

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

DEFINITION 7.1 *Overlap Matching & Intersection Patterns*

The notion of an *overlap match* expresses how a part or all of one pattern may match part or all of another pattern. 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 $-\mathcal{L} \leq c \leq \mathcal{R}$ such that $\mathcal{L} = \min\{s, t\}$ and $\mathcal{R} = \min\{|V| - 1 - s, |W| - 1 - t\}$ where $0 \leq s < |V|$ and $0 \leq t < |W|$. The index s is called the *head* of pattern V and 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 $u = \mathcal{L}$ as $(I, u) = (V, s) \cap (W, t)$, where $I(c) = V(c + s - \mathcal{L})$ for every integer c satisfying $0 \leq c \leq (\mathcal{R} + \mathcal{L})$ where $\mathcal{L} = \min\{s, t\}$ and $\mathcal{R} = \min\{|V| - 1 - s, |W| - 1 - t\}$.

DEFINITION 7.2 *Edge Pattern Substitution Operator*

Consider pattern $V = v_0 v_1 \dots v_n$, pattern $W = w_0 w_1 \dots w_n$ with heads s, t satisfying $0 \leq s, t \leq n$ and pattern $P = p_0 p_1 \dots p_m$ with head u satisfying $0 \leq u \leq m$. Suppose (P, u) *overlap matches* (V, s) . Then 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 $m - u > n - s$

p_0	p_{u-s}	p_u	...	p_{u+n-s}		p_m
v_0	v_1	...	v_s	...	v_n					
w_0	w_1	...	w_s	...	w_n					

$$E(k) = \left\{ \begin{array}{ll} W(k+s-u) & \text{when } u \leq k+s \leq u+n \\ P(k) & \text{when } 0 \leq k < u-s \text{ OR } u+n-s < k \leq m \end{array} \right\}$$

where the head of E is $u+t-s$. Observe that $|E| = m+1$

Case B.) $u > s$ and $m-u \leq n-s$

p_0	p_1	\dots	\dots	p_{u-s}	\dots	\dots	p_u	\dots	p_m
v_0	v_1	\dots	\dots	v_s	\dots	\dots	v_{s+m-u}		v_n
w_0	w_1	\dots	\dots	w_s	\dots	\dots	w_{s+m-u}		w_n

$$E(k) = \left\{ \begin{array}{ll} W(k+s-u) & \text{when } u-s \leq k \leq n+s-u \\ P(k) & \text{when } 0 \leq k < u-s \end{array} \right\}$$

where the head of E is $u+t-s$. Also, $|E| = n+s-u+1$

Case C.) $u \leq s$ and $m-u \leq n-s$

p_0	\dots	p_u	\dots	p_m				
v_0	\dots	v_{s-u}	\dots	v_s	\dots	v_{s+m-u}		v_n
w_0	\dots	w_{s-u}	\dots	w_s	\dots	w_{s+m-u}		w_n

$E(k) = W(k)$ when $0 \leq k \leq n$ and the head of E is t . Also, $|E| = |W| = n+1$.

Case D.) $u \leq s$ and $m-u > n-s$

p_0		\dots		p_u	\dots	p_{u+n-s}		\dots	p_m
v_0	\dots	v_{s-u}	\dots	\dots	v_s	\dots	v_n		
w_0	\dots	w_{s-u}	\dots	\dots	w_s	\dots	w_n		

$$E(k) = \left\{ \begin{array}{ll} P(k+u-s) & \text{when } n < k \leq m+s-u \\ W(k) & \text{when } 0 \leq k \leq n \end{array} \right\}$$

where the head of E is t . Also, $|E| = m+s-u+1$

Overlap and intersection matching and edge pattern substitution will be quite useful in sections 7, 8 and 10.

EXAMPLE 7.3 *Overlap Matching and Edge Substitution*

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 $(V, 2)$. Edge pattern substitution 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 $E = 4$.

Also, $(P, 4)$ overlap matches $(V, 0)$. $F = (P, 4) \oplus [(V, 0) \Rightarrow (W, 4)] = 0101\ 0100\underline{1}0$. The index of pattern $F = u + t - s = 4 + 4 - 0 = 8$.

DEFINITION 7.4 *State Cycle*

Consider N execution steps of Turing Machine (Q, A, η) . After each execution step, the machine is in some state q_k and the tape head is pointing to some alphabet symbol a_k . Relabeling the indices of the states and the alphabet symbols if necessary and assuming the machine has not halted after N execution steps in terms of the input commands is denoted as: $(q_0, a_0) \mapsto (q_1, a_1) \mapsto \dots \mapsto (q_{N-1}, a_{N-1}) \mapsto (q_N, a_N)$. A *state cycle* is a valid execution sequence of input commands such that the first and last input command in the sequence have the same state i.e. $(q_k, a_k) \mapsto (q_{k+1}, a_{k+1}) \mapsto \dots \mapsto (q_{N-1}, a_{N-1}) \mapsto (q_k, a_k)$. 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 $\{2, 3, 4\}$.

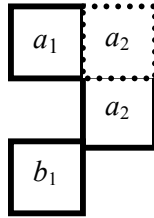
REMARK 7.5 *Any prime state cycle has length $\leq |Q|$*

This follows from the Dirichlet principle and the definition of a prime state cycle.

REMARK 7.6 *Maximum number of distinct prime state cycles*

Given an alphabet A and states Q , consider an arbitrary prime state cycle with length 1, $(q, a) \mapsto (q, b)$. There are $|Q| |A|$ choices for the first input command and $|A|$ choices for the second input command since the states must match. Thus, there are $|Q| |A|^2$ distinct prime state cycles with length 1.

Similarly, consider a prime state cycle with window of execution whose length is 2, this can be represented as $(q_1, a_1) \mapsto (q_2, a_2) \mapsto (q_1, b_1)$. For tape head move sequence RL , it looks like:



Then there are $|Q| |A|$ choices for (q_1, a_1) and once (q_1, a_1) is chosen there is only one choice for q_2 because it is completely determined by $\eta(q_1, a_1) = (q_2, b_1)$ where η is the program in (Q, A, η) . Similarly, there is only one choice for b_1 . There are $|A|$ choices for a_2 . Thus, there are $|Q| |A|^2$ distinct choices.

For an arbitrary prime state cycle $(q_1, a_1) \mapsto (q_2, a_2) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_{n+1})$ with window of execution of length k then there are $|Q| |A|$ choices for (q_1, a_1) and $|A|$ choices for a_2 since the current window of execution length after the first step increases by 1. There is only one choice for q_2 because it is determined by $\eta(q_1, a_1)$. Similarly, for the j th computational step, if the current window of execution length increases by 1, then there are $|A|$ choices for (q_{j+1}, a_{j+1}) . Similarly, for the j th computational step, if the current window of execution stays unchanged, then there is only one choice for a_{j+1} that was determined by one of the previous j computational steps. Thus, there are at most $|Q| |A|^k$ distinct prime state cycles whose window

of execution length equals k . Definitions 3.15 and remark 3.16 imply that a prime k -state cycle has a window of execution length less than or equal to k . Thus, from the previous and 7.5,

there are at most $|Q| \sum_{k=1}^{|Q|} |A|^k$ distinct prime state cycles in (Q, A, η) .

REMARK 7.7 *Any state cycle contains a prime state cycle*

PROOF. Relabeling if necessary let $\mathcal{C}(q_1, q_1) = (q_1, a_1) \mapsto \dots \mapsto (q_n, a_n) \mapsto (q_1, a_{n+1})$ be a state cycle. If q_1 is the only state visited twice, then the proof is completed. Otherwise, define

$\mu = \min \{ |\mathcal{C}(q_k, q_k)| : \mathcal{C}(q_k, q_k) \text{ is a subcycle of } \mathcal{C}(q_1, q_1) \}$. Then μ exists because

$\mathcal{C}(q_1, q_1)$ is a subcycle of $\mathcal{C}(q_1, q_1)$. Claim: Any state cycle $\mathcal{C}(q_j, q_j)$ with $|\mathcal{C}(q_j, q_j)| = \mu$ 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 $\mathcal{C}(q_j, q_j)$. But then $\mathcal{C}(q_r, q_r)$ is a cycle with length less than μ which contradicts μ 's definition.

DEFINITION 7.8 *Consecutive repeating state cycle for (Q, A, η)*

If machine (Q, A, η) starts execution and repeats a state cycle two consecutive times i.e. $(q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1)$, then (Q, A, η) has a *consecutive repeating state cycle*.

DEFINITION 7.9 *Execution node for (Q, A, η)*

An *execution node* (or node) is a triplet $\mathcal{N} = [q, w_0 w_1 \dots w_n, t]$ for some state q in Q where $w_0 w_1 \dots w_n$ is a pattern of $n + 1$ alphabet symbols each in A such that t is a non-negative integer satisfying $0 \leq t \leq n$. Intuitively, $w_0 w_1 \dots w_n$ is the pattern of alphabet symbols on $n + 1$ consecutive tape squares on the tape and t represents the location of the tape head.

LEMMA 7.10 *Every immortal periodic point induces a consecutive repeating state cycle.*

PROOF. Suppose p is an immortal periodic point with period n . Then by the Turing-Affine correspondence theorem the k th iterate of p is $f_{S(k)} f_{S(k-1)} \dots f_{S(1)}(p)$ and the application of affine function $f_{S(k)}$ corresponds to the execution of input command (q_k, b_k) . Thus, let the input command sequence $(q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_{n+1}, b_{n+1})$ denote the first n input commands that are executed. Since p has period n , $f_{S(n)} \dots f_{S(k)} \dots f_{S(1)}(p) = p$. Thus, $(q_1, b_1) = (q_{n+1}, b_{n+1})$. Thus, the first n steps are a state cycle $(q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1)$. Since the $n+1$ computational step corresponds to applying $f_{S(1)}$ to p which corresponds to input command (q_1, b_1) . By induction, the $n+k$ computational step corresponds to applying function $f_{S(k)}$ to the point $f_{S(k-1)} \dots f_{S(1)}(p)$ which by the previous paragraph corresponds to the execution of the input command (q_k, b_k) . Thus, the sequence of input commands is $(q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1)$.

LEMMA 7.11 *Every consecutive repeating state cycle induces an immortal periodic orbit*

Suppose Turing machine (Q, A, η) begins or resumes execution at some tape square and repeats a state cycle two consecutive times. Then (Q, A, η) has an immortal periodic point and this state cycle induces the immortal periodic point.

PROOF. Let the state cycle that is repeated two consecutive times be denoted as

$(q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1) \mapsto \dots \mapsto (q_n, b_n) \mapsto (q_1, b_1)$. Let s_k denote the tape square right 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 right before input command (q_k, b_k) is executed the second time where $1 \leq k \leq n$.

Thus, the window of execution for the first repetition of the state cycle, right before input command (q_1, b_1) is executed a second time, denoted $I_n = \{s_1, s_2, \dots, s_k, s_{k+1} \dots s_n, s_{n+1}\}$ where $s_{n+1} = t_1$. The window of execution for the second repetition of the state cycle is $J_n = \{t_1, t_2, \dots, t_n, t_{n+1}\}$ 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 $I_k = \{s_1, s_2, \dots, s_k, s_{k+1}\}$ where the tape square s_{k+1} is indicated 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 $J_k = \{t_1, t_2, \dots, t_k, t_{k+1}\}$ where the tape square t_{k+1} is indicated 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 – right 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 – right 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 – right 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 – right 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.

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 .

From the definition, $\eta(q_1, b_1) = \eta(q_2, a_1, x)$ for some a_1 in A and where x equals L or R . Note that L represents a left tape head move and R a right tape head move.

Case $x = R$. A right tape head move.

	s_1	s_2	
V_1	<u>b_1</u>	b_2	
V_2	a_1	<u>b_2</u>	

	t_1	t_2	
W_1	<u>b_1</u>	b_2	
W_2	a_1	<u>b_2</u>	

Then $s_2 = s_1 + 1$, $t_2 = t_1 + 1$ 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 on the window of execution J_2 . Furthermore, the tape head is at s_1 right before computational step 1 and input command (q_1, b_1) is executed; the tape head is at t_1 right 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 right before computational step 2 with input command (q_2, b_2) is executed; the tape head is at t_2 right before computational step $n+2$ with input command (q_2, b_2) is executed.

Case $x = L$. A left tape head move.

	s_2	s_1	
V_1	b_2	$\underline{b_1}$	
V_2	$\underline{b_2}$	a_1	

	t_2	t_1	
W_1	b_2	$\underline{b_1}$	
W_2	$\underline{b_2}$	a_1	

Then $s_2 = s_1 - 1$, $t_2 = t_1 - 1$ 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 on the window of execution J_2 . Furthermore, the tape head is at s_1 right before computational step 1 and input command (q_1, b_1) is executed; the tape head is at t_1 right 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 right before computational step 2 and input command (q_2, b_2) is executed; the tape head is at t_2 right 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 $1, 2, \dots, k-1$ computational steps and the corresponding $n+1, n+2, \dots, n+k-1$ steps that for every i with $1 \leq i \leq k$

- V_i restricted to the window of execution I_i equals W_i restricted on the window of execution J_i ; V_2 restricted to the window of execution I_i equals W_2 restricted on the window of execution J_i ; and \dots V_i restricted to the window of execution I_i equals W_i restricted on the window of execution J_i .

- Furthermore, the tape head is at s_i right before computational step i and input command (q_i, b_i) is executed; the tape head is at t_i right before computational 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 $V_k(s_k) = b_k = W_k(t_k)$.

From the definition, $\eta(q_k, b_k) = \eta(q_{k+1}, a_k, x)$ for some a_k in A and x equals L or R . Note that L represents a left tape head move and R a right tape head move.

Case $x = R$. A right tape head move for computational steps k and $n + k$.

	s_k	s_{k+1}	
V_k	<u>b_k</u>	b_k	
V_{k+1}	a_k	<u>b_{k+1}</u>	

	t_k	t_{k+1}	
W_k	<u>b_k</u>	b_k	
W_{k+1}	a_k	<u>b_{k+1}</u>	

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 $\eta(q_k, b_k) = \eta(q_{k+1}, a_k, R)$ 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 the window of execution I_{k+1} equals W_{k+1} restricted on the window of execution J_{k+1} . And for each j satisfying $1 \leq j \leq k$, then V_j restricted to the window of execution I_{k+1} equals W_j restricted on the window of execution J_{k+1} .

Case $x = L$. A left tape head move for computational steps k and $n + k$.

	s_{k+1}	s_k	
V_k	b_{k+1}	\underline{b}_k	
V_{k+1}	\underline{b}_{k+1}	a_k	

	t_{k+1}	t_k	
W_k	b_{k+1}	\underline{b}_k	
W_{k+1}	\underline{b}_{k+1}	a_k	

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 $\eta(q_k, b_k) = \eta(q_{k+1}, a_k, L)$ 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 the window of execution I_{k+1} equals W_{k+1} restricted on the window of execution J_{k+1} . And for each j satisfying $1 \leq j \leq k$, then V_j restricted to the window of execution I_{k+1} equals W_j restricted on the window of execution J_{k+1} .