This follows immediately from definition 8.29 and lemmas 8.15 and 8.19.

REMARK 8.31 *Period of an immortal periodic point contained in edge sequence*

If $E([p_1, p_2, \dots, p_r], r)$ contains a consecutive repeating state cycle, then the corresponding

immortal periodic point has period $\leq \frac{1}{2} \sum_{k=1}^r |\mathcal{H}_{p(k)} \Rightarrow \mathcal{T}_{p(k)}|$.

PROOF. This follows from lemma 7.11 that a consecutive repeating state cycle induces 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 $|\mathcal{H}_{p(k)} \Rightarrow \mathcal{T}_{p(k)}|$ in directed edge $\mathcal{H}_{p(k)} \Rightarrow \mathcal{T}_{p(k)}$.

METHOD 8.32 *Finding a consecutive repeating state cycle in an edge sequence*

Given an edge sequence whose corresponding prime input command sequence

$(q_0, a_0) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_N, a_N)$ has length N .

Set $n = N/2$ if N is even; otherwise, set $n = (N+1)/2$ if N is odd

for each k in $\{1, 2, \dots, n\}$

{

for each j in $\{0, 1, \dots, N-2k-1\}$

{

if sequence $(q_j, a_j) \mapsto (q_{j+1}, a_{j+1}) \mapsto \dots \mapsto (q_{j+k}, a_{j+k})$ is equal to sequence

$(q_{j+k+1}, a_{j+k+1}) \mapsto (q_{j+k+2}, a_{j+k+2}) \mapsto \dots \mapsto (q_{j+2k+1}, a_{j+2k+1})$

then

{

return consecutive repeating state cycle

$(q_j, a_j) \mapsto (q_{j+1}, a_{j+1}) \mapsto \dots \mapsto (q_{j+k}, a_{j+k}) \mapsto \dots \mapsto (q_{j+2k+1}, a_{j+2k+1})$

}

}

}

If exited outer for loop without finding a consecutive repeating state cycle

Return NO consecutive repeating state cycles were found.

EXAMPLE 8.33 *A new LISP function that finds a consecutive repeating sequence*

```
(define (find_pattern_repeats p_length seq)
  (let
    (
      (k 0)
      (max_k (- (length seq) (+ p_length p_length)) )
      (pattern nil)
      (repeat_pair nil)
      (no_repeats true)
    )
    (while (and (<= k max_k) no_repeats)
      (set 'pattern (slice seq k p_length))
      (if (= pattern (slice seq (+ k p_length) p_length))
        (begin
          (set 'repeat_pair (list pattern k))
          (set 'no_repeats false)
        )
      )
      (set 'k (+ k 1))
    )
    repeat_pair
  ))

(define (find_repeats seq)
  (let
    (
      (p_length 1)
      (max_p_length (/ (length seq) 2) )
      (repeat_pair nil)
    )
    (while (and (<= p_length max_p_length) (not repeat_pair))
      (set 'repeat_pair (find_pattern_repeats p_length seq))
      (set 'p_length (+ p_length 1))
    )
    repeat_pair
  ))

(set 'seq1 '(3 5 7 2 3 5 7 11 5 7) )
;; seq1 does not have a consecutive repeating sequence.

(set 'seq2 '(3 5 7 2 3 5 7 11 5 7 11 2 4 6 8) )
;; 5 7 11 5 7 11 is a consecutive repeating sequence starting at element in list seq2

(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) )
;; 0 1 0 1 is a consecutive repeating sequence starting at element 38 in list seq3

> (find_repeats seq1)
nil

> (find_repeats seq2)
( (5 7 11) 5)

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

METHOD 8.34 *Prime Directed Edge Search Method*

Given Turing Machine (Q, A, η) as input, the search method works as follows.

Set $\mathcal{P} = \emptyset$.

For each non-halting state q in Q

For each pattern $a_{-|Q|} \dots a_{-2} a_{-1} a_0 a_1 a_2 \dots a_{|Q|}$ selected from $A^{2|Q|+1}$

{

Tape Square			$- Q $		-2	-1	0	1	2		$ Q $	
Tape Contents			$a_{- Q }$	\dots	a_{-2}	a_{-1}	a_0	a_1	a_2	\dots	$a_{ Q }$	
Start State	q											

With tape head located at a_0 , start executing machine (Q, A, η) until one state has been visited twice or (Q, A, η) reaches a halting state. The Dirichlet principle implies this will take at most $|Q|$ computational steps. If it does not halt, let r be the state that is first visited twice. As defined in 8.1, over this window of execution, a prime directed edge $\mathcal{H} \Rightarrow \mathcal{T}$ is constructed where $\mathcal{H} = [q, v_0 v_1 \dots v_n, s]$, $\mathcal{T} = [r, w_0 w_1 \dots w_n, t]$ and $0 \leq s, t \leq n \leq |Q|$.

Set $\mathcal{P} = \mathcal{P} \cup \{\mathcal{H} \Rightarrow \mathcal{T}\}$

}

REMARK 8.35 *Prime Directed Edge Search Method finds all prime directed edges*

Method 8.34 finds all prime directed edges of (Q, A, η) and all halting nodes.

PROOF. Let $\mathcal{H} \Rightarrow \mathcal{T}$ be a prime directed edge of (Q, A, η) . Then $\mathcal{H} \Rightarrow \mathcal{T}$ has a head node $\mathcal{H} = [r, v_0 v_1 \dots v_n, s]$, for some state r in Q , for some tape pattern $v_0 v_1 \dots v_n$ that lies in A^{n+1} , such that $n \leq |Q|$ and $0 \leq s \leq n$. In the outer loop of 8.34, when r is selected from Q and in the inner loop when the tape pattern $a_{-|Q|} \dots a_{-2} a_{-1} a_0 a_1 a_2 \dots a_{|Q|}$ is selected from $A^{2|Q|+1}$ such that

$$\begin{array}{ccccccc}
 a_0 = v_s & a_1 = v_{s+1} & \dots & a_k = v_{s+k} & \dots & a_{n-s} = v_n & \\
 a_{-1} = v_{s-1} & a_{-2} = v_{s-2} & \dots & a_{-k} = v_{s-k} & \dots & a_{-s} = v_0 &
 \end{array}$$

then the machine execution in **8.34** will construct prime directed edge $\mathcal{H} \Rightarrow \mathcal{T}$.

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 thus, 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.

METHOD 8.36 *Immortal Periodic Point Search Method*

Given Turing Machine (Q, A, η) as input, the method works as follows.

Use method **8.34** to find all prime directed edges, \mathcal{P} .

set $k = 1$.

set $\mathcal{E}(1) = \{ E([1], 1), E([2], 1), \dots, E([|\mathcal{P}|], 1) \}$

while ($\mathcal{E}(k) \neq \emptyset$)

{

 set $\mathcal{E}(k+1) = \emptyset$.

 for each $E([p_1, p_2, \dots, p_k], k)$ in $\mathcal{E}(k)$

 {

 for each prime directed edge $\mathcal{H}_j \Rightarrow \mathcal{T}_j$ in \mathcal{P}

 {

 if $\mathcal{H}_j \Rightarrow \mathcal{T}_j$ link matches with $\mathcal{N}_{p(k)}$ then

 {

 set $p_{k+1} = j$

 set $\mathcal{E}(k+1) = \mathcal{E}(k+1) \cup E([p_1, p_2, \dots, p_k, p_{k+1}], k+1)$

 if $E([p_1, p_2, \dots, p_k, p_{k+1}], k+1)$ contains a consecutive repeating state cycle
 then return the consecutive repeating state cycle

 }

 }

 }

k is incremented.

}

if (while loop exited because $\mathcal{E}(m) = \emptyset$ for some m) then return \emptyset

REMARK 8.37 $|\mathcal{E}(k)|$ is finite and $|\mathcal{E}(k)| \leq |\mathcal{P}|^k$

PROOF. $|\mathcal{E}(1)| = |\mathcal{P}|$. Analyzing the nested loops, in method **8.36**

 for each $E([p_1, p_2, \dots, p_k], k)$ in $\mathcal{E}(k)$

for each $\mathcal{P}_j \Rightarrow \mathcal{T}_j$ in $\mathcal{P} \{ . . . \}$

For each edge sequence $E([p_1, p_2, \dots, p_k], k)$ chosen from $\mathcal{E}(k)$, at most $|\mathcal{P}|$ new edge sequences are put in $\mathcal{E}(k+1)$. Thus $|\mathcal{E}(k+1)| \leq |\mathcal{P}| |\mathcal{E}(k)|$, so $|\mathcal{E}(k)| \leq |\mathcal{P}|^k$.

THEOREM 8.38 *Method 8.36 terminates in a finite number of steps with either a consecutive repeating state cycle or for some positive integer J , then $\mathcal{E}(J) = \emptyset$*

PROOF. If (Q, A, η) has at least one configuration (q, k, T) that has an immortal orbit, then theorem 4.16 implies the existence of a periodic point p with some finite period N .

Thus, from lemma 7.10, there is a consecutive repeating state cycle that corresponds to the immortal periodic orbit of p . Since method 8.36 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, this immortal periodic point of period N will be reached before or while computing $\mathcal{E}(2N)$.

Otherwise, (Q, A, η) does not have any configurations with an immortal orbit; in other words, for every configuration, (Q, A, η) halts in a finite number of steps.

Claim: There is a positive integer J such that every edge sequence terminates while executing method 8.36. By reductio absurdum, suppose not. Then there is at least one infinite prime directed edge sequence that exists: this corresponds to an immortal orbit, which contradicts that (Q, A, η) does not have any configuration with an immortal orbit.

The next few definitions are paraphrased or directly stated from [HOOPER].

DEFINITION 8.39 *Instantaneous Description* [HOOPER]

A tape with designated scanned symbol and an internal state of the Turing Machine together constitute an *instantaneous description* (ID) of the Turing Machine. Recall from definitions 2.1, and 2.2 that given Turing Machine (Q, A, η) , and tape T , where $T: \mathbb{Z} \rightarrow A$, then a *configuration* is an element (q, k, T) for some state q in Q and for some tape head location k in \mathbb{Z} . Observe that Hooper's *scanned symbol* and our *tape head location* represent the same meaning. Thus, Hooper's *instantaneous description* is equivalent to the definition of a Turing Machine *configuration*.

DEFINITION 8.40 *Immortal and Mortal Configuration*

If a configuration (ID) upon execution of Turing Machine (Q, A, η) halts after a finite number of computational steps, then it is called a *mortal configuration* (mortal ID). Otherwise a configuration (ID) is called *immortal*.

DEFINITION 8.41 *Halting Machine*

Turing Machine (Q, A, η) is called a *halting machine* if it contains only *mortal* (halting) *configurations*.

COROLLARY 8.42 *Method 8.36 determines whether (Q, A, η) is a halting machine.*

This follows immediately from theorem 8.38 and method 8.36.

DEFINITION 8.43 *Turing Immortality Problem and Halting Problem*

The classic result of Turing machine theory is the undecidability of the problem of determining for a given *configuration* of a Turing machine (Q, A, η) is whether this configuration is immortal or mortal. This is called the *halting problem*.

The *immortality problem* is the problem of deciding for a given Turing machine (Q, A, η) whether it is a *halting machine*; in other words, determining whether an immortal configuration exists for the machine (Q, A, η) .

THEOREM 8.44 *Turing Immortality Problem is Turing Undecidable*

PROOF. This is the main result of [HOOPER].

THEESIS 8.45 *Church-Turing Thesis*

Quoted from [LEWIS, p. 223] with italics added for emphasis.

“Because the Turing machines can carry out any computation that can be carried out by any similar type of automata, and because these automata seem to capture the essential features of real computing machines, we take the Turing machine to be a precise formal equivalent of the intuitive notion of *algorithm*: nothing will be considered an algorithm if it cannot be rendered as a Turing machine. The principle that Turing machines are formal versions of algorithms and that no computational procedure will be considered an algorithm unless it can be presented as a Turing machine is known as Church's thesis or the Church-Turing thesis. It is a thesis, not a theorem, because it is not a mathematical result: It simply asserts that a certain informal concept corresponds to a certain mathematical object. It is theoretically possible, however, that Church's thesis could be overthrown at some future date, if someone were to propose an alternative model of computation that was publicly acceptable as fulfilling the requirement of *finite labor at each step* and yet was *provably capable of carrying out computations that cannot be carried out by any Turing machine*. No one considers this likely.”