Turing Incomputable Computation

Michael Stephen Fiske
Aemea Institute, San Francisco, California, U.S.A.
mf@aemea.org

Abstract

A new computing model, called the active element machine (AEM), is presented that demonstrates Turing incomputable computation using quantum random input. The AEM deterministically executes a universal Turing machine (UTM) program \( \eta \) with random active element firing patterns. These firing patterns are Turing incomputable when the AEM executes a UTM having an unbounded number of computable steps. For an unbounded number of computable steps, if zero information is revealed to an adversary about the AEM’s representation of the UTM’s state and tape and the quantum random bits that help determine \( \eta \)'s computation and zero information is revealed about the dynamic connections between the active elements, then there does not exist a “reverse engineer” Turing machine that can map the random firing patterns back to the sequence of UTM instructions. This casts a new light on Turing’s notion of a computational procedure. In practical terms, these methods present an opportunity to build a new class of computing machines where the program’s computational steps are hidden. This non-Turing computing behavior may be useful in cybersecurity and in other areas such as machine learning where multiple, dynamic interpretations of firing patterns may be applicable.

1 Introduction

Recent cyberattacks have demonstrated that current approaches to the malware problem (e.g., detection) are inadequate. This is not surprising as malware detection is Turing undecidable [6]. Further, some recent malware implementations use NP problems to encrypt and hide the malware [17]. Two goals guide an alternative approach: (a) Program execution should hide computational steps in order to hinder “reverse engineering” efforts by malware hackers; (b) New computational models should be explored that make it more difficult to hijack the purpose of program execution. The methods explained here pertain to (a).

Turing incomputable ([4], [7], [8]) computation is created using quantum random input. A new computing machine, called the active element machine (AEM), is presented that uses quantum randomness ([42], [43]) to deterministically execute a universal Turing machine (UTM) program \( \eta \) (table 2) with random firing interpretations. The active element machine computing model was introduced in [18], is summarized in section 2 and is further described in subsection 7.2 of the appendix. Furthermore, the active element firing patterns – computing the execution of the UTM program – are Turing incomputable when the UTM computation is non-halting or when the UTM computation is executed repeatedly on an unbounded number of computable initial configurations that halt.

The incomputability of the firing patterns depends on two quantum randomness axioms.

1. **No bias.** A single outcome \( x_i \) of a bit sequence \( (x_1x_2\ldots) \) generated by quantum randomness is unbiased: \( P(x_i = 1) = P(x_i = 0) = \frac{1}{2} \).

2. **History has no effect on the next event.** Each outcome \( x_i \) is independent of the history. No correlation exists between previous or future outcomes. \( P(x_i = 1 \mid x_1 = b_1, \ldots, x_{i-1} = b_{i-1}) = \frac{1}{2} \) and \( P(x_i = 0 \mid x_1 = b_1, \ldots, x_{i-1} = b_{i-1}) = \frac{1}{2} \) for each \( b_i \in \{0, 1\} \).
Let $\Omega = \{(b_1b_2\ldots) : b_i \in \{0,1\}\}$ be the space of infinite sequences of 0’s and 1’s representing infinite quantum random bit sequences. From these two axioms, in [4] Calude and Szovil conclude that if a quantum system producing the quantum randomness runs under ideal conditions to infinity, then the resulting infinite sequence of 0’s and 1’s (i.e., sequence in $\Omega$) is Turing incomputable. In other words, no Turing machine can exactly reproduce this infinite sequence of 0’s and 1’s. In this context, no Turing machine can reproduce an unbounded number of the aforementioned active element firing patterns.

The following definition helps clarify the one-way nature of the random AEM firing patterns.

**Definition 1.** Turing machine map back. Let $g : \mathbb{N} \to \{0,1\}$ and $f : \mathbb{N} \to \{0,1,\ldots,m\}$ be functions. A Turing machine maps $g$ back to function $f$ if conditions A and B hold.

A) The initial Turing machine tape has $g(k)$ stored on tape square $k$ for each $k$.

B) The Turing machine begins execution at tape square 1 and after a finite number of steps writes $f(1)$ on tape square 1. For each $k$, the Turing machine visits tape square $k$ and after a finite number of steps, it writes $f(k)$ on tape square $k$ and then moves to tape square $k+1$.

If zero information is revealed to an adversary about the AEM’s representation of the UTM’s state and tape and the quantum random bits that help determine $\eta$’s computation and zero information is revealed about the dynamic connections between the active elements, then there does not exist a “reverse engineer” Turing machine that can map the random firing patterns back to the sequence of UTM instructions when the UTM executes an unbounded number of computable steps. Overall, using quantum randomness, a finite active element machine can deterministically execute any Turing machine with active element firing patterns that are Turing incomputable. This casts a new light on the notion of a computational procedure ([5], [45]). In [29], Lewis and Papadimitriou discuss the Church-Turing notion of an algorithm (computational procedure):

The principle that Turing machines are formal versions of algorithms and that no computational procedure will be considered as 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.

The works of [14], [22], [23], [27] and [40] describe methods of hypercomputation that currently have no physical realization. In [11] Davis provides counterarguments to the physical realizability of hypercomputation. In [10] he calls hypercomputation “a myth” and appears to dismiss random number generators as a means to enhance the computing power of Turing machines.

The computing power of Turing machines provided with a random number generator was studied in the classic paper [12]. It turned out that such machines could compute only functions that are already computable by ordinary Turing machines.

In [12], de Leeuw et al. explain that their results depend on the definitions of the machines chosen and the tasks these machines perform.

The following question will be considered in this paper: Is there anything that can be done by a machine with a random element but not by a deterministic machine?

The question as it stands is, of course, too vague to be amenable to mathematical analysis. In what follows, it must be delimited in two respects. A precise definition of the class of machine to be considered must be given and an equally precise definition must be given of the tasks which they are to perform. It is clear that the nature of our results will depend strongly on these two choices and therefore our answer is not to be interpreted as a complete solution of the originally posed informal question.
What are relevant differences? In [12], the Turing machine reads a random infinite sequence of 0’s and 1’s off the tape – called a probabilistic machine – but the Turing machine program does not change as a consequence of this random input. (A definition of Turing machine program is in section 7.1 of the appendix.) In this paper, during AEM program execution, the Meta command changes the AEM architecture (program) based on the quantum random input, which changes the AEM program’s computing behavior. In particular, at each computational step of the UTM, the random input helps determine the element parameters and connection parameters of the active elements that collectively compute $\eta$; and the Meta command helps dynamically change these parameters based on the random input.

Another AEM example with random input also illustrates the differences. In [18], section 7 describes a method – using a finite AEM program, a quantum random input and the Meta command – for recognizing a non-Turing binary language $L \subset \{0,1\}^*$. After the AEM has received enough random input, then the AEM can determine whether, for example, if the string $011001$ is in the binary language $L$ recognized by the AEM. In other words, the random input helps determine the binary language $L$ and the Meta command helps dynamically build the appropriate AEM that recognizes $L$.

A physical realization of the methods shown here can be implemented using a quantum random generator device [43] with a USB plug connected to a laptop computer executing a finite active element machine program. It is important to note that this physical realization of incomputable firing patterns is a scientific thesis – not a mathematical proof, as this realization depends on quantum theory.

In a cryptographic system, Claude Shannon [38] defines the notion of perfect secrecy.

Perfect secrecy is defined by requiring of a system that after a cryptogram is intercepted by the enemy the a posteriori probabilities of this cryptogram representing various messages be identically the same as the a priori probabilities of the same messages before the interception.

Perfect secrecy here means that zero information is released about the state and the tape contents of the universal Turing machine, the quantum random bits that help determine how $\eta$ is computed and the dynamic connections of the active element machine. Formally, let $f_{1,j}, f_{2,j}, \ldots, f_{m,j}$ represent the random firing pattern computing $\eta$ during the $j$th computational step and assume an adversary can only eavesdrop on $f_{1,j}, f_{2,j}, \ldots, f_{m,j}$. Let $q$ denote the current state of the UTM, $a_k$ a UTM alphabet symbol and $q_k$ a UTM state. Perfect secrecy means that probabilities $P(q = q_k | f_{1,j} = b_1, \ldots, f_{m,j} = b_m) = P(q = q_k)$ and $P(T_k = a_k | f_{1,j} = b_1, \ldots, f_{m,j} = b_m) = P(T_k = a_k)$ for each $b_i \in \{0,1\}$ and each $T_k$ which represents the contents of the $k$th tape square.

In [28], Kocher et al. present differential power analysis. Differential power analysis obtains information about cryptographic computations executed by register machine hardware, by statistically analyzing the electromagnetic radiation leaked by the hardware during its computation. If a quantum active element computing system is built so that its internal components remain close to perfectly secret, then it could be extremely challenging for an adversary to carry out cyberattacks such as differential power analysis.

1.1 Brief Summary of Prior Computing Models

In [45], the Turing Machine is introduced. A brief review is in the appendix. In [44], Sturgis and Shepherdson show the computational equivalence of the register machine. The works [9], [29], [30], [31], [33] and [41] cover computability theory. Alternative models influenced by neurophysiology are discussed by McCulloch and Pitts in [32], Rosenblatt in [37], Minsky and Papert in [34], Rall in [35], Hertz et al. in [21] and Hopfield in [24], [26] and [25]. In [20] Halang
Turing Incomputable Computation

Michael Fiske

et al. describe some advantages of using time. Important work on quantum computing models is presented in [1], [2], [13], [15], [16], [19], [30], [31], and [39].

2 An Informal Summary of the Active Element Machine

A formal introduction to the active element machine is in section 7.2. An AEM is composed of computational primitives called active elements. There are three kinds of active elements: Input, Computational and Output active elements. Input active elements process information received from the environment or another active element machine. Computational active elements receive pulses from the input active elements and other computational active elements and transmit new pulses to computational and output active elements. The output active elements receive pulses from the input and computational active elements. Every active element is active in the sense that each one can receive and transmit pulses simultaneously.

Each pulse has an amplitude and a width, representing how long the pulse amplitude lasts as input to the active element receiving the pulse. If active element \( E_i \) simultaneously receives pulses with amplitudes summing to a value greater than \( E_i \)'s threshold and \( E_i \)'s refractory period has expired, then \( E_i \) fires. When \( E_i \) fires, it sends pulses to other active elements. If \( E_i \) fires at time \( t \), a pulse reaches element \( E_k \) at time \( t + \tau_{ik} \) where \( \tau_{ik} \) is the transmission time from element \( E_i \) to \( E_k \).

The AEM programming language has 5 commands and 2 special keywords. (See 7.3) Consider element command \((\text{Element} (\text{Time } 2) (\text{Name } L) (\text{Threshold } -3) (\text{Refractory } 2) (\text{Last } 0))\). At time 2, if active element \( L \) does not exist, then it is created. Element \( L \) has its threshold set to \(-3\), its refractory period set to 2, and its last time fired set to 0. After time 2, element \( L \) exists indefinitely with threshold = \(-3\), refractory = 2 until a new element command whose name value \( L \) is executed at a later time; in this case, the parameter values specified in the new command are updated.

A connection command sets the pulse parameters between two elements. Consider command \((\text{Connection} (\text{Time } 2) (\text{From } C) (\text{To } L) (\text{Amp } -7) (\text{Width } 6) (\text{Delay } 3))\). At time 2, the connection from active element \( C \) to active element \( L \) has its amplitude set to \(-7\), its pulse width set to 6, and transmission time set to 3. A Fire command fires an input active element in order to communicate program input to the AEM. \((\text{Fire} (\text{Time } 3) (\text{Name } C))\) causes element \( C \) to fire at time \( t = 3 \).

A Program command enables one command to execute multiple commands. The execution of \((Q (\text{Args } 0 \ E \ L))\) based on the following program definition

\[
\begin{align*}
\text{(Program} & \ Q \ (\text{Args} \ t \ x \ y) \\
& (\text{Element} (\text{Time} \ t) (\text{Name} \ x) (\text{Refractory} \ 7) (\text{Threshold} \ 8) (\text{Last } 0)) \\
& (\text{Connection} (\text{Time} \ t) (\text{From} \ x) (\text{To} \ y) (\text{Amp} \ 5) (\text{Width} \ 3) (\text{Delay} \ 4))
\end{align*}
\]

creates element \( E \) and a connection from \( E \) to \( L \) at time \( t = 0 \).

The Meta command \((\text{Meta} (\text{Name} \ E) (\text{Window} \ 1 \ 5) (\text{C} (\text{Args} \ a \ b)))\) causes command \( C \) to execute with arguments \( a \) and \( b \) each time that element \( E \) fires during the time window \([1, 5]\). The keyword \( dT \) denotes an infinitesimal amount of time that helps coordinate almost simultaneous events. \( dT > 0 \) and \( dT \) is less than every positive rational number. The keyword \( \text{clock} \) evaluates to an integer, which is the current time of the AEM clock.
3 AEM Interpretations of Boolean Functions

In this section, the same boolean function is computed by two or more distinct active element firing patterns, which can be executed at two distinct times or by two different circuits in the AEM. The use of level sets helps design distinct AEM firing patterns that can compute the same boolean function. These firing patterns can be generated using quantum randomness.

The following procedure uses a finite active element program and a quantum system to either fire input element I or not fire I at time \( t = n \) where \( n \) is a natural number \( \{0, 1, 2, 3, \ldots\} \). This random sequence of 0’s and 1’s can be generated by quantum systems discussed in [3], [42] or [43].

Procedure 1. *Randomness generates an AEM, representing a real number in \([0, 1]\)*

A random sequence of bits creates active elements named \( 0, 1, 2, \ldots \) that store the binary representation \( b_0 b_1 b_2 \ldots \) of real number \( x \in [0, 1] \). If input element I fires at time \( t = n \), then \( b_n = 1 \); thus, create active element \( n \) so that after \( t = n \), element \( n \) fires every unit of time indefinitely. If input element I doesn’t fire at time \( t = n \), then \( b_n = 0 \) and active element \( n \) is created so that it never fires. The following program *real* exhibits this behavior.

\[
\text{(Program real (Args t)}
\begin{align*}
\text{(&Connection (Time t) (From I) (To t) (Amp 2) (Width 1) (Delay 1))}
\text{(&Connection (Time t+dT) (From I) (To t) (Amp 0))}
\text{(&Connection (Time t) (From t) (To t) (Amp 2) (Width 1) (Delay 1))}
\text{)}
\end{align*}
\text{(Element (Time clock) (Name clock) (Threshold 1) (Refractory 1) (Last -1))}
\text{(Meta (Name I) (real (Args clock)))}
\]

Suppose the sequence of random bits begins with 1, 0, 1, \ldots . Thus, input element I fires at times 0, 2, \ldots . At time 0, the following commands are executed.

\[
\text{(Element (Time 0) (Name 0) (Threshold 1) (Refractory 1) (Last -1))}
\text{(real (Args 0))}
\]

The execution of \( \text{(real (Args 0))} \) causes three connection commands to execute.

\[
\begin{align*}
\text{(&Connection (Time 0) (From I) (To 0) (Amp 2) (Width 1) (Delay 1))}
\text{(&Connection (Time 1+dT) (From I) (To 0) (Amp 0))}
\text{(&Connection (Time 0) (From 0) (To 0) (Amp 2) (Width 1) (Delay 1))}
\end{align*}
\]

At time 0, input element I sends a pulse with amplitude 2 to element 0. Thus, element 0 fires at time 1. At time 1+dT, a moment after time 1, the connection from input element I to element 0 is removed. At time 0, a connection from element 0 to itself with amplitude 2 is created. Element 0 continues to fire indefinitely, indicating that \( b_0 = 1 \).

\[
\text{(Element (Time 1) (Name 1) (Threshold 1) (Refractory 1) (Last -1))}
\]

Since element 1 has no connections into it and threshold 1, it never fires. Thus \( b_1 = 0 \).

3.1 Active Element Firing Patterns

This subsection explains how a firing pattern can be used to compute a boolean function \( f : \{0, 1\}^2 \to \{0, 1\} \). These methods can be extended to \( f : \{0, 1\}^n \to \{0, 1\} \). In the next section, the level set methods described here are combined with procedure 4 so that a quantum random firing pattern computes the boolean functions representing the execution of universal Turing machine program \( \eta \).

Active elements \( X_0, X_1, X_2 \) and \( X_3 \) are designed so that each \( X_i \) either fires or doesn’t fire during window \( W = [a, b] \). After one of the 16 firing patterns is randomly generated, level sets \( f^{-1}\{0\} \) and \( f^{-1}\{1\} \) are used to compute \( f : \{0, 1\}^2 \to \{0, 1\} \). Element \( P \) represents the output of \( f \). The goal is for \( P \) to fire during window \( W \) if and only if \( P \) receives a unique firing pattern from elements \( X_0, X_1, X_2 \) and \( X_3 \). This goal motivates the following definition.
Definition 2. Number of Firings during a Window

Let $X$ denote the set of active elements $\{X_0, X_1, \ldots, X_{n-1}\}$ that determine a firing pattern during window of time $W$. Then $|(X_k, W)|$ is the number of times that element $X_k$ fired during $W$. Thus, define the number of firings during window $W$ as $|(X, W)| = \sum_{k=0}^{n-1} |(X_k, W)|$.

Observe that $|(X, W)| = 0$ for firing pattern 0000 and $|(X, W)| = 2$ for firing pattern 0101.

To isolate a firing pattern so that element $P$ only fires if this unique firing pattern occurs, set the threshold of element $P$ = $2 \cdot |(X, W)| - 1$. The element command for $P$ is

$$(\text{Element} (\text{Time } a-dT) (\text{Name } P) (\text{Threshold } 2 \cdot |(X, W)| - 1) (\text{Refractory } b-a) (\text{Last } 2a-b))$$

Each connection from $X_k$ to $P$ is based on whether $X_k$ is supposed to fire or not fire during $W$. If $X_k$ is supposed to fire during $W$, the following connection is established.

$$(\text{Connection} (\text{Time } a-dT) (\text{From } X_k) (\text{To } P) (\text{Amp } 2) (\text{Width } b-a) (\text{Delay } 1))$$

If $X_k$ is not supposed to fire during $W$, then the following connection is established.

$$(\text{Connection} (\text{Time } a-dT) (\text{From } X_k) (\text{To } P) (\text{Amp } -2) (\text{Width } b-a) (\text{Delay } 1))$$

The firing pattern is already known because it is determined based on a random source of bits received by input elements, as explained in procedure 1.

Example 1. Computing $\oplus$ with Firing Pattern 0010

Firing pattern 0010 computes exclusive-or $A \oplus B = (A \lor B) \land (\neg A \lor \neg B)$.

An AEM program is designed such that $A \oplus B = 1$ if and only if the firing pattern for $X_0, X_1, X_2, X_3$ is 0010. If $A \oplus B = 1$ then $P$ fires. If $A \oplus B = 0$ then $P$ doesn’t fire. Choose window $W = [2, 3]$. The following commands connect elements $A$ and $B$ to elements $X_0, X_1, X_2, X_3$.

$$(\text{Connection} (\text{Time } 0) (\text{From } A) (\text{To } X_0) (\text{Amp } 2) (\text{Width } 2) (\text{Delay } 2))$$

$$(\text{Connection} (\text{Time } 0) (\text{From } B) (\text{To } X_0) (\text{Amp } 2) (\text{Width } 2) (\text{Delay } 2))$$

$$(\text{Element} (\text{Time } 0) (\text{Name } X_0) (\text{Threshold } 3) (\text{Refractory } 1) (\text{Last } 1))$$

There are four cases for $A \oplus B$ shown below.

1. 1 $\oplus$ 0. Element $A$ fires at time $t = 0$ and element $B$ doesn’t fire at $t = 0$.
2. 0 $\oplus$ 1. Elements $A$ doesn’t fire at $t = 0$ and element $B$ fires at time $t = 0$.
3. 1 $\oplus$ 1. Element $A$ fires at time $t = 0$ and element $B$ fires at $t = 0$.
4. 0 $\oplus$ 0. Elements $A$ and $B$ both don’t fire at $t = 0$.

To isolate firing pattern 0010, set the threshold of $P$ to $2 \cdot |(X, W)| - 1 = 1$. The element command for $P$ is

$$(\text{Element} (\text{Time } 2-dT) (\text{Name } P) (\text{Threshold } 1) (\text{Refractory } 1) (\text{Last } 1))$$

To make $P$ fire if and only if firing pattern 0010 occurs during $W$, use the commands
For cases 1 and 2, \((1 \oplus 0 \text{ and } 0 \oplus 1)\) only \(X_2\) fires. A moment before \(X_2\) fires at \(t = 2\) (i.e., \(−dT\)), the amplitude from \(X_2\) to \(P\) is set to 2. At time \(t = 2\), a pulse with amplitude 2 is sent from \(X_2\) to \(P\), so \(P\) fires at time \(t = 3\) since its threshold = 1. In other words, \(1 \oplus 0 = 1\) or \(0 \oplus 1 = 1\) has been computed. For case 3, \((1 \oplus 1)\), \(X_0\), \(X_2\) and \(X_3\) fire. Thus, two pulses each with amplitude = -2 are sent from \(X_0\) and \(X_3\) to \(P\). And one pulse with amplitude 2 is sent from \(X_2\) to \(P\). Thus, \(P\) doesn’t fire. In other words, \(1 \oplus 1 = 0\) has been computed. For case 4, \((0 \oplus 0)\), \(X_1\) fires. One pulse with amplitude = -2 is sent to \(X_2\). Thus, \(P\) doesn’t fire. In other words, \(0 \oplus 0 = 0\) has been computed.

As shown in table 1, any of the sixteen boolean functions can be mapped to one of the sixteen firing patterns by an appropriate AEM program using level sets to separate elements of the domain \(
\{(0,0), (1,0), (0,1), (1,1)\}
\)
.

Each active element \(X_k\) in the firing pattern separates these members based on the (amplitude from \(A\) to \(X_k\), amplitude from \(B\) to \(X_k\), threshold of \(X_k\), element \(X_k\)) quadruplet. For example, the quadruplet \((0,2,1,1)\) separates \((1,1),(0,1)\) from \((1,0),(0,0)\) with respect to \(X_1\). Recall that \(A = 1\) means \(A\) fires and \(B = 1\) means \(B\) fires. Then \(X_1\) will fire with inputs \((1,1),(0,1)\) and \(X_1\) will not fire with inputs \((1,0),(0,0)\). The separation rule is expressed as \((0,2,1,1) \leftrightarrow \{(1,1),(0,1)\}\) \((1,0),(0,0)\). The separation rule \((0,−2,−1,1) 
\leftrightarrow \{(1,0),(0,0)\}\) \((1,1),(1,0)\) indicates that \(X_2\) has threshold \(-1\) and amplitudes \(0\) and \(-2\) from \(A\) and \(B\) respectively. Further, \(X_2\) will fire with inputs \((1,0),(0,0)\) and will not fire with inputs \((1,1),(0,1)\).

<table>
<thead>
<tr>
<th>Boolean Function</th>
<th>AEM Separation Rule(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1(A, B) = 1)</td>
<td>((0,0,−1,X_k) \leftrightarrow {(0,0),(1,0),(0,1),(1,1)})</td>
</tr>
<tr>
<td>(f_2(A, B) = 0)</td>
<td>((0,0,1,X_k) \leftrightarrow 0)</td>
</tr>
<tr>
<td>(f_3(A, B) = A)</td>
<td>((2,0,1,X_k) \leftrightarrow {(1,1),(1,0)})</td>
</tr>
<tr>
<td>(f_4(A, B) = B)</td>
<td>((0,2,1,X_k) \leftrightarrow {(1,1),(0,1)})</td>
</tr>
<tr>
<td>(f_5(A, B) = \neg A)</td>
<td>((-2,0,−1,X_k) \leftrightarrow {(0,1),(0,0)})</td>
</tr>
<tr>
<td>(f_6(A, B) = \neg B)</td>
<td>((0,0,−1,X_k) \leftrightarrow {(0,0),(1,0)})</td>
</tr>
<tr>
<td>(f_7(A, B) = A \wedge B)</td>
<td>((2,2,3,X_k) \leftrightarrow {(0,0),(1,1)})</td>
</tr>
<tr>
<td>(f_8(A, B) = A \lor B)</td>
<td>((2,2,1,X_k) \leftrightarrow {(1,0),(0,1)})</td>
</tr>
<tr>
<td>(f_9(A, B) = A \rightarrow B)</td>
<td>((-4,2,−3,X_k) \leftrightarrow {(0,0),(0,1),(1,1)})</td>
</tr>
<tr>
<td>(f_{10}(A, B) = A \leftarrow B)</td>
<td>((2,−4,−3,X_k) \leftrightarrow {(0,0),(1,0),(1,1)})</td>
</tr>
<tr>
<td>(f_{11}(A, B) = A \leftrightarrow B)</td>
<td>((2,−4,−3,X_k)) and ((-4,2,−3,X_j)) with (j \neq k)</td>
</tr>
</tbody>
</table>
4 Random Firing Interpretations Execute a UTM

A universal Turing Machine (UTM) is a Turing machine that can mimic the computation of any Turing Machine by reading the other Turing Machine’s description and input from the UTM’s tape. Table 3 in the appendix shows Minsky’s universal Turing machine described in [33]. A boolean representation of Minsky’s UTM is shown in table 2.

The elements of \(\{0,1\}^2\) are denoted as \(A = \{00, 01, 10, 11\}\). The tape symbols in Minsky’s UTM alphabet correspond to elements in \(A\) as follows: 0 ↔ 00, 1 ↔ 01, \(y\) ↔ 10 and \(A\) ↔ 11. The states of Minsky’s UTM correspond to the elements of \(Q\) as follows: \(q_1\) ↔ 001, \(q_2\) ↔ 010, \(q_3\) ↔ 011, \(q_4\) ↔ 100, \(q_5\) ↔ 101, \(q_6\) ↔ 110, \(q_7\) ↔ 111 and the halting state \(H\) ↔ 000. In regard to tape head moves, \(L\) ↔ 0 and \(R\) ↔ 1 in \(\{0,1\}\). Symbol \(h\) indicates that the tape head does not move, which occurs when the UTM has halted.

An active element machine is designed to compute the universal Turing Machine program \(\eta\) shown in table 2. Following the methods in [34] multiple AEM firing interpretations are created that compute \(\eta\). The three boolean variables \(U, W, X\) are concatenated to represent the current state of the UTM. The two boolean variables \(Y, Z\) represent the current tape symbol. Observe that \(\eta = (\eta_0, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5)\). The level sets of \(\eta_3\) are shown below.

\[
\eta_3^{-1}(UWX,YZ)\{1\} = \{(111,00), (110,00), (110,01), (110,10), (101,00), (101,01), (101,10), (100,00), (100,01), (011,01), (011,10), (010,11), (010,01), (010,00)\}
\]

\[
\eta_3^{-1}(UWX,YZ)\{0\} = \{(111,01), (111,10), (111,11), (110,11), (101,11), (100,01), (100,11), (011,00), (011,11), (010,10), (001,00), (001,01), (001,10), (001,11), (000,01), (000,10), (000,11), (000,00)\}
\]

<table>
<thead>
<tr>
<th>Boolean Function</th>
<th>AEM Separation Rule(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_{12}(A,B) = \neg(A \lor B))</td>
<td>((-2, -2, -1, X_k) \leftrightarrow {(0,0)} \cup {(1,0), (0,1), (1,1)})</td>
</tr>
<tr>
<td>(f_{13}(A,B) = \neg(A \land B))</td>
<td>((-2, -2, -3, X_k) \leftrightarrow {(1,0), (0,1), (0,0)} \cup {(1,1)})</td>
</tr>
<tr>
<td>(f_{14}(A,B) = A \oplus B)</td>
<td>((2, 2, 1, X_k) \text{ and } (-2, -2, -3, X_j) \text{ with } j \neq k)</td>
</tr>
<tr>
<td>(f_{15}(A,B) = A &lt; B)</td>
<td>((-2, 4, 3, X_k) \leftrightarrow {(0,1)} \cup {(0,0), (1,0), (1,1)})</td>
</tr>
<tr>
<td>(f_{16}(A,B) = A &gt; B)</td>
<td>((4, -2, 3, X_k) \leftrightarrow {(1,0)} \cup {(0,0), (0,1), (1,1)})</td>
</tr>
</tbody>
</table>

For each \(X_j\), use one of the separation rules to map the level set \(f_k^{-1}\{1\}\) or alternatively map the level set \(f_k^{-1}\{0\}\) to one of the sixteen firing patterns represented by \(X_0, X_1, X_2, X_3\).

Table 2: Boolean Universal Turing Machine program \(\eta = (\eta_0, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5)\)

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>00</th>
<th>01</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>(001, 00, 0)</td>
<td>(001, 00, 0)</td>
<td>(010, 01, 0)</td>
<td>(001, 01, 0)</td>
</tr>
<tr>
<td>010</td>
<td>(001, 00, 0)</td>
<td>(010, 10, 1)</td>
<td>(010, 11, 1)</td>
<td>(110, 10, 1)</td>
</tr>
<tr>
<td>011</td>
<td>(011, 10, 0)</td>
<td>(000, 00, h)</td>
<td>(011, 11, 0)</td>
<td>(100, 01, 0)</td>
</tr>
<tr>
<td>100</td>
<td>(010, 10, 0)</td>
<td>(101, 10, 1)</td>
<td>(111, 01, 0)</td>
<td>(100, 01, 0)</td>
</tr>
<tr>
<td>101</td>
<td>(010, 10, 1)</td>
<td>(011, 10, 0)</td>
<td>(011, 11, 1)</td>
<td>(101, 01, 1)</td>
</tr>
<tr>
<td>110</td>
<td>(110, 10, 1)</td>
<td>(011, 11, 0)</td>
<td>(110, 11, 1)</td>
<td>(110, 01, 1)</td>
</tr>
<tr>
<td>111</td>
<td>(111, 00, 1)</td>
<td>(110, 10, 1)</td>
<td>(111, 01, 1)</td>
<td>(010, 00, 1)</td>
</tr>
</tbody>
</table>

State set \(Q = \{001, 010, 011, 100, 101, 110, 111\}\). Alphabet \(A = \{10, 00, 01, 11\}\).
The level sets of $\pi_k : \{0,1\}^3 \times \{0,1\}^2 \to \{0,1\}$ where $k \in \{0,1,2,4,5\}$ are shown below.

$\pi_0^{-1}(UWX,YZ)\{1\} = \{(111,10),(111,01),(111,00),(110,11),(110,10),(101,01),(101,00),(100,11),(100,10),(100,01),(100,00),(011,11),(011,10),(011,01),(011,00),(010,10),(010,01),(010,00),(001,11),(001,10),(001,01),(001,00),(000,11),(000,10),(000,01),(000,00)\}$

$\pi_2^{-1}(UWX,YZ)\{1\} = \{(111,10),(111,01),(111,00),(110,11),(110,10),(101,01),(101,00),(100,11),(100,10),(100,01),(011,11),(011,10),(011,01),(011,00),(010,10),(010,01),(010,00),(001,11),(001,10),(001,01),(001,00)\}$

$\pi_4^{-1}(UWX,YZ)\{1\} = \{(111,10),(111,01),(111,00),(110,11),(110,10),(101,01),(101,00),(100,11),(100,10),(100,01),(011,11),(011,10),(011,01),(011,00),(010,10),(010,01),(010,00),(001,11),(001,10),(001,01),(001,00)\}$

$\pi_6^{-1}(UWX,YZ)\{1\} = \{(111,10),(111,01),(111,00),(110,11),(110,10),(101,01),(101,00),(100,11),(100,10),(100,01),(011,11),(011,10),(011,01),(011,00),(010,10),(010,01),(010,00),(001,11),(001,10),(001,01),(001,00)\}$

$\pi_8^{-1}(UWX,YZ)\{1\} = \{(111,10),(111,01),(111,00),(110,11),(110,10),(101,01),(101,00),(100,11),(100,10),(100,01),(011,11),(011,10),(011,01),(011,00),(010,10),(010,01),(010,00),(001,11),(001,10),(001,01),(001,00)\}$

The level set $\pi_5^{-1}(UWX,YZ)\{h\} = \{(011,00)\}$ is the special case when the UTM halts (i.e., $\pi(011,00) = (000,0,0,h)$). When the UTM halts, the AEM reaches a halting firing pattern $H$.

The next example copies one element’s firing state to another element’s firing state. This program helps assign the value of a random bit to an active element and perform other functions in the UTM. When the following copy program is called, active element $b$ fires if $a$ fired during the window of time $[s, t]$. Further, a connection is set up from $b$ to $b$ so that $b$ will keep firing indefinitely. This enables $b$ to store active element $a$’s firing state.

**Example 2.** Copy Program

An AEM program copies active element $a$’s firing state to element $b$.

(Program copy (Args $s \ t \ b \ a$)

(Element (Time $s-1$) (Name $b$) (Threshold 1) (Refractory 1) (Last $s-1$))

(Connection (Time $s-1$( (From $a$) (To $b$) (Amp 0) (Width 0) (Delay 1)))

(Connection (Time $s$ (From $a$) (To $b$) (Amp 2) (Width 1) (Delay 1)))

(Connection (Time $t$ (From $a$) (To $b$) (Amp 2) (Width 1) (Delay 1)))


Procedure 2. Computing Turing Program $\pi$ with Random Firing Patterns

Procedure 2 describes the computation of $\pi$ with random AEM firing patterns.

Consider function $\pi_3 : \{0,1\}^5 \to \{0,1\}$. The following scheme is used to represent boolean values 1 and 0 with the firing of active elements. If active element $U$ fires during window $W$, then this corresponds to input $U = 1$ in $\pi_3$; if active element $U$ doesn’t fire during window $W$, then this corresponds to input $U = 0$ in $\pi_3$. When $U$ fires, $W$ doesn’t fire, $X$ fires, $Y$ doesn’t fire and $Z$ doesn’t fire, this corresponds to computing $\pi_3(101,00)$. The value $1 = \pi_3(101,00)$ is the underlined bit in $(011,10,0)$, which is located in row 101, column 00 of table 2.

Procedure 2 and the separation rules in table 3 are synthesized so that $\pi_3$ is computed using a dynamic interpretation determined by quantum random bits. This creates random active element firing patterns that compute $\pi_3$. 

74
The firing activity of element $P_3$ represents the output value of $\gamma_3(UWX,YZ)$. Fourteen random bits are created from quantum randomness (See [42], [43]). These random bits create a corresponding random firing pattern of active elements $R_0, R_1, \ldots, R_{13}$. Meta commands dynamically build active elements and connections based on the separation rules in table 3 and the firing activity of elements $R_0, R_1, \ldots, R_{13}$. These dynamically created elements and connections determine the firing activity of element $P_3$ based on the firing activity of elements $U, W, X, Y$ and $Z$. This procedure is described below.

### Table 3: AEM Separation Rules for Level Set $\gamma_3^{-1}\{1\}$

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>$(U,D_3)$</th>
<th>$(W,D_3)$</th>
<th>$(X,D_3)$</th>
<th>$(Y,D_3)$</th>
<th>$(Z,D_3)$</th>
<th>$\theta_{D_3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 00</td>
<td>$D_0$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 00</td>
<td>$D_1$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>110 01</td>
<td>$D_2$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>110 10</td>
<td>$D_3$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>101 00</td>
<td>$D_4$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>101 01</td>
<td>$D_5$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>101 10</td>
<td>$D_6$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>100 00</td>
<td>$D_7$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>100 10</td>
<td>$D_8$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>011 01</td>
<td>$D_9$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>011 10</td>
<td>$D_{10}$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>010 00</td>
<td>$D_{11}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>010 01</td>
<td>$D_{12}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>010 11</td>
<td>$D_{13}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

**Step 2.1** A quantum source creates fourteen random bits $a_0, a_1, \ldots, a_{13}$. These bit values are stored in elements $R_0, R_1, \ldots, R_{13}$. If $a_k = 1$, then $R_k$ fires; if $a_k = 0$, $R_k$ doesn’t fire.

**Step 2.2** Set up dynamical connections from active elements $U, X, W, Y, Z$ to elements $D_0, D_1, \ldots, D_{13}$, which depend on meta commands that use the firing pattern from elements $R_0, R_1, \ldots, R_{13}$.

For $D_0$, look at the first row and first column of table 3. The firing pattern 111 00 means that $D_0$ dynamically separates this pattern with respect to the other firing patterns of $UWXYZ$. The separation is dynamic based on whether element $R_0$ fires or doesn’t fire. If $R_0$ fires, then $D_0$ fires when the firing pattern for $UWXYZ$ is 111 00; for all other firing patterns for $UWXYZ$, then $D_0$ doesn’t fire. If $R_0$ doesn’t fire, then $D_0$ doesn’t fire when the firing pattern for $UWXYZ$ is 111 00; for all other firing patterns for $UWXYZ$, then $D_0$ fires. In a similar way, $D_1$ dynamically separates the firing pattern 110 00 for $UWXYZ$ based on whether element $R_1$ fires. Observe that every firing pattern in the first column is in the level set of $\gamma_3^{-1}\{1\}$.

The firing pattern 111 00 corresponds to the $\gamma$ instruction (111, 00), whose output (110, 10, 1) is shown in the last row and second column of table 3. The amplitudes from $U, W, X, Y, Z$ to $D_0$ are labeled by the headers $(U, D_3)$, $(W, D_3)$, $(X, D_3)$, $(Y, D_3)$ and $(Z, D_3)$, respectively. $D_0$’s threshold 5 is in the first row under the header $\theta_{D_3}$.

```plaintext
(Program set_dynamic_C (Args s t f xk a w tau rk)
  (Connection (Time s-dT) (From f) (To xk) (Amp -a) (Width w) (Delay tau))
  (Meta (Name rk) (Window s t)
    (Connection (Time t) (From f) (To xk) (Amp a) (Width w) (Delay tau)))) )
```
(Program set_dynamic_E (Args s t xk h r L rk)
(Element (Time s-2dT) (Name xk) (Threshold -h) (Refractory r) (Last L))
(Meta (Name rk) (Window s t)
  (Element (Time t) (Name xk) (Threshold h) (Refractory r) (Last L))))

(set_dynamic_E (Args s t D0 5 1 s-2 R0))
(set_dynamic_E (Args s t U D0 2 1 1 R0))
(set_dynamic_E (Args s t X D0 2 1 1 R0))
(set_dynamic_E (Args s t Y D0 2 1 1 R0))
(set_dynamic_E (Args s t Z D0 -2 1 1 R0))

At time $s-dT$, \texttt{set_dynamic_C} initializes the amplitudes of the connections to $A_{UD0} = -2$, $A_{WD0} = -2$, $A_{XD0} = -2$, $A_{YD0} = 2$, $A_{ZD0} = 2$. If $R0$ fires, then the meta command in \texttt{set_dynamic_C} causes the connection command to execute at time $t$, which flips the sign of each of the amplitudes to $A_{UD0} = 2$, $A_{WD0} = 2$, $A_{XD0} = 2$, $A_{YD0} = -2$, $A_{ZD0} = -2$. Similarly, \texttt{set_dynamic_E} initializes the threshold of $D0$ to $\theta_{D0} = -5$. If $R0$ fires, then the meta command causes the element command to execute at time $t$, which flips the sign of the threshold of $D0$. In this case, $\theta_{D0} = 5$.

Similarly, for elements $D1, \ldots, D13$, \texttt{set_dynamic_E} and \texttt{set_dynamic_C} dynamically set the element parameters and the connections from $U,X,W,Y,Z$ to $D1, \ldots, D13$ based on the rest of the quantum random firing pattern $R1, \ldots R13$ and the appropriate parameter values shown in table 3.

**Step 2.3** Set up connections to active elements $G0, G1, G2, \ldots G14$ which represent the number of elements in $\{R0, R1, R2, \ldots R13\}$ that are firing. If 0 are firing, then only $G0$ is firing. Otherwise, if $k > 0$ elements in $\{R0, R1, R2, \ldots R13\}$ are firing, then only $G1, G2, G3, \ldots Gk$ are firing.

(firing_count (Args G0 a b -1)) (firing_count (Args G1 a b 1)) . . .
(firing_count (Args G13 a b 25)) (firing_count (Args G14 a b 27))

**Step 2.4** Element $P3$ represents the output of $\pi_3$. Initialize $P3$’s threshold based on meta commands that use the firing activity from elements $G0, G1, G2, \ldots G13$. Since $t + dT < t + 2dT < \cdots < t + 15dT$, the meta commands set the threshold for $P3$ to $-2(14 - k) + 1$ where $k$ is the number of firings. For example, if nine of the randomly chosen bits are high, then $G9$ will fire, so the threshold of $P3$ is set to $-9$. If five of the random bits are high, then the threshold of $P3$ is set to $-17$. Each element of the level set $\overline{\pi}_{13}^{-1}\{0\}$ creates a firing pattern of $D0, \ldots D13$ equal to the complement of the random firing pattern $R0R1\ldots R13$ (i.e., $Dk$ fires if and only if $Rk$ does not fire).

(set_P_threshold (Args G P s t a b theta kdT)
(Meta (Name G) (Window s t)
  (Element (Time t+kdT) (Name P) (Threshold theta) (Refractory b-a) (Last t-b+a))))

(set_P_threshold (Args G0 P3 s t a b -27 dT))
(set_P_threshold (Args G1 P3 s t a b -25 2dT))
the firing activity of $P$ or next tape symbol during a UTM computational step. If an eavesdropper is able to listen to $\eta$ above. However, if random bit $b$ step, the AEM uses six additional quantum random bits shown, respectively in tables 6, 7, 8, 9 and 10. The outputs of $R_i$ are represented by active elements $P_i$. For $P_3$, if random bit $b_3 = 1$, then the dynamical connections from $D_0, D_1, \ldots D_{13}$ to $P_3$ are chosen as described above. However, if random bit $b_3 = 0$, then the amplitudes of the dynamical connections from $D_0, D_1, \ldots D_{13}$ to $P_3$ and the threshold of $P_3$ are multiplied by $-1$. This causes $P_3$ to fire when $\bar{\eta}_3(UWX,YZ) = 0$ and $P_3$ doesn’t fire when $\bar{\eta}_3(UWX,YZ) = 1$.

This cloaking of $P_3$’s firing activity can be coordinated with a meta command based on the value of $b_3$ so that $P_3$’s firing is appropriately interpreted to dynamically change the active elements and connections that update the UTM tape contents and state after each computational step. This cloaking procedure can also be used for element $P_k$ and random bit $b_k$ where $k \in \{0, 1, 2, 4, 5\}$. Furthermore, the same methods can be used to cloak the active element firing patterns that represent the UTM’s state and tape contents. In particular, these methods may help cloak the firing pattern representing the halting state $H$.

Besides representing and computing the program $\bar{\eta}$ with quantum random firing patterns, there are other useful functions computed by active elements executing the UTM. Assume that these connections and the active element firing activity are kept perfectly secret as they represent the state and the tape contents of the UTM tape contents. Alternatively, the active elements representing the UTM tape contents and state may be cloaked similar to the description for cloaking elements $P_0, P_1, \ldots, P_5$.

- Three active elements $(q 0)$, $(q 1)$ and $(q 2)$ store the current state of the UTM.
- There are a collection of elements to represent the tape head location $k$ where $k$ is an integer.
A marker active element $L$ locates the leftmost tape square and a separate marker active element $R$ locates the rightmost tape square. Any tape symbols outside these markers are assumed to be blank (i.e., 0). If the tape head moves beyond the leftmost tape square, then $L$’s connection is removed and updated one tape square to the left and the machine is reading a 0. If the tape head moves beyond the rightmost tape square, then $R$’s connection is removed and updated one tape square to the right and the machine is reading a 0.

There are a collection of elements that represent the tape contents of the UTM. For each tape square $k$ inside the marker elements, there are two elements named $(S_k)$ and $(T_k)$ whose firing pattern determines the alphabet symbol at tape square $k$. For example, if elements $(S_5)$ and $(T_5)$ are not firing, then tape square 5 contains alphabet symbol 0. If element $(S_{-7})$ is firing and element $(T_{-7})$ is not firing, then tape square $-7$ contains alphabet symbol $y$. If element $(S_{-4})$ is not firing and element $(T_{-4})$ is firing, then tape square $-4$ contains alphabet symbol 1. If elements $(S_{13})$ and $(T_{13})$ are both firing, then tape square 13 contains alphabet symbol $A$.

• Representing alphabet symbol 0 with two active elements that are not firing is convenient because if the tape head moves beyond the initial tape contents of the UTM, then the meta command can add two elements that are not firing to represent the contents of the new square.

The copy program helps construct useful functionality in the UTM. The following program helps copy a new alphabet symbol to the tape.

(Program copy_symbol (Args s t b0 a0 b1 a1)
 (copy (Args s t b0 a0)) (copy (Args s t b1 a1)) )

The following program enables a new state to be copied.

(Program copy_state (Args s t b0 a0 b1 a1 b2 a2)
 (copy (Args s t b0 a0)) (copy (Args s t b1 a1)) (copy (Args s t b2 a2)) )

The sequence of steps by which the UTM is executed with an AEM are described below.

1. Tape contents are initialized and the marker elements $L$ and $R$ are initialized.

2. The tape head is initialized to tape square $k = 0$ and the current machine state is initialized to $q_2$. In other words, $(q \ 0)$ is not firing $(q \ 1)$ is firing and $(q \ 2)$ is not firing.

3. $(S_k)$ and $(T_k)$ are copied to $a_{in}$ and the current state $(q \ 0), (q \ 1), (q \ 2)$ is copied to $q_{in}$. and $m$ represents the tape head move.

4. If $q_{out} = H$, then the UTM halts: the AEM reaches a firing pattern that stores the current tape contents indefinitely and keeps the tape head fixed at tape square $k$ where the UTM halted.

5. Otherwise, the firing pattern of the three elements representing $q_{out}$ are copied to $(q \ 0), (q \ 1), (q \ 2)$. $a_{out}$ is copied to the current tape square represented by $(S_k), (T_k)$.

6. If $m = L$, then first determine if the tape head has moved to the left of the tape square marked by $L$. If so, then have $L$ remove its current marker and mark tape square $k - 1$. In either case, go back to step 3 where $(S \ k - 1)$ and $(T \ k - 1)$ are copied to $a_{in}$.
7. If \( m = R \), then first determine if the tape head has moved to the right of the tape square marked by \( R \). If so, then have \( R \) remove its current marker and mark tape square \( k + 1 \).

In either case, go back to step 3 where \((S\,k + 1)\) and \((T\,k + 1)\) are copied to \( a_{in} \).

First, a simple lemma is proved. This lemma will help show that for a non-halting UTM computation the firing activity of the elements computing \( \eta \) (i.e., \( \{A_0, \ldots, A_{14}, P_0\}, \{B_0, \ldots, B_{15}, P_1\}, \{C_0, \ldots, C_{14}, P_2\}, \{D_0, \ldots, D_{13}, P_3\}, \{E_0, \ldots, E_{12}, P_4\} \) and \( \{F_0, \ldots, F_{13}, P_5\} \)) is incomputable.

**Lemma 4.1.** Let \( \gamma: \mathbb{N} \rightarrow \{0, 1\} \) be an incomputable function. For each \( k \) such that \( 1 \leq k \leq m \) assume the boolean function \( B_k: \{0, 1\}^n \rightarrow \{0, 1\}^n \) is invertible. Let \( h: \mathbb{N} \rightarrow \{1, \ldots, m\} \) be a computable function. Define function \( g: \mathbb{N} \rightarrow \{0, 1\} \) where \( (g(1), g(2), \ldots, g(n)) = B_{h(1)}(\gamma(1), \gamma(2), \ldots, \gamma(n)) \) and where \( (g(n + 1), g(n + 2), \ldots, g(2n)) = B_{h(2)}(\gamma(n + 1), \gamma(n + 2), \ldots, \gamma(2n)) \), and for the \( j \)th n-tuple in \( \{0, 1\}^n \) \( (g(jn + 1), g(jn + 2), \ldots, g((j + 1)n)) \).

Then \( g \) is an incomputable function.

**Proof.** By reductio ad absurdum, suppose \( g \) is a computable function. Then \( g(1), g(2), \ldots, g(m) \) are computable. Since \( h(1) \) is computable and each \( B_k \) is invertible and boolean, \( B_{h(1)}^{-1} \) is computable. Then \( B_{h(1)}^{-1}(g(1), g(2), \ldots, g(n)) \) is computable which equals \( (\gamma(1), \ldots, \gamma(n)) \). This completes the base case that \( \gamma \) is computable. Repeating this computation \( j - 1 \) times, by induction, \( \gamma(1), \ldots, \gamma((j - 1)n) \) are computable. For the inductive step, \( g(jn + 1), g(jn + 2), \ldots, g((j + 1)n) \) is computable. Since \( h(j) \) is computable, then \( B_{h(j)}^{-1} \) is computable. Thus, \( B_{h(j)}^{-1}(g(jn + 1), g(jn + 2), \ldots, g((j + 1)n)) \) is computable which equals \( (\gamma(jn + 1), \gamma(jn + 2), \ldots, \gamma((j + 1)n)) \). The induction principle implies that \( \gamma(i) \) is computable for every natural number \( i \) which is a contradiction.

**Definition 3.** Representing the firing activity of the active elements computing \( \eta \)

Consider the \( j \)th computational step of the UTM. For \( k \) such that \( 1 \leq k \leq 15 \), let \( f_{k,j} = 1 \) if element \( A_{k-1} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{16,j} \) represent the firing activity of \( P_0 \). For \( k \) such that \( 17 \leq k \leq 32 \), let \( f_{k,j} = 1 \) if element \( B_{k-17} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{33,j} \) represent the firing activity of \( P_1 \). For \( k \) such that \( 34 \leq k \leq 48 \), let \( f_{k,j} = 1 \) if element \( C_{k-34} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{49,j} \) represent the firing activity of \( P_2 \). For \( k \) such that \( 50 \leq k \leq 63 \), let \( f_{k,j} = 1 \) if element \( D_{k-50} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{64,j} \) represent the firing activity of \( P_3 \). For \( k \) such that \( 65 \leq k \leq 77 \), let \( f_{k,j} = 1 \) if element \( E_{k-65} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{78,j} \) represent the firing activity of \( P_4 \). For \( k \) such that \( 79 \leq k \leq 92 \), let \( f_{k,j} = 1 \) if element \( F_{k-79} \) fires during this \( j \)th step and \( f_{k,j} = 0 \), otherwise. Let \( f_{93,j} \) represent the firing activity of \( P_5 \).

**Definition 4.** Unbounded number of computable UTM steps

If the UTM execution does not halt, then \( f_{k,j} \) is defined for every natural number \( j \). If the UTM execution halts on a finite or infinite number of tape inputs, then the definition of \( f_{1,j}, \ldots, f_{93,j} \) can be extended as follows. Suppose the first tape input halts after \( m_1 \) steps. Before the second UTM execution starts, it is assumed that the UTM’s tape contents are initialized using a computable method. For the first computational step, \( j = m_1 + 1 \). Inductively, if the \( k \)th UTM execution halts after \( m_k \) steps, then it is assumed that the UTM’s tape contents
are initialized using a computable method. On the first step of the \( k + 1 \)th UTM execution \( j = m_1 + m_2 + \ldots + m_k + 1 \).

For an unbounded number of computable UTM steps, define the function \( g : \mathbb{N} \to \{0, 1\} \) where \( g(93(j-1)+r+1) = f_{r+1},j \) and \( 0 \leq r < 93 \). In other words, \( g(1) = g(93(1-1)+1) = f_{1,1} \), \( g(2) = f_{2,1} \), \ldots, \( g(93) = f_{93,1} \) and so on.

**Theorem 4.2.** If an unbounded number of computable UTM steps are executed by the AEM according to Procedure 2 and the two quantum randomness axioms hold, then \( g \) is an incomputable function.

**Proof.** It suffices to show that the sequence \((f_{1,1}, f_{2,1}, \ldots, f_{93,1}, \ldots) \) in \( \Omega \) is Turing incomputable. The proof covers the computation of \( \eta_q \). Similar steps, described below, can be verified for \( \eta_{q_0}, \eta_{q_1}, \eta_{q_2}, \eta_{q_4} \) and \( \eta_5 \). Lemma 4.1 is applied and the proof is completed.

The function \( \gamma \) in (4.1) is defined based on the firing activity of \( R_0, R_1, \ldots, R_{13} \) discussed in steps 2.1 and 2.2 and quantum random bit \( b_3 \). From (4), the two quantum randomness axioms imply that \( \gamma \) is Turing incomputable. (Alternatively, it can be shown that the two quantum randomness axioms induce a Lebesgue probability measure on \( \Omega \). The computable sequences in \( \Omega \) have Lebesgue measure zero because they are countable. Thus, any infinite quantum random sequence is incomputable with probability = 1).

From definition 3, \( g \) represents the firing activity of elements \( A_0, \ldots, A_{14}, B_0, \ldots, B_{15}, C_0, \ldots, C_{14}, D_0, \ldots, D_{13}, E_0, \ldots, E_{12}, F_0, \ldots, F_{13} \) and output elements \( P_0, P_1, P_2, P_3, P_4, P_5 \). \( g \) corresponds to function \( g \) in lemma 4.1. The elements \( D_0, \ldots, D_{13} \) are discussed in steps 2.1 and 2.2. For each computational step of the UTM, table (4) shows how the AEM maps the firing activity of \( R_0, R_1, \ldots, R_{13}, b_3 \) to the firing activity of \( D_0, D_1, \ldots, D_{13}, P_3 \). This is verified by checking the program definitions of \texttt{set\_dynamic\_C} and \texttt{set\_dynamic\_E} in step 2.2. For example, if the current UTM instruction is \((111, 00)\) i.e., in state 111 and reading tape symbol 00 – then this corresponds to boolean function \( B_{(111,00)} \) shown in the first row of table (4). \( B_{(111,00)} \) defines the mapping of \( R_0, R_1, \ldots, R_{13}, b_3 \) to the firing activity of \( D_0, D_1, \ldots, D_{13}, P_3 \).

Each boolean function \( B_k \) in table (4) is invertible because \( B_k \circ B_k \) equals the identity.

The final verification is that \( h : \mathbb{N} \to \{1, 2, \ldots, 15\} \) is computable. First, identify 1 with instruction \((111, 00)\), 2 with instruction \((110, 00)\), \ldots, 14 with instruction \((010, 11)\) and 15 with \( \eta_{\pi_4^{-1}} \{0\} \). Let \( (q, T_k) \) be the current state and tape symbol of the UTM at the jth computational step. Define \( h(j) = (q, T_k) \) if \( \eta_{\pi_4}(q, T_k) = 1 \). Otherwise, define \( h(j) = \eta_{\pi_4^{-1}} \{0\} \). Since \( h \) can be computed based on the computation of the UTM, \( h \) is computable. In the other case, where the UTM execution halts one or more times, definition 4.4 assumes that the initialization of the UTM tape contents is computable, so \( h \) is computable.

**Theorem 4.3.** If an adversary can only eavesdrop on the firing activity of \((f_{1,1}, f_{2,1}, \ldots, f_{93,1}, \ldots, f_{1,j}, \ldots) \) then the AEM execution, described in Procedure 3 of the UTM is perfectly secret. In other words, \( P(q = q_k) = P(q = q_k \mid f_{1,j} = b_1 \ldots f_{m,j} = b_m) \) and \( P(T_k = a_k) = P(T_k = a_k \mid f_{1,j} = b_1 \ldots f_{m,j} = b_m) \) for each \( b_i \in \{0, 1\} \), where \( q \) is the current state of the UTM and \( T_k \) is the contents of the \( k \)th tape square.

**Proof.** The proof is a straightforward consequence of the two quantum randomness axioms, the boolean functions in table (4) characterizing the firing activity of \( f_{1,1}, \ldots, f_{1,j}, \ldots, f_{93,1}, \ldots \) and the definition of conditional probability.

Let event \( Q \) correspond to \( q = q_6 \). Let event \( B \) correspond to \( f_{1,j} = b_1 \). Then event \( B \) corresponds to \( f_{1,j} = 1 - b_1 \). The two quantum randomness axioms imply \( P(R_k = 0) \).
firing patterns that are Turing incomputable. For an unbounded number of computable UTM

<table>
<thead>
<tr>
<th>Boolean Function</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(B_{(11,00)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_0 = x_0. \text{ When } k \neq 0, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,00)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_1 = x_1. \text{ When } k \neq 1, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(11,01)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_2 = x_2. \text{ When } k \neq 2, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,10)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_3 = x_3. \text{ When } k \neq 3, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,01)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_4 = x_4. \text{ When } k \neq 4, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,11)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_5 = x_5. \text{ When } k \neq 5, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,10)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_6 = x_6. \text{ When } k \neq 6, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(10,00)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_7 = x_7. \text{ When } k \neq 7, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(01,10)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_8 = x_8. \text{ When } k \neq 8, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(01,01)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_9 = x_9. \text{ When } k \neq 9, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(01,11)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_{10} = x_{10}. \text{ When } k \neq 10, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(00,10)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_{11} = x_{11}. \text{ When } k \neq 11, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(00,11)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_{12} = x_{12}. \text{ When } k \neq 12, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(00,01)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_{13} = x_{13}. \text{ When } k \neq 13, \ y_k = 1 - x_k. \ p_3 = b_3.)</td>
</tr>
<tr>
<td>(B_{(01,01)}(x_0,x_1,\ldots,x_{13},b_1) = (y_0,y_1,\ldots,y_{13},p_1))</td>
<td>(y_{k} = 1 - x_k \text{ for each } k. \ p_3 = 1 - b_3.)</td>
</tr>
</tbody>
</table>

Table 4: Boolean functions \(B_k : \{0,1\}^{15} \to \{0,1\}^{15}\) for \(\pi_3\)

\(x_k \equiv 1\) if element \(R_k\) fires.\(x_k \equiv 0\) if element \(R_k\) doesn’t fire.

\(P(R_k = 1) = \frac{1}{2};\) from table \[4\] this implies \(P(B) = P(\overline{B}) = \frac{1}{2}.\) Now \(P(B) P(Q \mid B) = P(Q \cap B).\) Also, \(P(Q) = P(Q \cap B) + P(Q \cap \overline{B}).\) Then \(P(Q) = \frac{1}{2}P(Q \mid B) + \frac{1}{2}P(Q \mid \overline{B})\) because \(P(B)P(Q \mid B) = P(Q \cap B).\) Lastly, \(P(Q \mid B) = P(Q \mid \overline{B})\) because \(P(R_k = 0) = P(R_k = 1) = P(B) = P(\overline{B}) = \frac{1}{2}\) and from the properties of the boolean functions in table \[4\] Thus, \(P(Q) = P(Q \mid B)\) which means that \(P(q = q_k) = P(q = q_k \mid f_{1,j} = b_1)\) Similar steps can be repeated to show that \(P(q = q_k) = P(q = q_k \mid f_{m,j} = b_1 \ldots f_{m,j} = b_m)\) and \(P(T_k = \alpha_k) = P(T_k = \alpha_k \mid f_{1,j} = b_1 \ldots f_{m,j} = b_m).\)

It is worth mentioning that if the system reveals information about the dynamic connections from \(R_k\) to \(P_k\) or the quantum random bits generated or firing activity of elements \(R_k\) or those elements representing the UTM tape or state, then the perfect secrecy doesn’t hold.

**Corollary 1.** Consider an unbounded number of computable steps generated by the AEM in procedure \[4\] where the sequence of UTM instructions is \(I_1, I_2, \ldots, I_k, \ldots,\) and \(\pi_k \equiv Q \times A\) as defined in table \[2\]. Define function \(f : \mathbb{N} \to Q \times A\) as \(f(k) = I_k.\) If an adversary can only eavesdrop on \(g\) (the firing activity computing \(\pi\)), then there does not exist a Turing machine that can map \(g\) back to \(f.\)

**Proof.** This corollary follows from definition \[1\] and the work in lemma \[4.1\] and theorems \[4.2\] and \[4.3\]

## 5 Summary

By using a finite AEM program and the meta command, and executing procedure \[2\] any Turing machine program – with a computable unbounded execution – can be executed with AEM firing patterns that are Turing incomputable. For an unbounded number of computable UTM execution steps, when appropriate parameters of the AEM execution are not revealed to an adversary, then there does not exist a Turing machine that can map \(g\) (i.e., the active element
firing patterns computing \( \eta \) back to the sequence of universal Turing machine instructions executed by the AEM.

6 Acknowledgements

I would like to thank Wolfgang Halang, Michael Jones, Don Knuth, David Lewis, A. Mayer, Lutz Mueller, Don Saari and Mario Stipčević for their helpful advice.

References

7 Appendix

7.1 Turing Machine

Define a Turing Machine, where the program definition $\eta$ is explicitly represented as a function instead of quintuples (9, 45).

**Definition 5. Turing Machine**

A Turing machine is a triple $(Q, A, \eta)$ where

- $Q$ is a finite set of states that does not contain a unique halting state, represented as $H$.
- When machine execution begins, the machine is in an initial state $s$ and $s \in Q$.
- $A$ is a finite set of alphabet symbols that are read from and written to the tape.
- $L$ and $R$ represent advancing the tape head to the left or right square, respectively.
- $\eta$ is a function where $\eta : Q \times A \rightarrow Q \times A \times \{L, R\} \cup \{H\} \times A \times \{h\}$. $\eta$ 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:
  - Machine $(Q, A, \eta)$ changes to state $r$.
  - Machine $(Q, A, \eta)$ rewrites alphabet symbol $\alpha$ as symbol $\beta$ on the tape.
  - If $x = L$, 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.
  - If $x = R$, 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.
  - If $x = h$, machine $(Q, A, \eta)$ enters the halting state $H$ and the machine stops executing.

**Definition 6. Turing Machine Tape**

The Turing machine tape $T$ is represented as a function $T : \mathbb{Z} \rightarrow A$ where $\mathbb{Z}$ is 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 9 and 45 assume the initial tape, before program execution begins, is $M$-bounded and the tape contains only blank symbols, denoted here as #, outside the bound. The symbol on the $k$th square of the tape is $T(k)$.

**Definition 7. Configuration with Tape Head Location**

Let $(Q, A, \eta)$ be a Turing machine with tape $T$. A configuration is an element of the set $C = (Q \cup \{H\}) \times \mathbb{Z} \times \{T : T$ is tape with range $A\}$. If $(q, k, T)$ is a configuration, then $k$ is called the tape head location.
Consider the configuration \((p, 2, \ldots ##\alpha \beta ## \ldots)\). 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. The underlining of \(\beta\) indicates that the tape head is currently scanning tape square 2.

**Definition 8. Turing Machine Computational Step**

Given Turing machine \((Q, A, \eta)\) in current configuration \((q, k, T)\) such that \(T(k) = \alpha\). After the execution of one computational step, the new configuration is determined by one and only one of the three cases.

1. \((r, k - 1, S)\) if \(\eta(q, \alpha) = (r, \beta, L)\) for non-halting state \(r\).
2. \((r, k + 1, S)\) if \(\eta(q, \alpha) = (r, \beta, R)\) for non-halting state \(r\).
3. \((H, k, T)\) if \(\eta(q, \alpha) = (H, \alpha, h)\) for halting state \(H\).

In cases (1) and (2) the new tape \(S(j) = T(j)\) whenever \(j \neq k\) and \(S(k) = \beta\). In case (3) the machine execution halts. Sometimes \((q, \alpha)\) is called a Turing machine instruction.

If the machine is currently in configuration \((q_0, k_0, T_0)\) and over the next \(n\) steps the sequence of machine configurations (points) is \((q_0, k_0, T_0), (q_1, k_1, T_1), (q_2, k_2, T_2), \ldots, (q_n, k_n, T_n)\), then this execution sequence is sometimes called the next \(n\) computational steps.

### Table 5: Minsky Universal Turing Machine with Program \(\eta\) from \([33]\)

<table>
<thead>
<tr>
<th>(y)</th>
<th>(0)</th>
<th>1</th>
<th>(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_1)</td>
<td>((q_1, 0, L))</td>
<td>((q_1, 0, L))</td>
<td>((q_2, 1, L))</td>
</tr>
<tr>
<td>(q_2)</td>
<td>((q_1, 0, L))</td>
<td>((q_2, y, R))</td>
<td>((q_2, A, R))</td>
</tr>
<tr>
<td>(q_3)</td>
<td>((q_3, y, L))</td>
<td>((H, 0, h))</td>
<td>((q_3, A, L))</td>
</tr>
<tr>
<td>(q_4)</td>
<td>((q_4, y, L))</td>
<td>((q_5, y, R))</td>
<td>((q_7, 1, L))</td>
</tr>
<tr>
<td>(q_5)</td>
<td>((q_5, y, R))</td>
<td>((q_3, y, L))</td>
<td>((q_5, A, R))</td>
</tr>
<tr>
<td>(q_6)</td>
<td>((q_6, y, R))</td>
<td>((q_3, A, L))</td>
<td>((q_6, A, R))</td>
</tr>
<tr>
<td>(q_7)</td>
<td>((q_7, 0, R))</td>
<td>((q_6, y, R))</td>
<td>((q_7, 1, R))</td>
</tr>
</tbody>
</table>

State set \(Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}\). Alphabet \(A = \{y, 0, 1, A\}\). Halt state \(H\).

### 7.2 Active Element Machine Architecture

Define the extended integers as \(\mathbb{Z} = \{m + kdT : m, k \in \mathbb{Z} \text{ and } dT \text{ is a fixed infinitesimal}\}\). For more on infinitesimals, see keyword \([1]\) and reference \([30]\).

**Definition 9. Machine Architecture**

\(\Gamma, \Omega, \text{ and } \Delta\) are index sets that index the input, computational, and output active elements, respectively. Depending on the machine architecture, the intersections \(\Gamma \cap \Omega\) and \(\Omega \cap \Delta\) can be empty or non-empty. A machine architecture, denoted as \(A(I, E, O)\), consists of a collection of input active elements, denoted as \(I = \{E_i : i \in \Gamma\}\); a collection of computational active elements \(E = \{E_i : i \in \Omega\}\); and a collection of output active elements \(O = \{E_i : i \in \Delta\}\).

Each computational and output active element, \(E_i\), has the following components and properties:
A threshold $\theta_i$

A refractory period $r_i$ where $r_i > 0$.

A collection of pulse amplitudes $\{A_{ki} : k \in \Gamma \cup \Omega\}$.

A function of time, $\Psi_i(t)$, representing the time active element $E_i$ last fired.

A collection of transmission times $\{\tau_{ki} : k \in \Gamma \cup \Omega\}$, where $\tau_{ki} > 0$ for all $k \in \Gamma \cup \Omega$.

A function of time, $\Psi_i(t)$, representing the time active element $E_i$ last fired. $\Psi_i(t) = \sup\{s : s < t$ and $g_i(s) = 1\}$, where $g_i(s)$ is the output function of active element $E_i$ and is defined below. The sup is the least upper bound and is always defined here, whence $\Psi_i$ is well-defined.

A binary output function, $g_i(t)$, representing whether active element $E_i$ fires at time $t$. The value of $g_i(t) = 1$ if $\sum A_{ki}(t) > \theta_i$ where the sum ranges over all $k \in \Gamma \cup \Omega$ and $t \geq \Psi_i(t) + r_i$. In all other cases, $g_i(t) = 0$. For example, $g_i(t) = 0$, if $t < \Psi_i(t) + r_i$.

A set of firing times of active element $E_k$ within active element $E_i$’s integrating window, $W_{ki}(t) = \{s : \text{active element } E_k \text{ fired at time } s \text{ and } 0 \leq t - s - \tau_{ki} < \omega_{ki}\}$. Let $|W_{ki}(t)|$ denote the number of elements in the set $W_{ki}(t)$. If $W_{ki}(t) = \emptyset$, then $|W_{ki}(t)| = 0$.

A collection of input functions, $\{\phi_{ki} : k \in \Gamma \cup \Omega\}$, each a function of time, and each representing pulses coming from computational active elements, and input active elements. The value of the input function is computed as $\phi_{ki}(t) = |W_{ki}(t)|A_{ki}(t)$.

The refractory periods, transmission times and pulse widths are positive integers; and pulse amplitudes and thresholds are integers. These parameters are a function of time (i.e., $\theta_i(t), r_i(t), A_{ki}(t), \omega_{ki}(t), \tau_{ki}(t)$). Time $t$ is an element of the extended integers $\mathbb{Z}$.

Input active elements that are not computational have the same characteristics as computational elements, except they have no inputs $\phi_{ki}$ coming from elements in this machine. Input elements are assumed to be externally firable. An external source such as the environment or an output element from another distinct machine $M(T', E', O')$ can cause an input element to fire. An input element can fire at any time after its refractory period has expired. An element can be an input and computational element. Similarly, an element can be an output and computational element. Alternatively, when an output element, $E_i$, is not a computational element, where $i \in \Delta - \Omega$, then $E_i$ does not send pulses to elements in this machine.

If $g_i(s) = 1$, this means active element $E_i$ fired at time $s$. The refractory period, $r_i$, is the amount of time that must elapse after active element $E_i$ just fired before $E_i$ can fire again. The transmission time, $\tau_{ki}$, is the amount of time it takes for active element $E_i$ to find out that active element $E_k$ has fired. The pulse amplitude, $A_{ki}$, represents the strength of the pulse that active element $E_k$ transmits to active element $E_i$ after active element $E_k$ has fired. After this pulse reaches $E_i$, the pulse width $\omega_{ki}$ represents how long the pulse lasts as input to active element $E_i$. If $A_{ki} = 0$, then there is no connection from active element $E_k$ to active element $E_i$.

### 7.3 Active Element Machine Programming Language

This subsection describes a programming language for the active element machine. There are five types of commands Element, Connection, Fire, Program and Meta.

**Syntax 1.** AEM Program

In Backus-Naur form, an AEM program is defined as follows.
Syntax 2.  

AEM Symbols and Extended Integer Expressions

<ename> ::= <int> | <symbol>
<symbol> ::= <char_symbol><str_tail> | ((<ename> . . . <ename>)
<str_tail> ::= "" | <char_symbol><str_tail> | 0<str_tail> | <pos_int><str_tail>
<char_symbol> ::= <letter> | <special_char>
<letter> ::= <lower_case> | <upper_case>
<lower_case> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<upper_case> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<special_char> ::= _

The following rules represent the extended integers, addition and subtraction.

<int> ::= <pos_int> | <neg_int> | 0
<neg_int> ::= - <pos_int>
<pos_int> ::= <non_zero><digits>
<digits> ::= <numeral> | <numeral><digits>
<non_zero> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<numeral> ::= "" | <non_zero> | 0

<aint> ::= <aint><math_op><d> | <d><math_op><aint> | <d>
<math_op> ::= + | -
<d> ::= <int> | <symbol_string> | <infinitesimal>
<infinitesimal> ::= dT

Command 1.  

An Element command specifies the time when an active element is created or its parameter values are updated. This command has the following Backus-Naur syntax.

<element_cmd> ::= (Element (Time <aint>) (Name <ename>) (Threshold <int>)
(Refractory <pos_int>) (Last <int>))

The keyword Time tags the time value $s$ (extended integer) at which the element is created or updated. If the name symbol value is $E$, the keyword Name tags the name $E$ of the active element. The keyword Threshold tags the threshold $\theta_E(s)$ assigned to $E$. Refractory tags the refractory value $r_E(s)$. The keyword Last tags the last time fired value $\Psi_E(s)$.

Command 2.  

A Connection command creates or updates a connection from one active element to another active element. This command has the following Backus-Naur syntax.

<cnct_cmd> ::= (Connection (Time <aint>) (From <ename>) (To <ename>)
[(Amp <int>) (Width <pos_int>) (Delay <pos_int>])]

The keyword Time tags the time value $s$ at which the connection is created or updated. The keyword From tags the name $F$ of the active element that sends a pulse with these updated values. The keyword To tags the name $T$ of the active element that receives a pulse with these updated values. The keyword Amp tags the pulse amplitude value $A_{FT}(s)$ that is assigned to
this connection. The keyword Width tags the pulse width value \( \omega_{FT}(s) \). The keyword Delay tags the transmission time \( \tau_{FT}(s) \).

When the AEM clock reaches time \( s \), \( F \) and \( T \) are name values that must be the name of an element that has already been created or updated before or at time \( s \). Not all of the connection parameters need to be specified in a connection command. If the connection does not exist beforehand and the Width and Delay values are not specified appropriately, then the amplitude is set to zero and this zero connection has no effect on the AEM computation. The connection exists indefinitely with the same parameter values until a new connection is executed at a later time between From element \( F \) and To element \( T \).

**Command 3. Fire**

The Fire command has the following Backus-Naur syntax.

\[
\text{<fire_cmd>} ::= (\text{Fire} \ (\text{Time} \ <\text{aint}>\ ) \ (\text{Name} \ <\text{ename}>\ ) )
\]

The Fire command fires the active element indicated by the Name tag at the time indicated by the Time tag. This command can be used to fire input active elements.

**Command 4. Program**

The Program combines a sequence of commands into a single command. It has the following definition syntax.

\[
\text{<program_def>} ::= (\text{Program} \ <\text{pname}> \ [(\text{Cmds} \ <\text{cmds}>)] \ [(\text{Args} \ <\text{args}>)] \ <\text{cmd_sequence}>)
\]

\[
\text{<pname>} ::= <\text{ename}>
\]

\[
\text{<cmds>} ::= <\text{cmd_name}> | <\text{cmd_name}><\text{cmds}>
\]

\[
\text{<cmd_name>} ::= \text{Element} | \text{Connection} | \text{Fire} | \text{Meta} | <\text{pname}>
\]

\[
\text{<args>} ::= <\text{symbol}> | <\text{symbol}><\text{args}>
\]

The Program command has the following execution syntax.

\[
\text{<program_cmd>} ::= (<\text{pname}> \ [(\text{Cmds} \ <\text{cmds}>)] \ [(\text{Args} \ <\text{args_cmd}>)])
\]

\[
\text{<args_cmd>} ::= <\text{ename}> | <\text{ename}><\text{args_cmd}>
\]

**Keyword 1. dT**

The keyword \( dT \) represents a positive infinitesimal amount of time. If \( m \) and \( n \) are integers and \( 0 \leq m < n \), then \( mdT < ndT \). Furthermore, \( dT > 0 \) and \( dT \) is less than every positive rational number. Similarly, \( -dT < 0 \) and \( -dT \) is greater than every negative rational number. The infinitesimal \( dT \) helps coordinate almost simultaneous events that are non-commutative or indeterminate. For example, element \( A \) could be about to receive a pulse from element \( B \) at the same time that a connection between them is removed.

**Keyword 2. clock**

The keyword clock evaluates to an integer, which is the current active element machine time. clock is an instance of \( <\text{ename}> \).

If the current AEM time is 5, then command

\[
(\text{Element} \ (\text{Time} \ \text{clock}) \ (\text{Name} \ \text{clock}) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1))
\] executes as

\[
(\text{Element} \ (\text{Time} \ 5) \ (\text{Name} \ 5) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1))
\]

After \( (\text{Element} \ (\text{Time} \ \text{clock}) \ (\text{Name} \ \text{clock}) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1)) \) is created, then at each time step this command is executed with the current time of the AEM. If this command is in the original AEM program before the clock starts at 0, then the following sequence of elements named 0, 1, 2, \ldots are created.

\[
(\text{Element} \ (\text{Time} \ 0) \ (\text{Name} \ 0) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1))
\]

\[
(\text{Element} \ (\text{Time} \ 1) \ (\text{Name} \ 1) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1))
\]

\[
(\text{Element} \ (\text{Time} \ 2) \ (\text{Name} \ 2) \ (\text{Threshold} \ 1) \ (\text{Refractory} \ 1) \ (\text{Last} \ -1)) \ldots
\]
Command 5. *Meta*

The *Meta* command causes a command to execute when an element fires within a window of time. This command has the following execution syntax.

\[
\begin{align*}
\langle \text{meta\_cmd} \rangle & := (\text{Meta} \ (\text{Name} \ <\text{ename}>) \ [\text{<win\_time>}]) \ \langle \text{AEM\_cmd} \rangle \\
\langle \text{win\_time} \rangle & := (\text{Window} \ <\text{aint}> \ <\text{aint}>)
\end{align*}
\]

To understand the behavior of the meta command, consider the execution of

\[
(\text{Meta} \ (\text{Name} \ E) \ (\text{Window} \ l \ w) \ (\text{C} \ (\text{Args} \ t \ a)))
\]

where \(E\) is the name of the active element. The keyword \(\text{Window}\) tags an interval, which is called a *window of time*. \(l\) is an integer, which locates one of the boundary points of the window of time. Usually, \(w\) is a positive integer, so the window of time is \([1, 1+w]\). If \(w\) is a negative integer, then the window of time is \([1+w, 1]\).

The command \(C\) executes each time that \(E\) fires during the window of time, which is either \([1, 1+w]\) or \([1+w, 1]\), depending on the sign of \(w\). If the window of time is omitted, then command \(C\) executes at any time that element \(E\) fires.

In regard to the meta command, the following assumption is analogous to the Turing machine tape being unbounded as Turing program execution proceeds. (See Definitions 5 and 6.) During execution of a finite active element program, an active element can fire and due to one or more meta commands, new elements and connections can be added to the machine. As a consequence, at any time the active element machine only has a finite number of computing elements and connections but the number of elements and connections can be unbounded as a function of time as the active element program executes.

### 7.4 Active Element Machine Computation

In section 3 the firing patterns of active elements represent the computation of a boolean function. Firing representations, machine computation and interpretation are defined below.

**Definition 10. Firing Representation**

Consider active element \(E_i\)’s firing times in the interval of time \(W = [t_1, t_2]\). Let \(s_1\) be the earliest firing time of \(E_i\) lying in \(W\), and \(s_n\) the latest firing time lying in \(W\). Then \(E_i\)’s firing sequence \(F(E_i, W) = \{s_1, ..., s_n\} = \{s \in W : y_i(s) = 1\}\) is called a firing sequence of the active element \(E_i\) over the window of time \(W\). From active elements \(\{E_1, E_2, ..., E_n\}\), create the tuple \((F(E_1, W), F(E_2, W), ..., F(E_n, W))\), which is called a firing representation of the active elements \(\{E_1, ..., E_n\}\) within the window of time \(W\).

As \(E_i\)’s refractory period is a positive integer and \(t_1\) and \(t_2\) are finite, observe that \(s_1\) and \(s_n\) are well-defined and the set \(F(E_i, W)\) is finite. At the machine level of interpretation, firing representations (firing patterns) express the input to, the computation of, and the output of an active element machine. At a more abstract level, firing representations can represent a sequence of symbols, a spatio-temporal pattern, or even a family of program instructions.

**Definition 11. Sequence of Firing Representations**

Let \(W_1, ..., W_n\) be a sequence of time intervals. Let \(\mathcal{F}(\mathcal{E}, W_1) = (F(E_1, W_1), F(E_2, W_1), ..., F(E_n, W_1))\) be a firing representation of active elements \(\mathcal{E} = \{E_1, ..., E_n\}\) over the interval \(W_1\). In general, let \(\mathcal{F}(\mathcal{E}, W_k) = (F(E_1, W_k), F(E_2, W_k), ..., F(E_n, W_k))\) be a firing representation over the interval of time \(W_k\). From these, a sequence of firing representations, \([\mathcal{F}(\mathcal{E}, W_1), \mathcal{F}(\mathcal{E}, W_2), ..., \mathcal{F}(\mathcal{E}, W_n)]\) is created.
Definition 12. Machine Computation

Let \( [\mathcal{F}(E, W_1), \mathcal{F}(E, W_2), \ldots, \mathcal{F}(E, W_n)] \) be a sequence of firing representations. Suppose \( [\mathcal{F}(E, S_1), \mathcal{F}(E, S_2), \ldots, \mathcal{F}(E, S_m)] \) is some other sequence of firing representations. Suppose machine architecture \( M(I, E, O) \) has input active elements \( I \) fire with the pattern \( [\mathcal{F}(E, S_1), \mathcal{F}(E, S_2), \ldots, \mathcal{F}(E, S_m)] \) and consequently \( M \)'s output active elements \( O \) fire according to \( [\mathcal{F}(E, W_1), \mathcal{F}(E, W_2), \ldots, \mathcal{F}(E, W_n)] \). In this case, the machine \( M \) computes \( [\mathcal{F}(E, W_1), \mathcal{F}(E, W_2), \ldots, \mathcal{F}(E, W_n)] \) from \( [\mathcal{F}(E, S_1), \mathcal{F}(E, S_2), \ldots, \mathcal{F}(E, S_m)] \).

An active element machine is an interpretation between two sequences of firing representations if the machine computes the output sequence from the input sequence.

7.5 Active Element Machine Level Set Rules

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>( (U, A_i) )</th>
<th>( (W, A_i) )</th>
<th>( (X, A_i) )</th>
<th>( (Y, A_i) )</th>
<th>( (Z, A_i) )</th>
<th>( \theta_{A_i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 10</td>
<td>( A_0 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>7</td>
</tr>
<tr>
<td>111 01</td>
<td>( A_1 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>111 00</td>
<td>( A_2 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 11</td>
<td>( A_3 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 10</td>
<td>( A_4 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 01</td>
<td>( A_5 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>101 11</td>
<td>( A_6 )</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>101 10</td>
<td>( A_7 )</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>101 01</td>
<td>( A_8 )</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>100 11</td>
<td>( A_9 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>100 10</td>
<td>( A_{10} )</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>100 01</td>
<td>( A_{11} )</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>100 00</td>
<td>( A_{12} )</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>011 11</td>
<td>( A_{13} )</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>010 11</td>
<td>( A_{14} )</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>( (U, B_i) )</th>
<th>( (W, B_i) )</th>
<th>( (X, B_i) )</th>
<th>( (Y, B_i) )</th>
<th>( (Z, B_i) )</th>
<th>( \theta_{B_i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 11</td>
<td>( B_0 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>111 10</td>
<td>( B_1 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>7</td>
</tr>
<tr>
<td>111 01</td>
<td>( B_2 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>111 00</td>
<td>( B_3 )</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 11</td>
<td>( B_4 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 10</td>
<td>( B_5 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 01</td>
<td>( B_6 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>110 00</td>
<td>( B_7 )</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>101 00</td>
<td>( B_8 )</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>100 01</td>
<td>( B_9 )</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>011 10</td>
<td>( B_{10} )</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>011 01</td>
<td>( B_{11} )</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>010 11</td>
<td>( B_{12} )</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>010 01</td>
<td>( B_{13} )</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>010 00</td>
<td>( B_{14} )</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>001 01</td>
<td>( B_{15} )</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 8: AEM Separation Rules for Level Set $\pi_2^{-1}\{1\}$

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>$(U, E_i)$</th>
<th>$(W, E_i)$</th>
<th>$(X, E_i)$</th>
<th>$(Y, E_i)$</th>
<th>$(Z, E_i)$</th>
<th>$\theta_{E_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 10</td>
<td>$C_0$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>7</td>
</tr>
<tr>
<td>111 01</td>
<td>$C_1$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 00</td>
<td>$C_2$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>101 11</td>
<td>$C_3$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>101 10</td>
<td>$C_4$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>101 01</td>
<td>$C_5$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>100 00</td>
<td>$C_6$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>100 01</td>
<td>$C_7$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>100 00</td>
<td>$C_8$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>011 10</td>
<td>$C_9$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>011 01</td>
<td>$C_{10}$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>010 10</td>
<td>$C_{11}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>001 11</td>
<td>$C_{12}$</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>001 10</td>
<td>$C_{13}$</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>001 00</td>
<td>$C_{14}$</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 9: AEM Separation Rules for Level Set $\pi_4^{-1}\{1\}$

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>$(U, E_i)$</th>
<th>$(W, E_i)$</th>
<th>$(X, E_i)$</th>
<th>$(Y, E_i)$</th>
<th>$(Z, E_i)$</th>
<th>$\theta_{E_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 01</td>
<td>$E_0$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 11</td>
<td>$E_1$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 01</td>
<td>$E_2$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 00</td>
<td>$E_3$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>101 11</td>
<td>$E_4$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>101 01</td>
<td>$E_5$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>100 11</td>
<td>$E_6$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>100 01</td>
<td>$E_7$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>011 11</td>
<td>$E_8$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>011 01</td>
<td>$E_9$</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>010 01</td>
<td>$E_{10}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>001 11</td>
<td>$E_{11}$</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>001 01</td>
<td>$E_{12}$</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 10: AEM Separation Rules for Level Set $\pi_5^{-1}\{1\}$

<table>
<thead>
<tr>
<th>Firing Pattern</th>
<th>Element</th>
<th>$(U, F_i)$</th>
<th>$(W, F_i)$</th>
<th>$(X, F_i)$</th>
<th>$(Y, F_i)$</th>
<th>$(Z, F_i)$</th>
<th>$\theta_{F_i}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>111 11</td>
<td>$F_0$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>111 10</td>
<td>$F_1$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>111 01</td>
<td>$F_2$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>111 00</td>
<td>$F_3$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>110 11</td>
<td>$F_4$</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>110 10</td>
<td>$F_5$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>110 01</td>
<td>$F_6$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>101 11</td>
<td>$F_7$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>101 10</td>
<td>$F_8$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>101 01</td>
<td>$F_9$</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>100 00</td>
<td>$F_{10}$</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>010 11</td>
<td>$F_{11}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>010 01</td>
<td>$F_{12}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>010 00</td>
<td>$F_{13}$</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>