

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

Design of the IDRM

SECTION 10

In the next section, a complete description of a *dynamic register machine program*, called a IDRM, is presented that implements methods 8.34 and 8.36, which includes method 8.32. In other words, for any arbitrary Turing Machine (Q, A, η) as input, the IDRM determines whether machine (Q, A, η) has an immortal periodic point and hence solves the Turing immortality problem. As a result, this IDRM is a *constructive demonstration* that the Church-Turing Thesis is false. In this section, the design of this machine is explained from a human-friendly perspective. This section describes how the Turing machine (Q, A, η) is represented and where it is located in the registers of the IDRM. This section also describes how methods 8.34 and 8.36 are implemented and executed by the IDRM program.

In section 11, the complete formal description of the IDRM is a program of 1590 instructions composed from the *Constant* (C m n), *Successor* (S m), *Transfer* (T m n), *Address* (T m n), *Jump* (J m n q), *Delete* (D m n), and *Update* (U m n q) *Instructions*. With a valid Turing Machine as input in its registers, this IDRM solves the Turing Immortality problem after a finite number of execution steps.

REPRESENTATION 10.1 *Turing Machine State, Alphabet and Command*

The states Q of machine (Q, A, η) are represented with the natural numbers $\{1, 2, \dots, |Q|\}$. The alphabet symbols are represented with natural numbers $\{1, 2, \dots, |A|\}$. The function η determines all the Turing machine commands. A Turing machine command is of the form $(q \ a \ r \ b \ m)$ where $2 \leq q \leq |Q|$, where $1 \leq r \leq |Q|$, where $1 \leq a, b \leq |A|$

and where $1 \leq m \leq 2$. The variable $m = 1$ represents a left tape head move and $m = 2$ represents a right tape head move. As described in definition 2.1, the design of the IDRM assumes that 1 with respect to Q is the halting state. This is why $q \geq 2$.

REPRESENTATION 10.2 *Turing Machine Program as Input in the IDRM Registers*

Registers 0 to 155 are reserved for computation, program flow and important constants. The design of the IDRM assumes that registers 156 through 160 contain the first Turing machine command which is of the form $(2 \ 1 \ r \ b \ m)$. Further, it is assumed that the instructions for the Turing machine are dictionary ordered over the first two coordinates. In other words, each Turing machine command of the form $(q \ a \ r \ b \ m)$ is stored as follows:

- A.) q is stored in register number $5(q-1)a + 151$
- B.) a is stored in register number $5(q-1)a + 152$
- C.) r is stored in register number $5(q-1)a + 153$
- D.) b is stored in register number $5(q-1)a + 154$
- E.) m is stored in register number $5(q-1)a + 155$

The initial register contents starting at register 156 are shown in the table:

2	1	r_{21}	b_{21}	m_{21}	2	2	r_{22}	b_{22}	m_{22}	.	.	.	2	$ A $	$r_{2 A }$	$b_{2 A }$	$m_{2 A }$
3	1	r_{31}	b_{31}	m_{31}	3	2	r_{32}	b_{32}	m_{32}	.	.	.	3	$ A $	$r_{3 A }$	$b_{3 A }$	$m_{3 A }$
4	1	r_{41}	b_{41}	m_{41}	4	2	r_{42}	b_{42}	m_{42}	.	.	.	4	$ A $	$r_{4 A }$	$b_{4 A }$	$m_{4 A }$
.
$ Q $	1	$r_{ Q 1}$	$b_{ Q 1}$	$m_{ Q 1}$	$ Q $	2	$ Q $	$ A $	$r_{ Q A }$	$b_{ Q A }$	$m_{ Q A }$
0																	

This means register 157 contains a 1; register 158 contains r_{21} ; register 159 contains b_{21} register 160 contains m_{21} ; register $(160 + 5|A|)$ contains m_{31} and so on.

The \emptyset in register number $5(|Q|-1)|A| + 156$ indicates the end of the Turing machine program. At the beginning of program execution, the design of the IDRM assumes that register 0 contains the address $5(|Q|-1)|A| + 156$. In other words, $(R \emptyset) = 5(|Q|-1)|A| + 156$. This assumption is made so that the IDRM can determine if the Turing machine program is presented in a *valid format* in registers 156 through register $(5(|Q|-1)|A| + 155)$. This is analogous to the notion of a *well-formed formula* in mathematical logic. See [ENDERTON].

DESCRIPTION 10.3 IDRM *Program Summary*

Overall, the purpose of instructions 0 to 296 in the IDRM program is to check that the Turing machine (Q, A, η) is stored in a valid format starting at register 156 and to set up registers for specific purposes based on $|Q|$ and $|A|$ in machine. Overall, the purpose of instructions 297 to 857 in the IDRM is to construct a linked list of all, if any exist, the prime directed edges of the given Turing Machine (Q, A, η) whose command set starts at register 156. These 858 instructions execute method **8.34**. After the prime directed edge search method described in **8.34** is completed, instructions 858 to 1589 of the IDRM program execute the *immortal periodic point search* method **8.36**. Next, further details of the IDRM program are discussed.

DESCRIPTION 10.4 IDRM *Program Instructions 0 to 296*

Instructions 0 through 126 in the IDRM determine whether the Turing machine represented in registers 156 through register $(5(|Q|-1)|A| + 155)$ is in a valid format. If not, then execution of the IDRM jumps to instruction 1570 and writes “ERROR TURING PROGRAM” in registers 0 through 19 using the ASCII code representation. In other words, 65 corresponds to “A”; 69 corresponds to “E”; 82 corresponds to “R” and so on. By doing this valid format checking, it

assures that the execution of the IDRM always terminates regardless of the initial contents of the registers in the IDRM.

Observe that rather than omitting a Turing machine command, as this is the convention in [TURING] and [DAVIS], if $\eta(q, a)$ halts, the IDRM uses the convention $(q \ a \ 1 \ b \ m)$ to represent this halting instruction as specified in section 2. Precisely, the natural number q is stored in register $5(q-1)a + 151$; the number a is stored in register $5(q-1)a + 152$; 1 is stored in register $5(q-1)a + 153$; b is stored in register $5(q-1)a + 154$; and m is stored in register $5(q-1)a + 155$.

After the \emptyset in register $(5(|Q|-1)|A| + 156)$ there are four unused registers, followed by the A offsets: $A_2, A_3, \dots, A_{|Q|}$. The A offsets are used to help the IDRM more effectively lookup the next Turing command based on the prior state and tape symbol. The A offsets are determined and stored by instructions 127 through 153 of the IDRM.

After the A offsets, some registers are used to store information necessary to do the prime edge computations for each tape pattern element. The location of these registers are based on the values of $|A|$ and $|Q|$. In more detail, instructions 154 through 296 set up a *scratch pad* of registers needed to execute method **8.34** i.e. the finding and storing of all prime directed edges of (Q, A, η) -- if any exist. In particular, instructions 154 through 167 set up registers to record the current prime input command sequence. Instructions 168 to 190 set up registers to record the current prospective head node. Instructions 191 to 212 set up registers to record the current prospective tail node.

Instructions 213 to 233 set up registers to record the execution tape during the computation of a prospective prime directed edge. Instructions 234 to 254 set up registers to store an iteration tape, which is used to iterate through every tape pattern in $A^{2|Q|+1}$. Instructions 255 to 265 set up registers to record the tape head moves during the computation of the prospective prime edge. Instructions 266 to 292 set up registers to compute the window of execution. Instructions 293 to 296 set up memory pointers to record the number of prime edges found so far and store a linked list of prime edges. In instruction 296, a *free heap pointer* is stored in register 155.

DESCRIPTION 10.5 IDRM *Program implementing prime directed edge search method 8.34*

For each non-halting state q in Q and for each tape pattern in $A^{2|Q|+1}$ the IDRM program executes machine (Q, A, η) for at most $n \leq |Q|$ computational steps of the Turing machine. As a result, this particular tape pattern and starting state q form a prime directed edge or they do not. If they do, the prime directed edge is stored in a *prime edge linked list*.

Each element (node) in the *prime edge linked list* is stored contiguously in registers as follows:
 next_node, pe_number, h_node, t_node, s_node, n, q, s, s_p, $v_0, v_1 \dots v_n$,
 $r, t, t_p, w_0, w_1 \dots w_n, 2N, (q, v_s), (q_1, a_1) \dots (q_N, a_N)$

In other words, if next_node is stored in register 7000, then pe_number is stored in register 7001. h_node is stored in register 7002. t_node is stored in register 7003. s_node is stored in register 7004. n is stored in register 7005. q is stored in register 7006. s is stored in register 7007. s_p is stored in register 7008 and so on.

- The contents of register next_node stores the register number of the next node.
- The register contents pe_number stores the name (a number) of this prime edge.

- Register h_node contains the address of the register state q in the head node.
- Register t_node contains the address of the register state r in the tail node.
- Register s_node contains the address of $2N$ in the prime input command sequence.
- The window of execution size is stored in the register n
- q is the state of the head node.
- s is the index of the head node.
- s_p points to the location of tape symbol v_s (i.e. index pointer of the head node)
- r is the state of the tail node.
- t is the index of the tail node.
- t_p points to the location of tape symbol v_s (i.e. index pointer of the head node)
- N is the number of computational steps for this prime directed edge.
- $N+1$ is the number of prime input commands in the prime input command sequence.
- Tape symbol $a_0 =$ Tape symbol v_s
- State $r =$ State q_N
- Register 81 stores the number of prime directed edges.
- The node numbers count sequentially up to the number of prime directed edges.
- $v_0, v_1 \dots v_n$ is the initial tape pattern of the head node.
- $w_0, w_1 \dots w_n$ is the final tape pattern of the tail node.
- $(q, v_s), (q_1, a_1), \dots, (q_N, a_N)$ is the prime input command sequence corresponding to this prime directed edge.

Now that the data structure for representing and storing prime directed edges in the registers has been described, more detail is provided on the instructions 297 to 857 that execute the prime directed edge search method.

In particular, instructions 297 to 313 initialize the tape pattern in $A^{2|Q|+1}$ to all 1's. Instruction 314 begins the outer loop that iterates from state 2 up to state $|Q|$. This is the outer for loop in method 8.34, expressed as: For each non-halting state q in Q .

Instructions 331 to 349 copy the iterated tape pattern, starting at the register pointed to by register 68. Instruction 334 stores the value of $2|Q|+1$ in register 2. Instructions 350 to 367 initialize the prospective head node state and the tape symbols. Instructions 371 to 395 initialize register 33 to the correct command $(q \ a \ r \ b \ m)$ in the Turing command table. The current state q of the Turing machine is stored in register 65. The current tape symbol a is stored in register 66.

Instructions 396 to 408 store $(q_k \ a_k)$ in the register that is pointed to by the contents of register 51. Instructions 409 to 483 execute one computational step of the Turing machine whose command table starts at register 156. Instruction 484 increments register 48 which stores the number of computational steps for this prospective prime directed edge.

Instructions 485 to 496 check to see if state q_{k+1} has already been visited. If so, the program execution jumps to instruction 497. Instructions 497 to 509 store $(q_N \ a_N)$ starting at the register pointed to by the contents of register 51. Instructions 510 to 550 compute the window of execution. Instructions 551 to 578 increment the tape pattern which is an element of $A^{2|Q|+1}$. This enables the program to search every element in $A^{2|Q|+1}$ as a prospective head node.

Instructions 579 to 778 copy this new prime directed edge found to the end of the prime directed edge linked list, whose data structure format has already been described.

Instructions 779 and 780 update the prime directed edge linked list pointers. Instructions 781 to 839 check if the new head node just found is the same as the head node at the end of the prime directed edge linked list. If yes, then the new prime directed edge is the same as the last one found and it is ignored.

Otherwise, the prime directed edge just found is new and so it is added to the end of the prime directed edge linked list. This is performed in instructions 840 to 854. The number of prime directed edges found so far is incremented in instruction 855.

Instructions 856 and 857 cause program execution to jump back to instruction 314, where the next tape pattern is examined to decide if the current state and the new tape pattern determine a new prime directed edge. When the tape pattern reaches all 1's again, instructions 328, 329 and 330 increment the state q and the next prime directed edge is searched for. Once all states have been exhausted up to the last state value $|Q|$, then the program jumps to instruction 858 where the immortal periodic search method **8.36** starts.

DESCRIPTION 10.6 IDRМ Program implementing immortal periodic search method **8.36**

After the prime directed edge search method described in **8.34** is completed, instructions 858 to 1589 of the IDRМ program execute the *immortal periodic point search* method **8.36**. Following the notation of **8.36**, $\mathcal{E}(k)$ is the set of all edge sequences of length k . The IDRМ program, represents $\mathcal{E}(1)$, the set of prime directed edges as a linked list where each node is in square brackets:

$$[ptr_1, 2N_1, r_1, idx_1, ub_1, lb_ptr_1, idx_ptr_1, ub_ptr_1, 1, \\ w_{0,1}, \dots, w_{idx,1}, \dots, w_{ub,1}, (q_{0,1}, a_{0,1}), (q_{1,1}, a_{1,1}), \dots, (q_{N_1,1}, a_{N_1,1})]$$

$[ptr_2, 2N_2, r_2, idx_2, ub_2, lb_ptr_2, idx_ptr_2, ub_ptr_2, 2,$

$w_{0,2}, \dots w_{idx_2} \dots w_{ub_2}, (q_{0,2}, a_{0,2}), (q_{1,2}, a_{1,2}), \dots, (q_{N_2,2}, a_{N_2,2})]$

\dots

$[ptr_m, 2N_m, r_m, idx_m, ub_m, lb_ptr_m, idx_ptr_m, ub_ptr_m, m,$

$w_{0,m}, \dots w_{idx,m} \dots w_{ub,m}, (q_{0,m}, a_{0,m}), (q_{1,m}, a_{1,m}), \dots, (q_{N_m,m}, a_{N_m,m})]$

- The contents of register ptr_j points to register ptr_{j+1}
- m is the number of prime directed edges.
- $(q_{0,j}, a_{0,j}), (q_{1,j}, a_{1,j}), \dots, (q_{N_j,j}, a_{N_j,j})$ is the sequence of input commands corresponding to the prime directed edges that have been linked matched for this j th edge sequence. For $\mathcal{E}(1)$, this is just the prime input command sequence corresponding to the j th prime edge.
- $2N_j$ is stored in the register following ptr_j where $N_j + 1$ is the number input command pairs.
- r_j is the state of tail node τ for the j th prime edge which corresponds to state r in definition 8.1.
- $w_{0,j} \dots w_{idx,j} \dots w_{ub,j}$ represents the tape of the tail node and the tape head points to alphabet symbol $w_{idx,j}$
- idx_j contains the tape head index which corresponds to the index t in the tail node τ of definition 8.1.
- ub_j contains the upper bound, where the window of execution is $[0, ub_j]$
- The contents of register lb_ptr_j points to the register containing tape symbol $w_{0,j}$
- The contents of register idx_ptr_j points to the register containing tape symbol $w_{idx,j}$
- The contents of register ub_ptr_j points to the register containing tape symbol $w_{ub,j}$
- j indicates the j th prime edge.

The above representation of $\mathcal{E}(1)$ is constructed by the execution of instructions 858 to 1016 of the IDRM program. In other words, in method **8.36**, the expression

Set $\mathcal{E}(1) = \{ E([1], 1), E([2], 1), \dots, E([\mathcal{P}], 1) \}$ is executed by instructions 858 to 1016.

In more detail, instructions 858 to 875 set up an array of m prime edge pointers. Instructions 876 and 877 test if register 0 contains m , the number of prime directed edges. Instructions 878 to 885 initialize registers so that register 33 points to the next prime directed edge pointer, register 155 points to the free heap, and register 99 points to the head node of the edge sequence linked list. Also, register 97 is initialized to 1 which stores the number of distinct prime edges composed together in every element of $\mathcal{E}(1)$ i.e. the 1 in $\mathcal{E}(1)$. Register 96 is initialized to 1 which stores at that time during program execution the number of edge sequences in $\mathcal{E}(k)$. Instructions 886 and 887 test if the program is finished with constructing $\mathcal{E}(1)$.

Instructions 888 to 903 set up the h_node , t_node , and s_node pointers. Instructions 904 to 910 store state r_j . Instructions 911 to 918 store idx_j . Instructions 919 to 926 store ub_j . Instructions 927 to 938 store lb_ptr_j . Instructions 939 to 945 compute idx_ptr_j . Instructions 946 to 951 store idx_ptr_j . Instructions 952 to 957 compute ub_ptr_j . Instructions 958 to 963 store ub_ptr_j . Instructions 964 to 969 store $p_{1,j}$ which equals j . Instructions 970 to 987 store $w_{0,j}, \dots, w_{idx,j}, \dots, w_{ub,j}$. Instructions 988 to 1004 store $(a_{0,j}, a_{0,j}), (a_{1,j}, a_{1,j}), \dots, (a_{N_j,j}, a_{N_j,j})$. Instructions 1005 to 1016 update pointers and registers to jump back to instruction 886 and repeat this loop for the $j+1^{st}$ prime edge that is stored in $\mathcal{E}(1)$.

After $\mathcal{E}(1)$ is constructed, instruction 1018 is the start of the loop where $\mathcal{E}(k+1)$ is constructed from $\mathcal{E}(k)$. Instruction 1018 corresponds to the beginning of the while loop *while* ($\mathcal{E}(k) \neq \emptyset$) in method **8.36**. In particular, if $\mathcal{E}(k) = \emptyset$ is executed by instructions 1021 and 1022. If $\mathcal{E}(k) = \emptyset$, then the IDR program jumps to instruction 1551. After this the program writes “HALT” in registers 0 through 4 using the ASCII code representation. Then the IDR program jumps to instruction 1590 which does not exist and this terminates the execution.

Instruction 1028 corresponds to the beginning of the for loop *for each* $E([p_1, p_2, \dots, p_k], k)$ in $\mathcal{E}(k)$ in method **8.36**. Instruction 1032 corresponds to the beginning of the for loop: *for each prime directed edge* $\mathcal{N}_j \implies \mathcal{T}_j$ in \mathcal{P} in method **8.36**.

In regard to *if* $\mathcal{N}_j \implies \mathcal{T}_j$ link matches with $\mathcal{N}_{p(k)}$ in method **8.36**, instructions 1070 to 1461 test for an overlap match between the current prime edge found and the execution node $\mathcal{N}_{p(k)}$ in the loop starting at instruction 1032.

if $E([p_1, p_2, \dots, p_k, p_{k+1}], k+1)$ contains a consecutive repeating state cycle in method **8.36** is executed in instructions 1475 to 1537 of the IDR program. Instructions 1475 to 1537 look for a consecutive repeating state cycle in $(a_{0,j}, a_{0,j}), (a_{1,j}, a_{1,j}), \dots, (a_{N_j,j}, a_{N_j,j})$

If a consecutive repeating state cycle is found, the program jumps to instruction 1557. Then it writes “IMMORTAL” in registers 0 through 7 using the ASCII code representation. After the program writes the length of one repeat of the consecutive repeating state cycle in register 8. After the program writes the starting register of the first repeat of the state cycle in register 9 and writes the starting register of the second repeat of the state cycle in register 10. Then the IDR program jumps to instruction 1590 which does not exist and this terminates the execution.

Next the representation of $\mathcal{E}(k)$ in the registers of the IDRM program is discussed. Similar to $\mathcal{E}(1)$ the j th element in $\mathcal{E}(k)$ is of the form:

$$[ptr_j, 2N_j, r_j, idx_j, ub_j, lb_ptr_j, idx_ptr_j, ub_ptr_j, p_1, p_2, \dots, p_k \\ w_{0,j}, \dots w_{idx,j} \dots w_{ub,j}, (q_{0,j}, a_{0,j}), (q_{1,j}, a_{1,j}), \dots, (q_{N_j,j}, a_{N_j,j})]$$

- k is the number prime directed edges composed together.
- $2N_j$ is stored in the register following ptr_j where $N_j + 1$ is the number input command pairs corresponding to the edge sequence $E([p_1, p_2, \dots, p_k], k)$ in **8.28**.
- r_j is the state of the execution node $\mathcal{N}_{p(k)}$ as defined in **8.26**, **8.27**, and **8.28**.
- idx_j contains the tape head index of the execution node $\mathcal{N}_{p(k)}$
- ub_j contains the upper bound of the tape of the execution node $\mathcal{N}_{p(k)}$
- The contents of register lb_ptr_j points to the register containing tape symbol $w_{0,j}$
- The contents of register idx_ptr_j points to the register containing tape symbol $w_{idx,j}$
- The contents of register ub_ptr_j points to the register containing tape symbol $w_{ub,j}$
- p_1, p_2, \dots, p_k records the sequence of prime directed edges that were link matched. Each p_j is determined by its `pe_number` stored in the prime edge linked list.
- $w_{0,j} \dots w_{idx,j} \dots w_{ub,j}$ represents the tape of the execution node and the tape head points to alphabet symbol $w_{idx,j}$
- $(q_{0,j}, a_{0,j}), (q_{1,j}, a_{1,j}), \dots, (q_{N_j,j}, a_{N_j,j})$ is the composition of the prime input command sequences corresponding to the prime directed edges p_1, p_2, \dots, p_k that were link matched as described in definition **8.4**.