A computational implementation of the Rowlands-Diaz universal rewrite system

[1]Sydney Rowlands, [2]Peter Rowlands

[1]jrist29@gmail.com
[2]Physics Department, University of Liverpool, Oliver Lodge Laboratory, Oxford St, Liverpool. L69 7ZE, UK; p.rowlands@liverpool.ac.uk

Abstract. The meta-pattern of the universe, first formulated by Rowlands and Diaz [2002], is a universal rewrite system or URS. This universal pattern finds a formulation in formal language theory that is centered around the fundamental semantic unit of the zero word or the zero string: $0 = X_0$, $X_0^{\#}$. This is realized successively in the deterministic Turing machine, Post machine, Finite machine with two pushdown stores, and non-deterministic linear bounded automaton.

1.Introduction

A universal alphabet and rewrite system, first formulated by Rowlands and Diaz [1-6], has a strong claim to be the fundamental meta-system in Nature, sought by Bateson among others [7]. Essentially, it predicates an infinitely degenerate totality zero as the state of the universe at all time, which is realized by an infinite succession of zero-totality alphabets, each of which ensures uniqueness by incorporating its predecessor. The succession is not necessarily temporal, but rather supervenient, as time is a product of the process, rather than an assumed component. It is a succession of zero cardinalities, rather than a succession of infinite ones.

The technical details of the process, which are remarkably simple as we would expect, are described in many publications but are outlined in section 2 of this paper as Rewrite Rules 1.0. Essentially, there is only one process of transition between successive states of the universe, but it simultaneously requires two aspects, signified respectively by $\rightarrow$ and $\Rightarrow$, and referred to, for convenience, as 'conserve' and 'create'. A state of the universe is described by a zero-totality alphabet, in which each component is always accompanied by a conjugate (signified by *). The alphabet can then be concatenated either with one or more components of itself or 'subalphabet' (conserve) or with its entire self (create). The first aspect yields only automorphisms of the alphabet, whereas the second produces an entirely new alphabet. However, the new alphabet will only be valid if it both contains the previous alphabet and also fulfils the requirements of 'conserve' in that all the new components concatenate with the new alphabet to produce automorphisms of that alphabet. (The automorphisms differ in producing different ordering of the terms, but are identical in totality.)

We can describe the process in practical terms using symbols, though symbols are not themselves necessary to the process. There is no fixed start or end state, though we can define a start and end for our convenience, and the process is effectively a fractal. The alphabet is not fixed but extends continuously, and the production rules are recreated at every stage, although there are generic similarities between the stages. The concatenation process can be conveniently described using replacement rules for the symbols, which are illustrated in the way that the automorphisms occur. At each stage, just one entirely new symbol is created, but this is accompanied by concatenations with all the previously created

symbols. The replacement rules are determined by the requirement that the new symbol must also describe a new process – newness cannot be created by the symbol itself. Various things emerge from the symbolic representation and the need for it always to produce something new, including the fact that any symbol, other than the starting symbol R (or identity), does not concatenate with itself to produce R, but rather its conjugate, and that successive new symbols following R, beginning, say with A and B, concatenate to produce AB, which also concatenates with itself to produce the conjugate of R, which we represent as R*. In principle, an *anticommutativity* is introduced into the system to ensure that A and B are new and not just a new representation of R. The anticommutativity also introduces an aspect of closure and discreteness, not previously assumed. It additionally means that the only way to continue the sequence is to introduce new pairs of symbols which are anticommutative to each other but commutative to all the others. The series can then continue to infinity with the uniqueness of each new symbol assured by the fact that it has a unique partner with which it anticommutes. In effect the alphabet continues to infinity by incorporating a generically repeating aspect.

In addition, entropy is built into the structure in a significantly pure form in that each successive stage effectively doubles the alphabet for the previous one by adding a new symbol and all its concatenations. If the number of independent 'microstates' is $W = 2^n$ at level n, then taking a logarithmic function of this reveals that the entropy is simply an index of the relative level reached. In effect, entropy is a description of the working of the system of Nature, not an additional property requiring explanation [4, 6, 8]. The system is also deterministic in that no symbol repeats; each is distinct, only repeating generically at a higher level.

The system has, since its first conception, been used in many applications, for example, in generating mathematical structures, such as the real numbers, integers, quaternions, Clifford algebra, and even Conway's surreal numbers, in addition to those of mathematical logic. The mathematics that it resembles most is Clifford algebra, suggesting the particular significance of this algebra in the description of many aspects of Nature, but no mathematical structure is excluded. The only basic assumption is totality zero and no further assumptions need to be made to generate them. It is even possible to generate the full sequence of Cayley-Dickson algebras *as mathematic*s, though the evidence suggests that the primary version has no need to introduce antiassociativity along with anticommutativity. In fact, the rewrite system as a purely *natural* process needs no inputs precisely because it is not antiassociative. Antiassociativity forces us to choose between options, whereas anticommutativity merely forces into the only available one. When time emerges as an intrinsic component of physics and all sciences based upon it, we see that antiassociativity would require time as well as space reversal at a fundamental level, and, in effect, time, because of the algebraic structure which emerges with its definition, has an associativity which cannot be changed.

By contrast with mathematics, it has been shown that physics has a special structure in that its four basic parameters, mass, time, charge and space have the properties (real / imaginary, conserved / non-conserved, dimensional / nondimensional) and mathematical structures (real, complex, quaternion and complex quaternion) required by the first four alphabets starting from R, and that these, when combined into the highest alphabet, lead to structures which concatenate to a complete zero, meaning that all subsequent alphabets will automatically become zero without being specified. This *nilpotent structure* is ubiquitous in physics at all levels and in all natural systems defined by the conservation of energy or Newton's third law of motion, or with a changing energy that can be defined by a known process. In effect, the principles of relativistic quantum mechanics, defined by the nilpotent structure, become the template for investigating all higher order systems [9-21]. A parallel system of genetics, also using four component units (A, T, G, C) uses exactly the same 64-part mathematics as physics (the algebra of a double space or space and conjugate space); while the identical algebra also leads to fundamental particle structures [4, 22-25].

The structures from the rewrite system are determined by characteristic mathematical patterns (in particular, duality, anticommutativity and symmetry-breaking, associated with the numbers 2, 3 and 5) which scale upward via a replacement of the original components by higher order ones, and which ultimately include such areas as nuclear physics, atomic structure and the Periodic Table, chemistry,

systems (physical, biological, higher order and constructed), physiology, evolutionary and cell biology, and consciousness among others, as has been demonstrated in a long series of publications. Computing aspects include automated reasoning or AI and the complexity problem with an investigation of the p / np question by Marcer and Rowlands suggesting that the structured nature of the rewrite system and the regularity of its application must lead to an answer favouring the p (or polynomial) alternative [4, 26-32]. A category theory application is currently under development by the present authors.

A key area of investigation is in the theory of formal languages in computer science. Applications of this in current technology include programming languages such as C++, Java, xml, html, extensible markup languages, compiling, parsing (text mining). The aim of this paper is to reveal the formal language aspect of the rewrite system, and to demonstrate that the pattern of this system conforms to the rules of its own language generation structure and that this language is recognizable by a Turing machine algorithm. In principle, we show that the rewrite system can simulate language defined by a set of meaningful pattern units ('words'). This language, which we identify as the language of nature, is a type 1 (context sensitive) language.

Current formal language theory suggests that an infinite alphabet requires a finite repeating unit and an infinite countable set of symbols. These are related to the duality and anticommutativity (2 and 3) of the rewrite system. Diaz and Rowlands have already used the rewrite to develop computer language by developing algebraic interpretation as infinite square roots of –1 [3], including a special unit repeated (quaternions), and extending to infinity in Clifford algebra, which was originally developed from the closed group of quaternions plus Grassmann's infinite tensor outer product. We have already shown that physics, derived from the rewrite system, has such a structure, combining nilpotent fermions as a generically repeated unit, with an infinite Hilbert space derived from the Grassmann algebra. In fact the rewrite system itself is a universal version of this pattern, probably the most general that can be derived. The remainder of the paper shows how to construct a computer-compatible simulation of the rewrite system, up to a limit determined by the user.

2.Language generation

To produce sentences in a particular language requires knowledge of the rules of sentence formation. Sentient intelligence possesses this ability and computational devices simulate this ability. But the universe itself seems to exist on a simple fundamental meta-pattern (never repeated) that can be formulated as a language with appropriate grammar rules. This universal pattern was discovered by Peter Rowlands and Bernard Diaz, together with the initial form of the grammar rules [1-6]. A formulation of the grammar rules of the universal rewriting system (URS) adapted to formal language theory engineered by Sydney Rowlands, assisted by Peter Rowlands, faithfully reproduces the same output as the original formulation of the grammar rules. However, this adapted and re-engineered version of the grammar rules introduces the universal rewrite system to the subject of formal language analysis and compilation theory, which might have applications for computerized simulation of the physical laws operating in the universe, in addition to technological consequences. The generative grammar is given in this section, followed by the Turing machine algorithm in section 3. It is significant that this algorithm only accepts zero words, as required by the universal rewrite system, but the simulated process (unlike the natural one) can be terminated by a halting condition set by the user.

2.1 Generative Grammar in Formal Language Theory

Grammar G = (V_N, V_T, P, S) where

V_N = set of non-terminal symbols (generally assumed to be a finite set)
V_T = set of terminal symbols (generally assumed to be a finite set)
P = set of production rules
S = start symbol

2.2 Generative Grammar for the Universal Rewrite System (URS)
Grammar G(URS) = (V_N(URS), V_T(URS), P(URS), S) where

V_N(URS) = {X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$, X_{2i+2}, $X_{2i+2}^{\#}$| i ≥ 0 and i ∈ Z^+} (a countably infinite set)
V_T(URS) = {R_0, $R_0^{\#}$, R_{2i+1}, $R_{2i+1}^{\#}$, R_{2i+2}, $R_{2i+2}^{\#}$| i ≥ 0 and i ∈ Z^+} (a countably infinite set)
P(URS) = {

(0) create first new symbol (assume a nonzero symbol): S → X_0,
(1) conserve first zero totality: X_0→ X_0, $X_0^{\#}$
(2) conserve zero totality: X_0, $X_0^{\#}$ → X_0, X_0, $X_0^{\#}$
(3) conserve zero totality: X_0, $X_0^{\#}$ → $X_0^{\#}$, X_0, $X_0^{\#}$,
(4) create a new symbol: X_0, $X_0^{\#}$, X_0, $X_0^{\#}$ → X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$
(5) creation rule with terminal symbols, V_T: X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$ → R_0, $R_0^{\#}$, R_{2i+1}, $R_{2i+1}^{\#}$
}

S = start symbol

How to apply the rules:

(0) create S→ X_0
(1) conserve X_0→ X_0, $X_0^{\#}$
(2) conserve X_0, $X_0^{\#}$ → X_0 , X_0 , $X_0^{\#}$
conserve (3) into conserve (2) X_0, (X_0, $X_0^{\#}$) → X_0, ($X_0^{\#}$, X_0, $X_0^{\#}$)
(4) create X_0, $X_0^{\#}$, X_0, $X_0^{\#}$ → X_0 , $X_0^{\#}$, X_{2i+1} , $X_{2i+1}^{\#}$
(5) create X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$ → R_0, $R_0^{\#}$, R_{2i+1}, $R_{2i+1}^{\#}$

The language generated by the grammar URS, L(URS) is the set of words {w_1 | w = $R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}$ and w_2 = $R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}^{\#}R_{2i+2}^{\#}$} derived from the infinite product:

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1}) \text{ , i} \in Z^+.$$

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1})$$ in expanded form becomes the word w_1

$$R_0R_0R_0R_{2i+1}R_0^{\#}R_0R_0^{\#}R_{2i+1}$$

or more familiarly rearranged as

$$R_0, R_0^{\#}, R_{2i+1}, R_{2i+1}^{\#}$$

assuming the symbols operate as though they are quaternions following the quaternion multiplication rules where R_0 the identity, $R_0^{\#}$ minus identity, R_{2i+1} quaternion ***i***, $R_{2i+1}^{\#}$ quaternion –***i***

$R_0, R_0 = R_0$
$R_0^{\#}, R_0 = R_0^{\#}$
$R_0, R_{2i+1} = R_{2i+1}$

$R_0^{\#}, R_{2i+1} = R_{2i+1}^{\#}$

$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$ = in expanded form becomes the word w_2

$$R_0R_0R_0R_0R_0R_{2i+2}R_0R_{2i+1}R_0R_0R_{2i+1}R_{2i+2}R_0^{\#}R_0R_0R_0^{\#}R_0R_{2i+2}R_0^{\#}R_{2i+1}R_0R_0^{\#}R_{2i+1}R_{2i+2}$$

or more familiarly rearranged as

$$R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$$

assuming the symbols operate as though they are quaternions following the quaternion multiplication rules where R_0 the identity, $R_0^{\#}$ minus identity, R_{2i+1} quaternion ***i***, $R_{2i+1}^{\#}$ quaternion –***i***, R_{2i+2} quaternion ***j***, $R_{2i+2}^{\#}$ quaternion –***j***, $R_{2i+1}R_{2i+2}$ quaternion ***ij***, $R_{2i+1}R_{2i+2}{}^{\#}$ quaternion –***ij***

$R_0, R_0, R_0 = R_0$
$R_0^{\#}, R_0, R_0 = R_0^{\#}$
$R_0, R_{2i+1}, R_0 = R_{2i+1}$
$R_0^{\#}, R_{2i+1}, R_0 = R_{2i+1}^{\#}$
$R_0, R_0, R_{2i+2} = R_{2i+2}$
$R_0^{\#}, R_0, R_{2i+2} = R_{2i+2}^{\#}$
$R_0, R_{2i+1}, R_{2i+2} = R_{2i+1}, R_{2i+2}$
$R_0^{\#}, R_{2i+1}, R_{2i+2} = R_{2i+1}R_{2i+2}{}^{\#}$

Instead of continuing the product to infinity, as we propose that Nature does in principle, we can choose to adapt this product to a real machine by terminating the product at some particular power *n* of the order 2^n in the URS, with the help of two functions of *n*: i(*n*) and l(*n*). i(*n*) labels each power *n* and *n* + 1 in the order 2^n of the URS with the same number i(*n*) of complete anti-commutative cycles and l(*n*) numerically labels each conjugate pair symbol, using 2i+1 (= ***i*** in a quaternion representation) in power *n* and 2i + 2 (= ***j*** in a quaternion representation) in power *n*+1 with the same number l(*n*), excluding the conjugate unit pair $R_0, R_0^{\#}$ at power *n* = 1 where both i(*n*) and *l*(*n*) = 0 [see table below].

$$\text{i(n)} = \frac{1}{4}(2n - 3 - (-1)^n)$$

and

$$\text{l(n)} = \frac{1}{4}(2n - 1 - (-1)^{n-1})$$

where i, l ∈ {0, 1, 2 3, …} and *n* ∈ {1, 2, 3, …}.
Additionally, for every power *n* in each order 2^n of the URS there exist (*n* — 1)/2 anti-commutative sets that alternate between complete (for *n* odd, 2i+1 (quaternion ***i***) and 2i + 2 (quaternion ***j***)) and incomplete (for *n* even, 2i+1 (quaternion ***i***)) anticommutative sets.

Here is a table explaining the progression of the URS through increasing values of *n*:

Power of URS n	Order of URS 2^n	i(n)	l(n)	Number of quaternion cycles as a function of n, f(n)
1	2	0	0	0
2	4	0	1	0.5
3	8	1	1	1
4	16	1	2	1.5
5	32	2	2	2
6	64	2	3	2.5
7	128	3	3	3
8	256	3	4	3.5
9	512	4	4	4
10	1024	4	5	4.5
11	2048	5	5	5

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2}) = R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$$

and

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1}) = R_0, R_0^{\#}, R_{2i+1}, R_{2i+1}^{\#}$$

Example of $\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$ and $\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_0, R_{2i+1})$ in operation in the URS:

Order of URS 2^n	Tensor-like Product
2^1	$(R_0, R_0^{\#})$
2^2	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})$
2^3	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$
2^4	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})$
2^5	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$
2^6	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=2}^{2}(R_0, R_0^{\#})(R_0, R_{2i+1})$
2^7	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=2}^{2}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$
2^8	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=2}^{2}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=3}^{3}(R_0, R_0^{\#})(R_0, R_{2i+1})$
2^9	$\prod_{i=0}^{0}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=1}^{1}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=2}^{2}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})\prod_{i=3}^{3}(R_0, R_0^{\#})(R_0, R_{2i+1})(R_0, R_{2i+2})$

These two possibilities will be outlined, with alternative state diagrams, in section 3. Here, we use the idea that one word represents an infinite set of words of that type of expression as a function of the

variable i [33]. The following table shows a comparison of the natural rules (Rewrite Rules 1.0) and those of the simulation (Rewrite Rules 2.0).

Rewrite Rules 1.0	**Rewrite Rules 2.0**
create a new symbol: (X,X*)(X,X*) ⇒ (X,X*Y,Y*) conserve zero totality: X(X,X*) → (X,X*) X*(X,X*) → (X,X*)	(0) create first new symbol (assume a nonzero symbol): S → X_0, (1) conserve first zero totality: X_0→ X_0, $X_0^{\#}$ (2) conserve zero totality: X_0, $X_0^{\#}$ → X_0, X_0, $X_0^{\#}$ (3) conserve zero totality: X_0, $X_0^{\#}$ → $X_0^{\#}$, X_0, $X_0^{\#}$, (4) create a new symbol: X_0, $X_0^{\#}$, $X_{0.}$, $X_0^{\#}$ → X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$ (5) creation rule with terminal symbols, V_T: X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$ → R_0, $R_0^{\#}$, R_{2i+1}, $R_{2i+1}^{\#}$
	(0) create S→ X_0 (1) conserve X_0→ X_0, $X_0^{\#}$ (2) conserve X_0, $X_0^{\#}$ → X_0 , X_0 , $X_0^{\#}$ conserve (3) into conserve (2) X_0, (X_0, $X_0^{\#}$) → X_0, ($X_0^{\#}$, X_0, $X_0^{\#}$) (4) create X_0, $X_0^{\#}$, X_0, $X_0^{\#}$ → X_0 , $X_0^{\#}$, X_{2i+1} , $X_{2i+1}^{\#}$ (5) create X_0, $X_0^{\#}$, X_{2i+1}, $X_{2i+1}^{\#}$ → R_0, $R_0^{\#}$, R_{2i+1}, $R_{2i+1}^{\#}$
(0) 0 → (R) (1) (R) → (RR*) (2) R(RR*) → (RR*) (3) R*(RR*) → (RR*) (4) (RR*)(RR*) ⇒ (RR*, AA^*) (2) (R,A)(RR*, AA*) → (RR*, AA*) (3) (R*, A*)(RR*, AA*) → (RR*, AA*) (4) (RR*, AA*)(RR*, AA*) ⇒ (RR*, AA*, BB* ABAB*) (2) (R, A, B) (RR*, AA*, BB*, $ABAB^*$) → (RR*, AA*, BB*,ABAB*) (3) (R*, A*, B*) (RR*, AA*, BB*, ABAB*) → (RR*, AA*, BB*,ABAB*) (4) (RR*, AA*, BB*, ABAB*)(RR*, AA*, BB*, ABAB*) ⇒ (RR*, AA*, BB*, ABAB*, CC*, ACAC*, BCBC*, ABCABC*) and so on …..	(0) S → X_0 (1) →X_0, $X_0^{\#}$ (2) → X_0, X_0, $X_0^{\#}$ (3) into (2) → X_0, $X_0^{\#}X_0$, $X_0^{\#}$ (4) → X_0, $X_0^{\#}X_1$, $X_1^{\#}$ (2) → X_0, X_0, $X_0^{\#}$, X_1, X_1, $X_1^{\#}$ (3) into (2) → X_0, $X_0^{\#}X_0$, $X_0^{\#}X_1$, $X_1^{\#}X_1$, $X_1^{\#}$ (4) → X_0, $X_0^{\#}X_1$, $X_1^{\#}X_2$, $X_2^{\#}X_1X_2$, $X_1X_2^{\#}$ (2) → X_0, X_0, $X_0^{\#}$, X_1, X_1, $X_1^{\#}$, X_2, X_2, $X_2^{\#}$, X_1, X_2, X_1, X_2, X_1, $X_2^{\#}$ (3) into (2) → X_0,$X_0^{\#}X_0$, $X_0^{\#}X_1$, $X_1^{\#}X_1$, $X_1^{\#}X_2$, $X_2^{\#}X_2$, $X_2^{\#}X_1X_2$, $X_1X_2^{\#}X_1X_2$, $X_1X_2^{\#}$ (4) → X_0, $X_0^{\#}X_1$, $X_1^{\#}X_2$, $X_2^{\#}X_1X_2$, $X_1X_2^{\#}X_3$, $X_3^{\#}X_1X_3$, $X_1X_3^{\#}X_2X_3$, $X_2X_3^{\#}X_1X_2X_3$, $X_1X_2X_3^{\#}$ and so on unless halted by using rule (5) ….

number of symbols per order in URS = 2^n
number of words per order in the URS = 2^{n-1}

3.Language recognition

To understand sentences in a particular language requires knowledge of the rules of sentence formation. Sentient intelligence possesses this ability and computational devices simulate this ability. In the Rowlands-Diaz Rewrite System there do not exist choices – the URS recognized by a Turing Machine over an infinite alphabet moves in only one direction. The repeating part of the pattern of the Rowlands-Diaz Rewrite System is caused by anticommutativity which involves the symmetries of the number 3 or 3-dimensionality. This is the reason for the existence of the repetition in the infinite sequence of square roots of –1.

Formal Definition of a General Turing Machine $T_m = (Q, \Sigma, \Gamma, B, \delta, q_0, F)$ where

Q = set of states
Σ = a finite set of alphabet symbols (input alphabet)
Γ = finite set of auxiliary alphabet symbols (tape alphabet)
δ = transition function
q_0 = start state
F = set of accept states

A procedure that recognises the language of the URS is a Turing Machine composed of two Turing machines, one a subroutine of the other: $T_{m2} = (Q_2, \Sigma, \Gamma, \delta_2, q_0, F_2)$ is a subroutine of $T_{m1} = (Q_1, \Sigma, \Gamma, \delta_1, q_0, F_1)$.

$T_{m1} = (Q_1, \Sigma, \Gamma, \delta_1, q_0, F_1)$

$Q_1 = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$
$\Sigma = V_T(URS) = \{R_0, R_0^{\#}, R_{2i+1}, R_{2i+1}^{\#}, R_{2i+2}, R_{2i+2}^{\#} \mid i \geq 0 \text{ and } i \in Z^+\}$ (a countably infinite set)
$\Gamma = \{r_0, r_0^{\#}, r_{2i+1}, r_{2i+1}^{\#}, r_{2i+2}, r_{2i+2}^{\#} \mid i \geq 0 \text{ and } i \in Z^+\}$ (a countably infinite set)
$\delta_1 = Q_1 \times \Sigma \rightarrow Q_1 \times \Gamma - \{B\} \times \{L, R\}$
q_0 = start state
$F_1 = \{q_8\}$

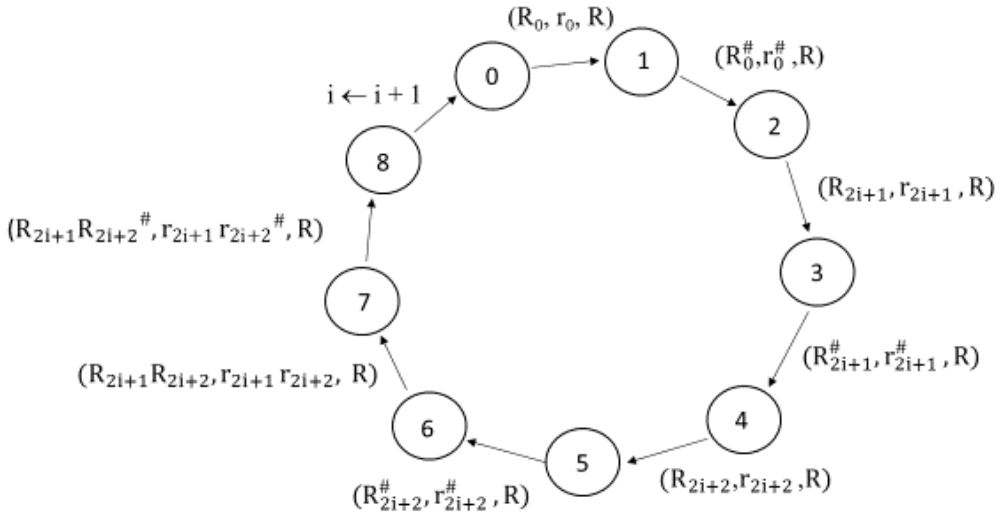
input word $w_1 = R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

Transition Function δ_1

$\delta_1(q_0, R_0) = (q_1, r_0, R)$
$\delta_1(q_1, R_0^{\#}) = (q_2, r_0^{\#}, R)$
$\delta_1(q_2, R_{2i+1}) = (q_3, r_{2i+1}, R)$
$\delta_1(q_3, R_{2i+1}^{\#}) = (q_4, r_{2i+1}^{\#}, R)$
$\delta_1(q_4, R_{2i+2}) = (q_5, r_{2i+2}, R)$
$\delta_1(q_5, R_{2i+2}^{\#}) = (q_6, r_{2i+2}^{\#}, R)$
$\delta_1(q_6, R_{2i+1}R_{2i+2}) = (q_7, r_{2i+1}r_{2i+2}, R)$
$\delta_1(q_7, R_{2i+1}R_{2i+2}{}^{\#}) = (q_8, r_{2i+1}r_{2i+2}{}^{\#}, R)$
$i \leftarrow i + 1$

It should be understood that the transition function in the formal definition of the Turing machine (the deterministic version and the nondeterministic version) is a partial function. There exist ordered pairs (state, input symbol) in the domain of the transition function $Q \times \Sigma$ that are not defined or are not equal to any ordered triples (state, output symbol, move). This indicates that either the Turing machine rejects or loops on these ordered pairs and hence the unsolvable halting problem is revealed through the existence of these undefined pairs. The domain $Q \times \Sigma$ is a Cartesian Product of the two sets Q and Σ which is defined as the set of all (state, input symbol) ordered pairs where state is in Q and symbol is in Σ. There only exist a certain number of such ordered pairs that are defined as having an output action. The transition functions for the Turing machines in this paper only include those (state, input symbol) ordered pairs that are defined as equal to (state, output symbol, move) ordered triples as their corresponding output definition.

state diagram:



state table:

	R_0	$R_0^{\#}$	R_{2i+1}	$R_{2i+1}^{\#}$	R_{2i+2}	$R_{2i+2}^{\#}$	$R_{2i+1}R_{2i+2}$	$R_{2i+1}R_{2i+2}{}^{\#}$
q_0	q_1	undefined	undefined	undefined	undefined	undefined	undefined	undefined
q_1	undefined	q_2	undefined	undefined	undefined	undefined	undefined	undefined
q_2	undefined	undefined	q_3	undefined	undefined	undefined	undefined	undefined
q_3	undefined	undefined	undefined	q_4	undefined	undefined	undefined	undefined
q_4	undefined	undefined	undefined	undefined	q_5	undefined	undefined	undefined
q_5	undefined	undefined	undefined	undefined	undefined	q_6	undefined	undefined
q_6	undefined	undefined	undefined	undefined	undefined	undefined	q_7	undefined
q_7	undefined	undefined	undefined	undefined	undefined	undefined	undefined	q_8

symbol table

	R_0	$R_0^{\#}$	R_{2i+1}	$R_{2i+1}^{\#}$	R_{2i+2}	$R_{2i+2}^{\#}$	$R_{2i+1}R_{2i+2}$	$R_{2i+1}R_{2i+2}{}^{\#}$
q_0	R_0	undefined	undefined	undefined	undefined	undefined	undefined	undefined
q_1	undefined	$R_0^{\#}$	undefined	undefined	undefined	undefined	undefined	undefined
q_2	undefined	undefined	R_{2i+1}	undefined	undefined	undefined	undefined	undefined
q_3	undefined	undefined	undefined	$R_{2i+1}^{\#}$	undefined	undefined	undefined	undefined
q_4	undefined	undefined	undefined	undefined	R_{2i+2}	undefined	undefined	undefined
q_5	undefined	undefined	undefined	undefined	undefined	$R_{2i+2}^{\#}$	undefined	undefined
q_6	undefined	undefined	undefined	undefined	undefined	undefined	$R_{2i+1}R_{2i+2}$	undefined
q_7	undefined	undefined	undefined	undefined	undefined	undefined	undefined	$R_{2i+1}^{\#}R_{2i+2}^{\#}$

$T_{m2} = (Q_2, \Sigma, \Gamma, \delta_2, q_0, F_2)$

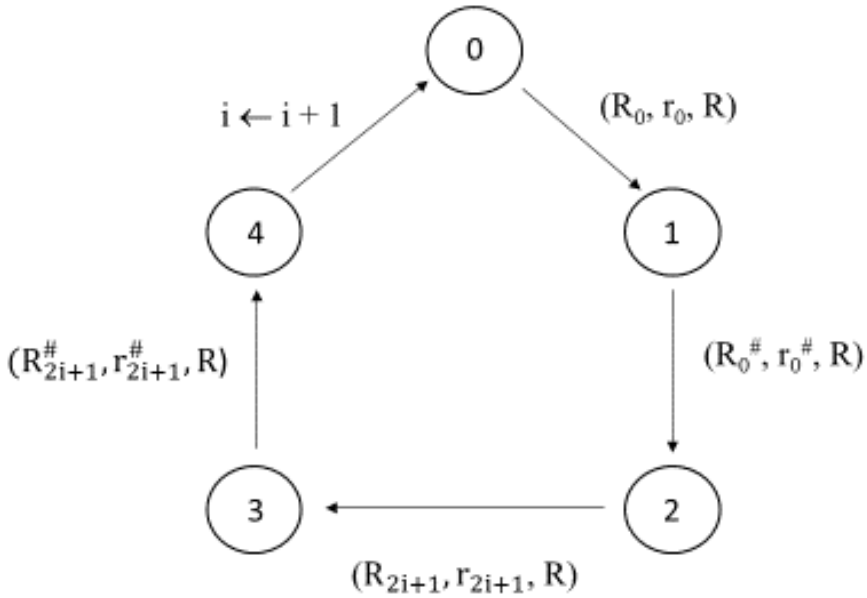
$Q_2 = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = V_T(URS) = \{R_0, R_0{}^{\#}, R_{2i+1}, R_{2i+1}{}^{\#} | \ i \geq 0 \text{ and } i \in Z^+\}$ (a countably infinite set)
$\Gamma = \{r_0, r_0{}^{\#}, r_{2i+1}, r_{2i+1}{}^{\#} | \ i \geq 0 \text{ and } i \in Z^+\}$ (a countably infinite set)
$\delta_2 = Q_2 \times \Sigma \rightarrow Q_2 \times \Gamma - \{B\} \times \{L, R\}$
q_0 = start state
$F_2 = \{q_4\}$

input word $w_2 = R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}$

Transition Function δ_2

$\delta_2 (q_0, R_0) = (q_1, r_0, R)$
$\delta_2 (q_1, R_0^{\#}) = (q_2, r_0{}^{\#}, R)$
$\delta_2(q_2, R_{2i+1}) = (q_3, r_{2i+1}, R)$
$\delta_2(q_3, R_{2i+1}^{\#}) = (q_4, r_{2i+1}{}^{\#}, R)$
$i \leftarrow i + 1$

state diagram:



state table:

	R_0	$R_0^{\#}$	R_{2i+1}	$R_{2i+1}^{\#}$
q_0	q_1	undefined	undefined	undefined
q_1	undefined	q_2	undefined	undefined
q_2	undefined	undefined	q_3	undefined
q_3	undefined	undefined	undefined	q_4

symbol table

	R_0	$R_0^{\#}$	R_{2i+1}	$R_{2i+1}^{\#}$
q_0	R_0	undefined	undefined	undefined
q_1	undefined	$R_0^{\#}$	undefined	undefined
q_2	undefined	undefined	R_{2i+1}	undefined
q_3	undefined	undefined	undefined	$R_{2i+1}^{\#}$

Performance of a Turing Machine T_m that recognises the URS

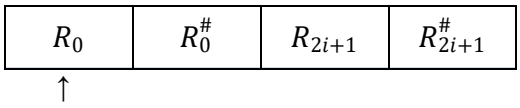
A Turing machine that recognises the URS Tm alternates between T_{m2} and T_{m1} in an infinite loop. The length of the two input words of the URS w_1 and w_2 are $|w_1| = 4$ and $|w_2| = 8$. The Turing machine recognises the universal rewrite system by rewriting every square that has an input alphabet symbol on

it with a corresponding symbol from the set of output symbols. The process proceeds for an infinite succession of alternations between these types of words w_1 and w_2 in the URS language.

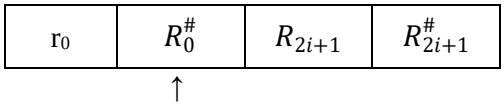
input word x written in input alphabet $\Sigma \rightarrow$ input word x rewritten in output alphabet Γ

T_{m2} subroutine

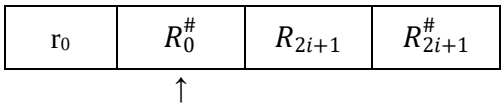
$\delta_2(q_0, R_0)$



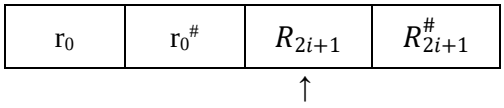
(q_1, r_0, R)



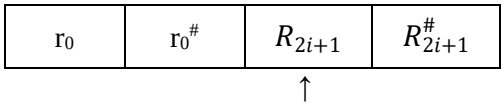
$\delta_2(q_1, R_0^{\#})$



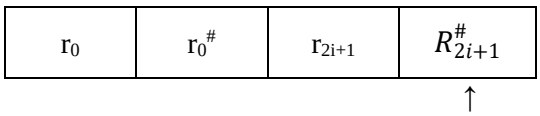
$(q_2, r_0^{\#}, R)$



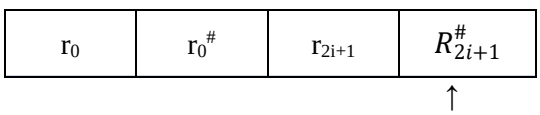
$\delta_2(q_2, R_{2i+1})$



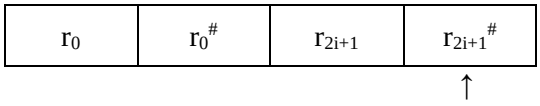
(q_3, r_{2i+1}, R)



$\delta_2(q_3, R_{2i+1}^{\#})$

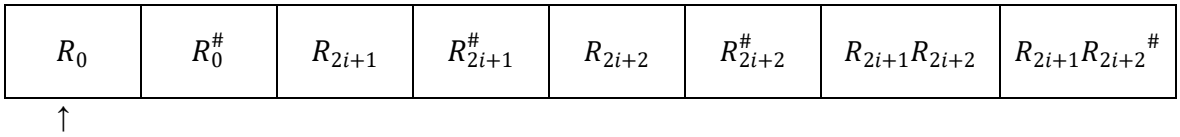


$(q_4, r_{2i+1}^{\#})$

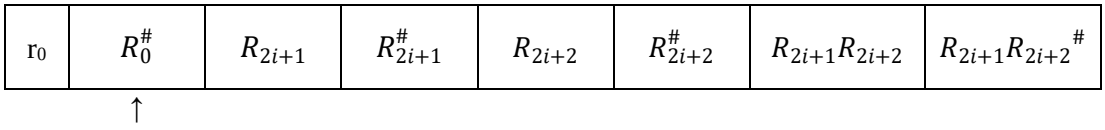


T_{m1}

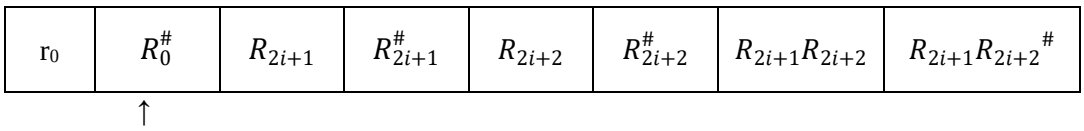
$\delta_1(q_0, R_0)$



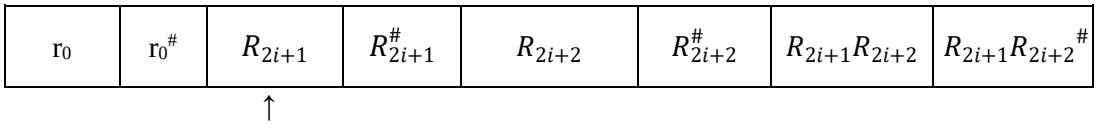
(q_1, r_0, R)



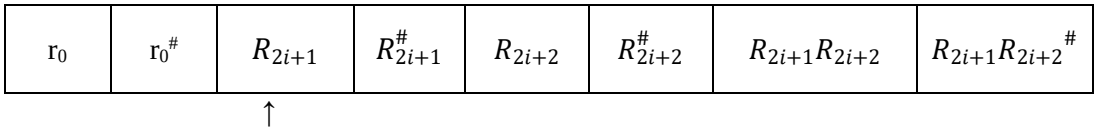
$\delta_1(q_1, R_0^{\#})$



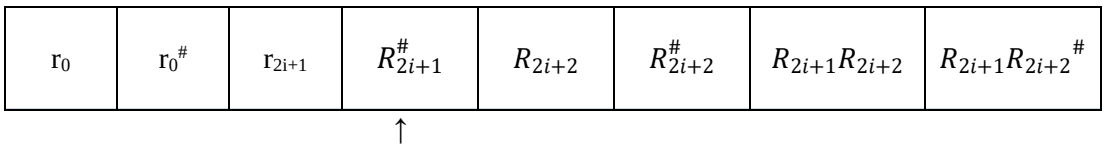
$(q_2, r_0^{\#}, R)$



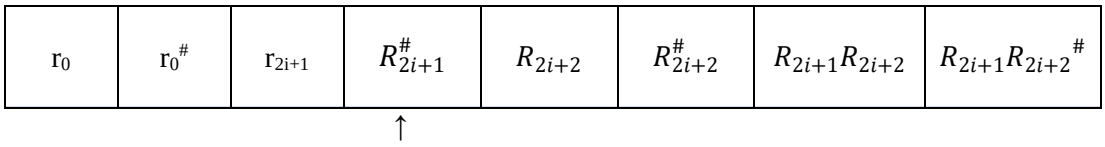
$\delta_1(q_2, R_{2i+1})$



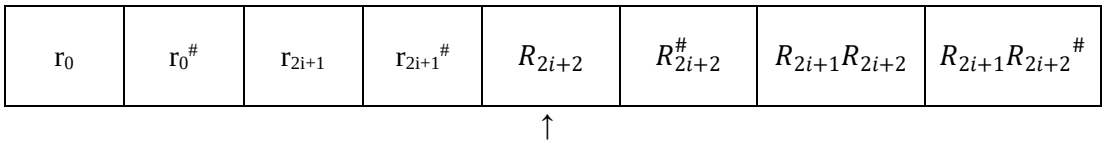
(q_3, r_{2i+1}, R)



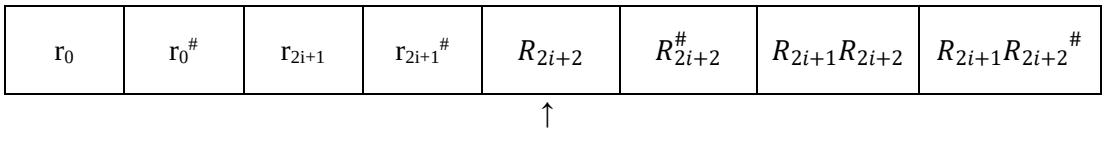
$\delta_1(q_3, R_{2i+1}^{\#})$



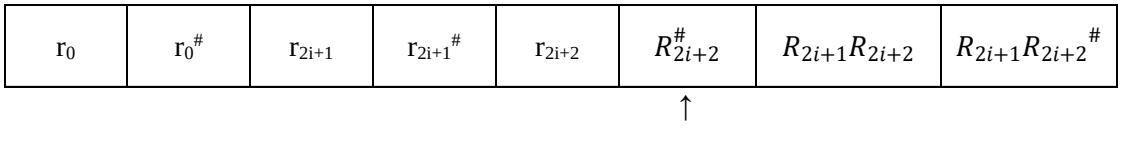
$(q_4, r_{2i+1}^{\#}, R)$



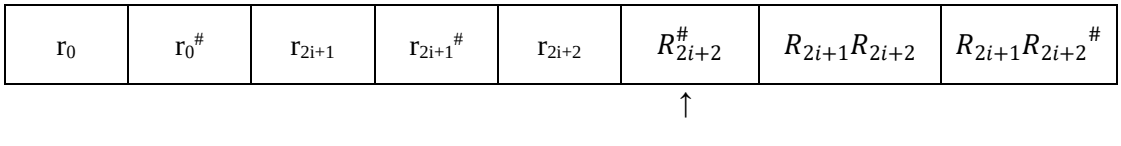
$\delta_1(q_4, R_{2i+2})$



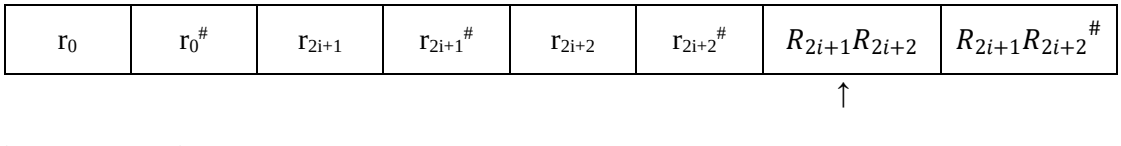
(q_5, r_{2i+2}, R)



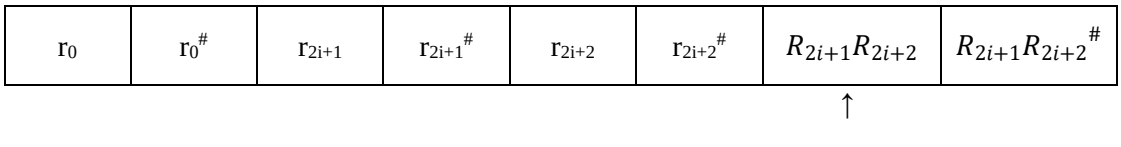
$\delta_1(q_5, R_{2i+2}^{\#})$



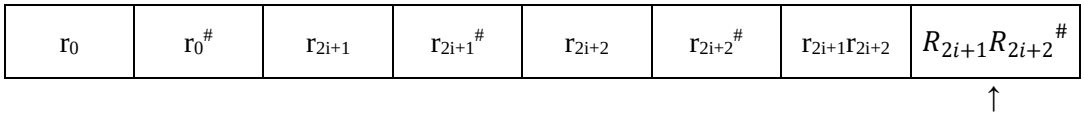
$(q_6, r_{2i+2}^{\#}, R)$



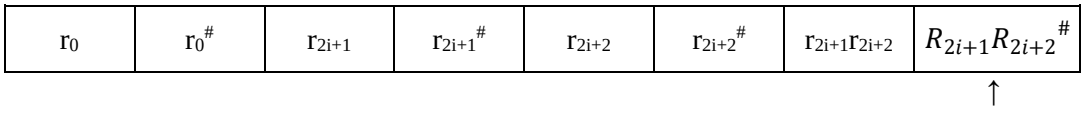
$\delta_1(q_6, R_{2i+1}R_{2i+2})$



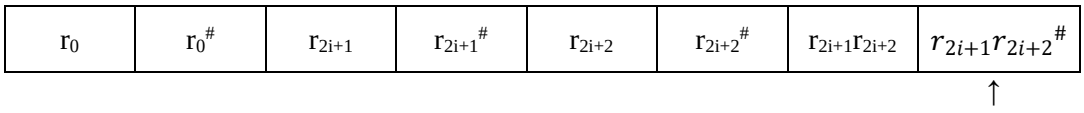
$(q_7, r_{2i+1}r_{2i+2}, R)$



$\delta_1(q_7, R_{2i+1}^{\#}R_{2i+2}^{\#})$



$(q_8, r_{2i+1}^{\#}r_{2i+2}^{\#}, R)$



4.Two equivalent Post machines

There exist two other machines which are equivalent in power, i.e. recognising the same set of words or language, to the Turing machine. These are the Post machine and the Finite machine with two pushdown stores [34]. In the next sections we will generate the flow diagrams for the URS as recognised by these two machines for the two possible input word patterns.

4.1 Formal Definition of a Post Machine

A Post Machine M over $\Sigma \cup \Gamma \cup \{B\}$ is a flow-diagram with one variable x, which may have as a value any word over $\Sigma \cup \Gamma \cup \{B\} \cup \{!\}$, where! is a special auxiliary symbol. Each statement in the flow diagram has one of the following forms:

START statement (exactly one)

x = word w

head(x): gives the head (leftmost letter) of the word x

tail(x): gives the tail of the word x (that is, x with the leftmost letter removed)

$\sigma \cdot x$: concatenates the letter σ and the word x

The idea behind how the Post machine operates in a few lines:

START:x = complete word w in input alphabet Σ ! →
x = tail(x) of word w in input alphabet Σ ! head(x) of word w in output alphabet Γ →
ACCEPT: x = ! complete word w in output alphabet Γ

The Post Machine M is an iterative simulation of the universal rewrite system, compared to the recursive simulation of the Turing machine.

As already stated, equivalence in power means describing the same language or the same set of words.

The set of words over the infinite alphabet Σ that the Post Machine M_1 accepts is

accept(M_1) = $\{r_0, r_0^{\#}r_{2i+1}, r_{2i+1}^{\#}r_{2i+2}, r_{2i+2}^{\#}r_{2i+1}r_{2i+2}, r_{2i+1}r_{2i+2}^{\#} \mid i \geq 0 \text{ and } i \in \mathcal{N}\}$.

reject(M_1) = Σ^* - accept(M_1)

loop(M_1) = ∅

START

x ← x!

x = $R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#}$!

head(x) = R_0

x = $R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#}$!

x ← xr_0

x = $R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0$

head(x) = $R_0^{\#}$

x = $R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0$

x ← x$r_0^{\#}$

x = $R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#}$

head(x) = R_{2i+1}

x = $R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} r_0^{\#}$

x ← xr_{2i+1}

x = $R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1}$

head(x) = $R_{2i+1}^{\#}$

x = $R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1}$

x ← x$r_{2i+1}^{\#}$

x = $R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#}$

head(x) = R_{2i+2}

x = $R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#}$

x ← xr_{2i+2}

x = $R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2}$

head(x) = $R_{2i+2}^{\#}$

x = $R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2}$

x ← x$r_{2i+2}^{\#}$

x = $R_{2i+1} R_{2i+2} R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#}$

head(x) = $R_{2i+1} R_{2i+2}$

x = $R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#}$

x ← x$r_{2i+1} r_{2i+2}$

x = $R_{2i+1} {R_{2i+2}}^{\#} ! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#} r_{2i+1} r_{2i+2}$

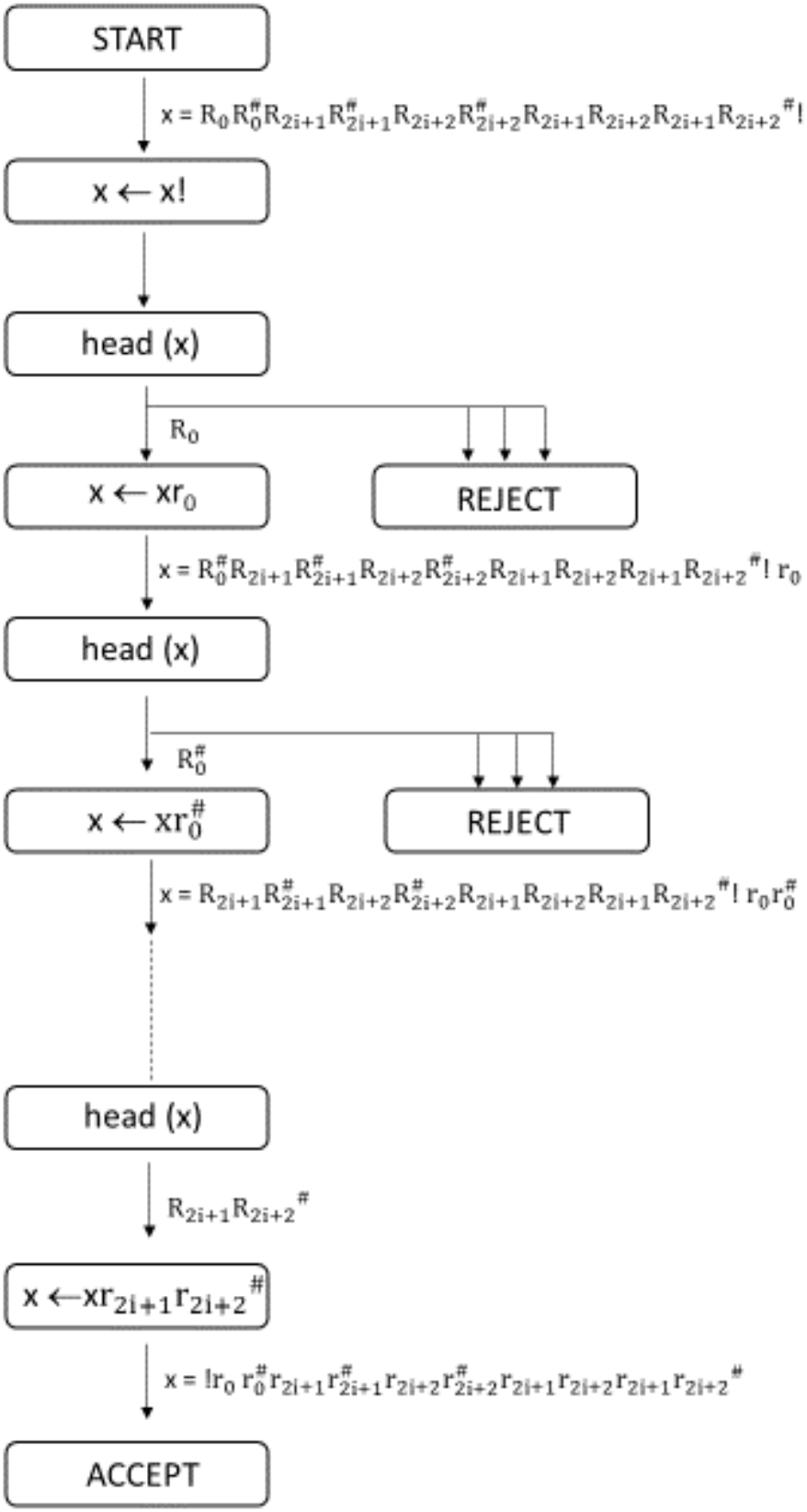
head(x) = $R_{2i+1} {R_{2i+2}}^{\#}$

x = $! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#} r_{2i+1} r_{2i+2}$

x ← x$r_{2i+1}^{\#} r_{2i+2}^{\#}$

x = $! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#} r_{2i+1} r_{2i+2} {r_{2i+2}}^{\#}$

ACCEPT

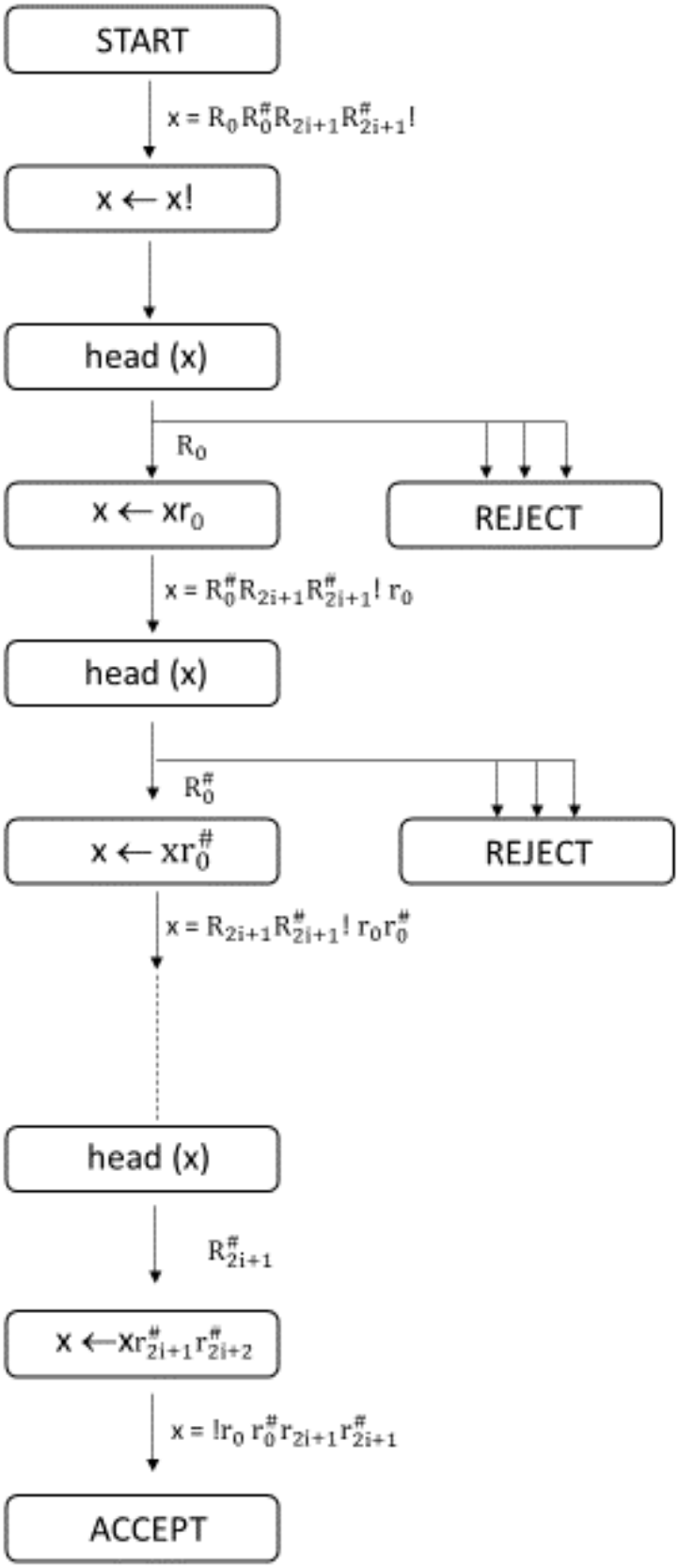


The set of words over the infinite alphabet Σ that the Post Machine M_2 accepts is

accept(M_2) = $\{r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} \mid i \geq 0 \text{ and } i \in \mathcal{N}\}$.
reject(M_2) = Σ^* - accept(M_1)

loop(M_2) = $\emptyset$
START
x ← x!
x = $R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#}!$

head(x) = R_0
x = $R_0^{\#} R_{2i+1} R_{2i+1}^{\#}!$
x ← xr_0
x = $R_0^{\#} R_{2i+1} R_{2i+1}^{\#}! r_0$
head(x) = $R_0^{\#}$
x = $R_{2i+1} R_{2i+1}^{\#}! r_0$
x ← x$r_0^{\#}$
x = $R_{2i+1} R_{2i+1}^{\#}! r_0 r_0^{\#}$
head(x) = R_{2i+1}
x = $R_{2i+1}^{\#}! r_0 r_0^{\#}$
x ← xr_{2i+1}
x = $R_{2i+1}^{\#}! r_0 r_0^{\#} r_{2i+1}$
head(x) = $R_{2i+1}^{\#}$
x = $! r_0 r_0^{\#} r_{2i+1}$
x ← x$r_{2i+1}^{\#}$
x = $! r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#}$
ACCEPT


START
$x = R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#} !$
$x \leftarrow x!$
head (x)
R_0
$x \leftarrow xr_0$
REJECT
$x = R_0^{\#} R_{2i+1} R_{2i+1}^{\#} ! \; r_0$
head (x)
$R_0^{\#}$
$x \leftarrow xr_0^{\#}$
REJECT
$x = R_{2i+1} R_{2i+1}^{\#} ! \; r_0 r_0^{\#}$
head (x)
$R_{2i+1}^{\#}$
$x \leftarrow xr_{2i+1}^{\#} r_{2i+2}^{\#}$
$x = !r_0 \; r_0^{\#} r_{2i+1} r_{2i+1}^{\#}$
ACCEPT

5.Two equivalent finite machines with two pushdown stores

5.1 Formal Definition of a Finite Machine with Two Pushdown Stores

A Finite Machine with Two Pushdown Stores M over $\Sigma \cup \Gamma \cup \{B\}$ is a flow-diagram with one variable x, which may have as a value any word over $\Sigma \cup \Gamma \cup \{B\}$. Each statement in the flow diagram has one of the following forms:

START statement (exactly one)

x = word w
head(x): gives the head (leftmost letter) of the word x
tail(x): gives the tail of the word x (that is, x with the leftmost letter removed)
$\sigma \cdot x$: concatenates the letter σ and the word x
Λ = empty word (word with no letters)
y_1 = pushdown store 1
y_2 = pushdown store 2

The idea behind how the Finite Machine with Two Pushdown Stores operates in a few lines:

START: x = complete word w in input alphabet Σ and $y_1 = \Lambda$ and $y_2 = \Lambda \rightarrow$
x = tail(x) of word w in input alphabet Σ and $y_1 \leftarrow$ head(x) of word w in output alphabet Γ and $y_2 \leftarrow$
x = tail(x) of word w in input alphabet $\Sigma \rightarrow$
ACCEPT: x = Λ and y_1 = complete word w in output alphabet Γ and $y_2 = \Lambda$

START

x = $R_0, R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 = \Lambda$ and $y_2 = \Lambda$

head(x) = R_0

x = tail(x) = $R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_0 y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_0$ and $y_2 = R_0^{\#}R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = $R_0^{\#}$

x = tail(x) = $R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_0^{\#} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_0^{\#}, r_0$ and $y_2 = R_{2i+1}, R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = R_{2i+1}

x = tail(x) = $R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_{2i+1} y_1$. and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+1}r_0^{\#}, r_0$ and $y_2 = R_{2i+1}^{\#}R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = $R_{2i+1}^{\#}$

x = tail(x) = $R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_{2i+1}^{\#} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = R_{2i+2}, R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = R_{2i+2}

x = tail(x) = $R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_{2i+2} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+2}r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = R_{2i+2}^{\#}R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = $R_{2i+2}^{\#}$

x = tail(x) = $R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_{2i+2}^{\#} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+2}^{\#}, r_{2i+2}r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = $R_{2i+1}R_{2i+2}$

x = tail(x) = $R_{2i+1}R_{2i+2}{}^{\#}$

$y_1 \leftarrow r_{2i+1}r_{2i+2} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+1}r_{2i+2}r_{2i+2}^{\#}, r_{2i+2}r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = R_{2i+1}R_{2i+2}{}^{\#}$

head(x) = $R_{2i+1}R_{2i+2}{}^{\#}$

x = tail(x) = Λ

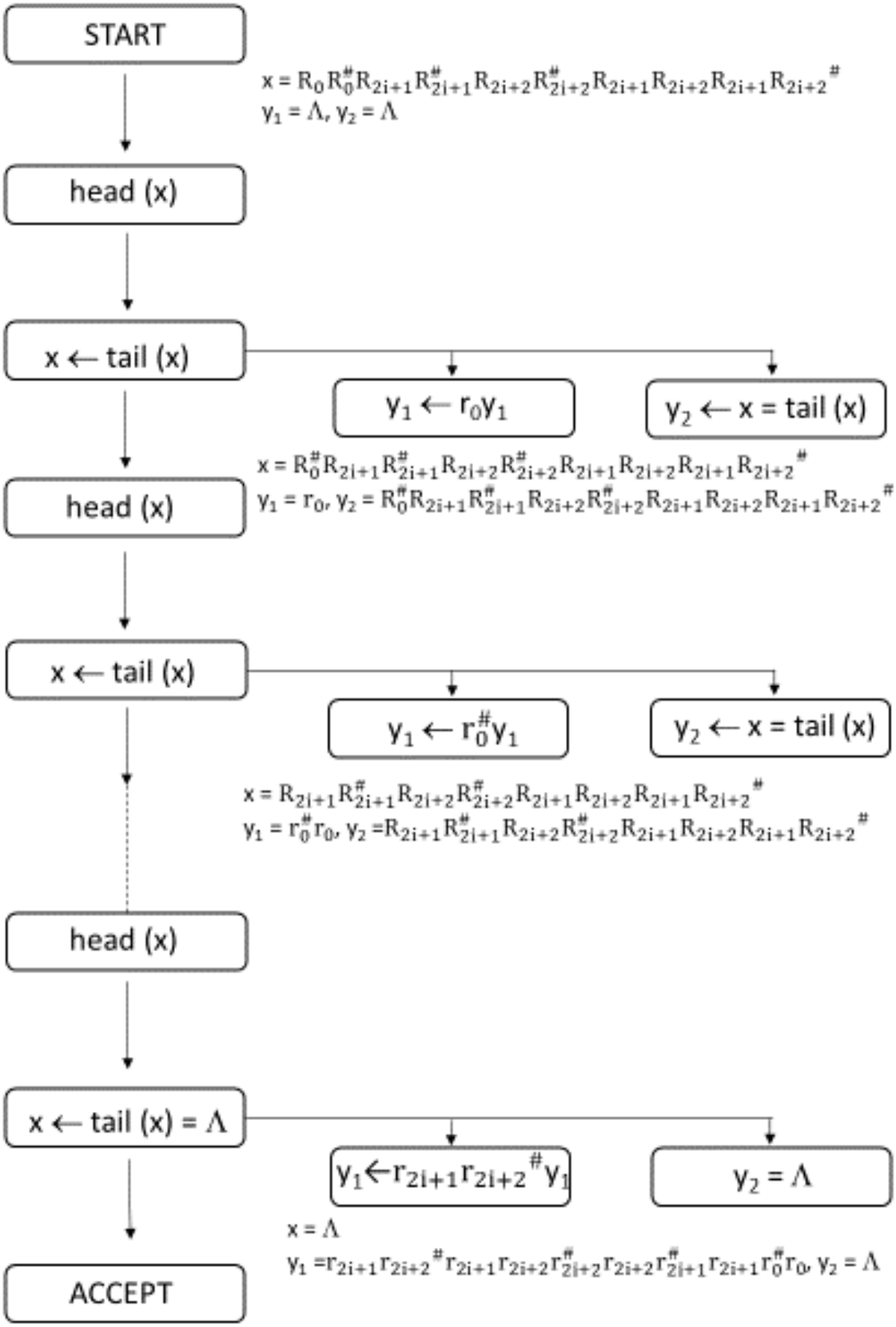
$y_1 \leftarrow r_{2i+1}r_{2i+2}{}^{\#} y_1$ and $y_2 \leftarrow$ x = tail(x)

$y_1 = r_{2i+1}r_{2i+2}{}^{\#}, r_{2i+1}r_{2i+2}r_{2i+2}^{\#}, r_{2i+2}r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = \Lambda$

ACCEPT

x = Λ

$y_1 = r_{2i+1}r_{2i+2}{}^{\#}, r_{2i+1}r_{2i+2}r_{2i+2}^{\#}, r_{2i+2}r_{2i+1}^{\#}, r_{2i+1}r_0^{\#}, r_0$ and $y_2 = \Lambda$


START
$x = R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}{}^{\#}$
$y_1 = \Lambda, y_2 = \Lambda$
head (x)
x ← tail (x)
$y_1 \leftarrow r_0 y_1$
$y_2 \leftarrow x = tail (x)$
$x = R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}{}^{\#}$
$y_1 = r_0, y_2 = R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}{}^{\#}$
head (x)
x ← tail (x)
$y_1 \leftarrow r_0^{\#} y_1$
$y_2 \leftarrow x = tail (x)$
$x = R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}{}^{\#}$
$y_1 = r_0^{\#} r_0, y_2 = R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}{}^{\#}$
head (x)
x ← tail (x) = Λ
$y_1 \leftarrow r_{2i+1} r_{2i+2}{}^{\#} y_1$
$y_2 = \Lambda$
$x = \Lambda$
$y_1 = r_{2i+1} r_{2i+2}{}^{\#} r_{2i+1} r_{2i+2} r_{2i+2}^{\#} r_{2i+2} r_{2i+1}^{\#} r_{2i+1} r_0^{\#} r_0, y_2 = \Lambda$
ACCEPT

START

x = $R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$

$y_1 = \Lambda$ and $y_2 = \Lambda$

head(x) = R_0

x = tail(x) = $R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$

$y_1 \leftarrow r_0 y_1$ and $y_2 \leftarrow$ tail(x)

$y_1 = r_0$ and $y_2 = R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$

head(x) = $R_0^{\#}$

x = tail(x) = $R_{2i+1} R_{2i+1}^{\#}$

$y_1 \leftarrow r_0^{\#} y_1$ and $y_2 \leftarrow$ tail(x)

$y_1 = r_0^{\#} r_0$ and $y_2 = R_{2i+1} R_{2i+1}^{\#}$

head(x) = R_{2i+1}

x = tail(x) = $R_{2i+1}^{\#}$

$y_1 \leftarrow r_{2i+1} y_1$. and $y_2 \leftarrow$ tail(x)

$y_1 = r_{2i+1} r_0^{\#} r_0$ and $y_2 = R_{2i+1}^{\#}$

head(x) = $R_{2i+1}^{\#}$

x = tail(x) = Λ

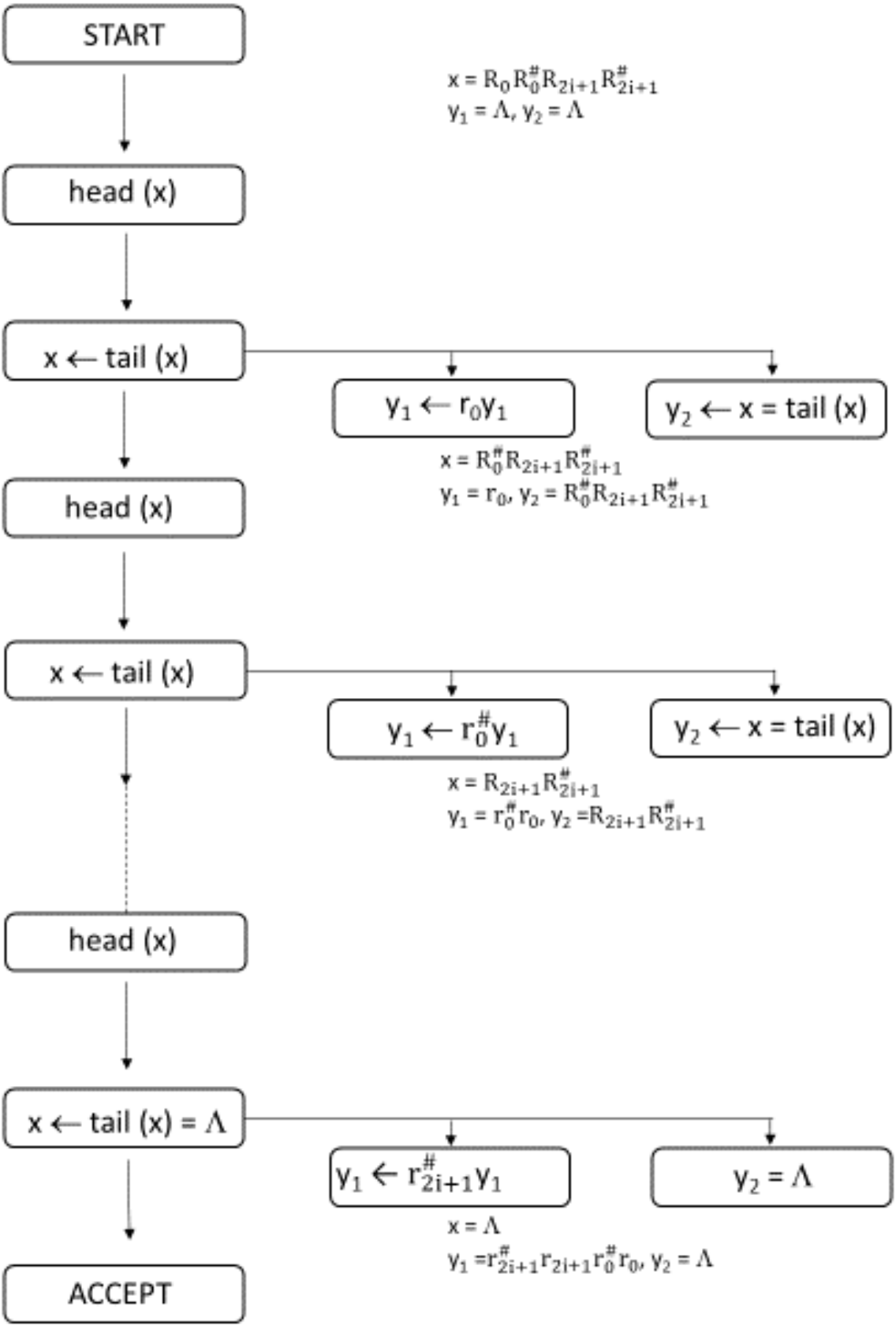
$y_1 \leftarrow r_{2i+1}^{\#} y_1$. and $y_2 \leftarrow$ tail(x)

$y_1 = r_{2i+1}^{\#} r_{2i+1} r_0^{\#} r_0$ and $y_2 = \Lambda$

ACCEPT

x = Λ

$y_1 = r_{2i+1}^{\#} r_{2i+1} r_0^{\#} r_0$ and $y_2 = \Lambda$


START
head (x)
x ← tail (x)
head (x)
x ← tail (x)
head (x)
x ← tail (x) = Λ
ACCEPT
$x = R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$
$y_1 = \Lambda, y_2 = \Lambda$
$y_1 \leftarrow r_0 y_1$
$y_2 \leftarrow x = tail\ (x)$
$x = R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$
$y_1 = r_0, y_2 = R_0^{\#} R_{2i+1} R_{2i+1}^{\#}$
$y_1 \leftarrow r_0^{\#} y_1$
$y_2 \leftarrow x = tail\ (x)$
$x = R_{2i+1} R_{2i+1}^{\#}$
$y_1 = r_0^{\#} r_0, y_2 = R_{2i+1} R_{2i+1}^{\#}$
$y_1 \leftarrow r_{2i+1}^{\#} y_1$
$y_2 = \Lambda$
$x = \Lambda$
$y_1 = r_{2i+1}^{\#} r_{2i+1} r_0^{\#} r_0, y_2 = \Lambda$

6.Linear Bounded Automaton

Any set of input words x accepted by a non-deterministic linear bounded automaton (LBA) is accepted by a deterministic Turing Machine [35]. A linear bounded automaton (lba) is a nondeterministic single tape Turing Machine that never leaves the cells on which the input symbols were placed. A nondeterministic lba's computational space requirement can be a quadratic function of the length of the input word x: $c_2|x|^2 + c_1|x| + c_0$, where c_2, c_1, c_0 are real number constants. Intermediate sentential forms are never longer than the length of the input word x. This means that the maximal amount of tape space required for an lba to compute the input word is a bounded function of the length of the input word |x|. This is less than the amount of tape required for the deterministic Turing machine for the same word. The exponential amount of tape required by the deterministic Turing machine was shown earlier in the tape diagrams of section 3. As mentioned before, it is a known result that the linear bounded automaton and the deterministic Turing machine accept the same languages [35]. Whether a machine is deterministic or nondeterministic is decided by whether the output for each input is unique or nonunique. The crucial difference between the nondeterministic lba and the deterministic lba is that the deterministic machine is programmed by a transition function and the nondeterministic machine is programmed by a transition relation. The differences are down to the distinctions between a function and a relation. A function has a unique output for each input. A relation has more than one output for each input.

Formal Definition of Nondeterministic Linear Bounded Automaton (lba)

lba = (Q, Σ, Γ, δ, q_0, F) where

Q = set of states
Σ = set of input symbols ⊆ Γ
Γ = set of output symbols plus left end marker £ and right end marker $
$\Delta = Q \times (\Gamma \setminus \{£, \$\}) \rightarrow 2^{Q \times \Gamma \times \{-, +, 0\}}$
($2^{Q \times \Gamma \times \{-, +, 0\}}$ = set of subsets of Q × Γ x {-, +, 0} where {-} means move left, {+} means move right, {0} means do not move)
q_0 = start state
F = set of final states

Formal Definition of Deterministic Linear Bounded Automata (lba) that accepts the URS language

lba = (Q, Σ, Γ, δ, q_0, F) where
$Q = \{q_0, q_1, q_2, q_3, q_4, q_f\}$
Σ = {a, b, c} ⊆ Γ
Γ = {a, b, c, A, B, C, £, $}
$\delta = Q \times (\Gamma \setminus \{£, \$\}) \rightarrow Q \times \Gamma \times \{+\}$
q_0 = start state
$F = \{q_2, q_4\}$

A Deterministic Linear Bounded Automaton (lba) has one input string w_2 that works for both the n = 2i +1 or n = 2i + 2 distinct parity cycles of the URS (unlike the other machines) but both even and odd computations will be described below for completeness. It should be understood that dividing the URS process into even and odd parts is done for instructional purposes to show how the URS works on a machine but in nature the combined even and odd components of the URS happen simultaneously in the infinite product.

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_{2i+1}, R_{2i+1}^{\#})(R_{2i+2}, R_{2i+2}^{\#})(R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}^{\#}) = ab_nc_nd_n.$$

$w_2 = R_0 R_0^{\#} R_{2i+1} R_{2i+1}^{\#} R_{2i+2} R_{2i+2}^{\#} R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}^{\#}$
$= (R_0 R_0^{\#})(R_{2i+1} R_{2i+1}^{\#})(R_{2i+2} R_{2i+2}^{\#})(R_{2i+1} R_{2i+2} R_{2i+1} R_{2i+2}^{\#})$
$= ab_n c_n d_n$

Output form of words w_1 and w_2

$w_2 = r_0 r_0^{\#} r_{2i+1} r_{2i+1}^{\#} r_{2i+2} r_{2i+2}^{\#} r_{2i+1} r_{2i+2} r_{2i+1} r_{2i+2}^{\#}$
$= (r_0 r_0^{\#})(r_{2i+1} r_{2i+1}^{\#})(r_{2i+2} r_{2i+2}^{\#})(r_{2i+1} r_{2i+2} r_{2i+1} r_{2i+2}^{\#})$
$= AB_n C_n D_n$

where

$a = R_0, R_0^{\#}$
$b_n = R_{2i+1}, R_{2i+1}^{\#}$
$c_n = R_{2i+2}, R_{2i+2}^{\#}$
$d_n = R_{2i+1} R_{2i+2}, R_{2i+1} R_{2i+2}{}^{\#}$

$A = r_0 r_0^{\#}$
$B_n = r_{2i+1} r_{2i+1}^{\#}$
$C_n = r_{2i+2} r_{2i+2}^{\#}$
$D_n = r_{2i+1} r_{2i+2}, r_{2i+1} r_{2i+2}{}^{\#}$

Transition Function δ of the deterministic Linear Bounded Automaton that accepts the URS language

1. $\delta(q_0, a) = (q_1, A, +)$
2. $\delta(q_1, b_n) = (q_2, B_n, +)$
3. $\delta(q_2, c_n) = (q_3, C_n, +)$
4. $\delta(q_3, d_n) = (q_4, D_n, +)$

Deterministic Linear Bounded Automaton that accepts the URS word form $ab_n c_n d_n$ in the odd parity cycle for n = 2i +1

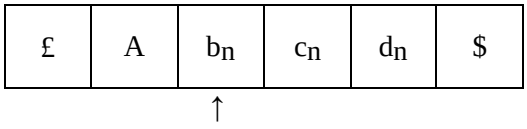
1. $\delta(q_0, a)$

£	a	b_n	c_n	d_n	$
	↑				

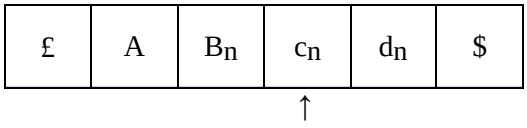
1. $\delta(q_0, a) = (q_1, A, +)$

£	A	b_n	c_n	d_n	$
		↑			

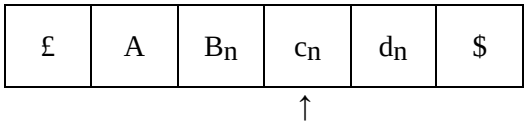
2. $\delta(q_1, b_n)$



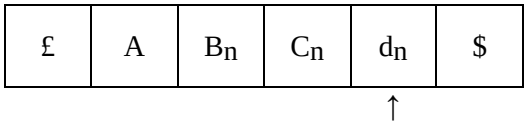
2. $\delta(q_1, b_n) = (q_2, B_n, +)$



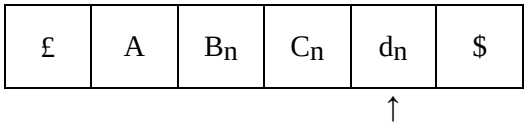
3. $\delta(q_2, c_n)$



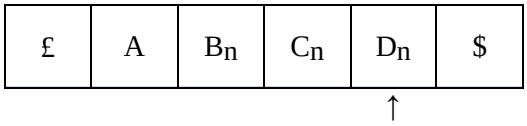
3. $\delta(q_2, c_n) = (q_3, C_n, +)$



4. $\delta(q_3, d_n)$



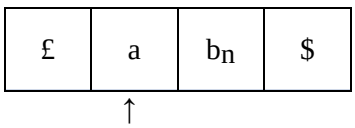
4. $\delta(q_3, d_n) = (q_4, D_n, +)$



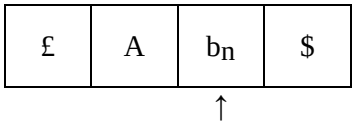
$(q_4, \$)$ = Accept

Deterministic Linear Bounded Automaton that accepts the URS word form ab_n in the even parity cycle for $n = 2i + 2$

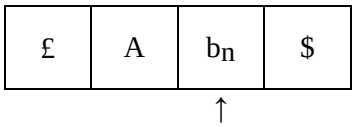
1. $\delta(q_0, a)$



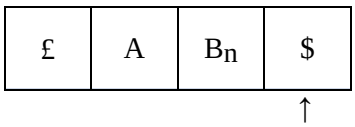
1. $\delta(q_0, a) = (q_1, A, +)$



2. $\delta(q_1, b_n)$



2. $\delta(q_1, b_n) = (q_2, B_n, +)$



$(q_2, \$)$ = Accept

A recursive language is decidable. This means that the machine always halts upon acceptance or nonacceptance of the input word. The machine will not loop on any input word that is not accepted.

$$\prod_{i=0}^{\infty}(R_0, R_0^{\#})(R_{2i+1}, R_{2i+1}^{\#})(R_{2i+2}, R_{2i+2}^{\#})(R_{2i+1}R_{2i+2}, R_{2i+1}R_{2i+2}^{\#}) = ab_nc_nd_n$$

Instead of taking an exponential amount of tape space derived from an exponential function of the length of the input word x, as computed on a deterministic Turing machine, the word computed on a linear bounded automaton takes only a linear amount of tape space; this is derived from a linear function of the length of the input word x. The universal rewrite system, as realized on a linear bounded automaton, is then constructed from the totality of all the linear word blocks for each distinct n.

7.Conclusion

The Rowlands-Diaz universal alphabet and rewrite system appears to provide a meta-pattern for mathematics and science, extending from quantum mechanics and particle physics up to biology and even consciousness. Here, we have shown that it is compatible with computational and formal language theory, using a deterministic Turing machine, a Post machine, a Finite machine with two pushdown stores, and a linear bounded automaton, which means that we can apply computational theory and practice directly to all these systems. Mathematics gives us a guide to the patterns of Nature, rather than the meta-pattern, because it is structured on exact repetition, e.g. of the integer series 1, 2, 3, 4, 5 …, rather than uniqueness. The meta-pattern is a unique birth-ordering in an infinite process, and any reproduction of it will only ever be finite. In addition, mathematics only reads (rewrite rules 2.0 conserve rules (1), (2) and (3)), bounded by the zero cardinalities, each of which creates a new algebra (rewrite rules 2.0 create rules (0), (4) and (5)). Physics requires a unique sequence of events, never repeated, which means that, to simulate it, we must use machines that can write as well as read, as in the cases discussed here, which makes its requirements different from those of much mathematically-based computational theory, which is frequently concerned with pure reading automata. Physics has a special system for obtaining zeros using nilpotents, which are each unique and hence at its most basic level can describe a unique birthordering. We see that the mathematics most appropriate for this development is a form of Clifford algebra. This does not prevent us from developing alternative mathematical ideas, such as the higher

Cayley-Dickson algebras, but they cannot be used to describe the system itself. The development of a computational representation based on the most general devices that can be imagined gives a powerful indication that the Rowlands-Diaz universal rewrite system provides the most general, generic and efficient description so far known for Nature's most fundamental processes.

References

1. Rowlands P and Diaz B 2002 A universal alphabet and rewrite system arXiv:cs.OH/0209026
2. Diaz B and Rowlands P 2005 A computational path to the nilpotent Dirac equation *International Journal of Computing Anticipatory Systems* 16 203-18
3. Diaz B and Rowlands P 2006 The infinite square roots of –1 *International Journal of Computing Anticipatory Systems* 19 229-235
4. Rowlands P 2007 Zero to Infinity: The Foundations of Physics (World Scientific)
5. Rowlands P 2010 Mathematics and Physics as Emergent Aspects of a Universal Rewrite System *International Journal of Computing Anticipatory Systems* 25 115-131
6. Rowlands P 2014 *The Foundations of Physical Law* (World Scientific)
7. Bateson G 1979 *Mind and Nature: A Necessary Unity* (Bantam Books Toronto)
8. Marcer P and Rowlands P 2015 Information, Bifurcation and Entropy in the Universal Rewrite System *International Journal of Computing Anticipatory Systems* 27 203-215
9. Rowlands P 2001 A foundational approach to physics, arXiv:physics/0106054
10. Rowlands P 2003 The nilpotent Dirac equation and its applications in particle physics arXiv:quant-ph/0301071
11. Rowlands P 2004 Symmetry breaking and the nilpotent Dirac equation *AIP Conference Proceedings* 718 102-115
12. Rowlands P 2005 Removing redundancy in relativistic quantum mechanic arXiv.org:physics/0507188
13. Rowlands P 2008 What is vacuum? arXiv:0810.0224
14. Rowlands P 2010 Physical Interpretations of Nilpotent Quantum Mechanics, arXiv: 1004.1523
15. Rowlands P 2013 Space and Antispace in Amoroso R L Kauffman L H and Rowlands P (eds.), *The Physics of Reality Space, Time, Matter, Cosmos* (World Scientific) 29-37
16. Rowlands P 2013 Symmetry in Physics from the Foundations *Symmetry* 24 41-56
17. Rowlands P 2017 How symmetries become broken, *Symmetry* 28 244-254
18. Rowlands P 2018 Idempotent or nilpotent? *AIP Conference Proceedings* 2046 020081
19. Marcer P and Rowlands P 2017 Nilpotent Quantum Mechanics: Analogs and Applications *Frontiers in Physics* 5 article 28 1-8, 2017
20. Rowlands P 2019 Constructing the Standard Model fermions? *Journal of Physics Conference Series* 1251 012004
21. Rowlands P and Rowlands S 2019 Are octonions necessary to the Standard Model? *Journal of Physics Conference Series* 1251 012044
22. Hill V J and Rowlands P 2008 Nature's code *AIP Conference Proceedings* 1051 117-126
23. Hill V J and Rowlands P 2010 Nature's Fundamental Symmetry Breaking *International Journal of Computing Anticipatory Systems* 25 144-159
24. Hill V J and Rowlands P 2010 The Numbers of Nature's Code *International Journal of Computing Anticipatory Systems* 25 160-175
25. Hill V J and Rowlands P 2015 A mathematical representation of the genetic code in Amoroso R L Kauffman L H and Rowlands P (eds.) *Unified Field Mechanics Natural Science Beyond the Veil of Spacetime Proceedings of the IX Symposium Honoring Noted French Mathematical Physicist Jean-Pierre Vigier* (World Scientific) 553-559
26. Marcer P Mitchell E Rowlands P and Schempp W Zenergy: The 'phaseonium' of dark energy that fuels the natural structures of the Universe 2005 *International Journal of Computing Anticipatory Systems* 16 189-202
27. Marcer P and Rowlands P 2007 How intelligence evolved? in Quantum Interaction, Papers from

the AAAI Spring Symposium, Technical Report SS-07-08, 2007
28. Marcer P and Rowlands P 2010 Further Evidence in Support of the Universal Nilpotent Grammatical Computational Paradigm of Quantum Physics *AIP Conference Proceedings* 1316 90-101
29. Marcer P and Rowlands P 2010 The 'Logic' of Self-Organizing Systems *AAAI Technical Reports* 2010-08-020
30. Marcer P and Rowlands P 2010 The Grammatical Universe and the Laws of Thermodynamics and Quantum Entanglement *AIP Conference Proceedings* 1303 161-167
31. Marcer P and Rowlands P 2013 A Computational Unification of Scientific Law: Spelling out a Universal Semantics for Physical Reality in Amoroso R L Kauffman L H and Rowlands P (eds.) *The Physics of Reality Space, Time, Matter, Cosmos* (World Scientific)
32. Marcer P and Rowlands P 2015 Computational tractability – beyond Turing? in Amoroso R L Kauffman L H and Rowlands P (eds.) *Unified Field Mechanics Natural Science Beyond the Veil of Spacetime* (World Scientific) 33-38
33. Grumberg O Kupferman O and Sheinvald S Variable Automata over Infinite Alphabets 2010 in Dediu A-H, Fernau H and Martin-Vide (eds) LATA 2010 LNCS 6031 561-572
34. Manna Z 1974 Mathematical Theory of Computation (McGraw-Hill)
35. Hopcroft J E and Ullman J D 1969 Formal Language Theory and their Relation to Automata (Addison-Wesley)