WPI-CS-TR-95-2 August 1995

Self-Modifying Finite Automata — Basic Definitions and Results

by

Roy S. Rubinstein

and

John N. Shutt

Computer Science Technical Report Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Self-Modifying Finite Automata — Basic Definitions and Results

Roy S. Rubinstein & John N. Shutt
roy@cs.wpi.edu jshutt@cs.wpi.edu
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609

August 1995

Abstract

We formally define the Self-Modifying Finite Automaton (SMFA), a model of computation introduced in [RS93, RS94, RS95], as a subclass of a new more general model, the Self-Modifying Automaton (SMA). SMAs are similar to standard finite automata, but changes to the transition set are allowed during a computation. An SMFA is constrained in that it can have only finitely many different modification instructions, and the effect of each instruction must be computable.

1 Introduction

The Self-Modifying Finite Automaton (SMFA) is a model of computation introduced in [RS93, RS94, RS95]. SMFAs are similar to standard finite automata, but changes to the transition set are allowed during a computation. While retaining much of the simplicity of finite automata, SMFAs have greater power. A weak form of SMFAs has been shown to accept the class of metalinear languages, as well as some other classes of context-free and even non-context-free languages [RS94, RS95].

The treatment of SMFAs in previous papers, although precise and sufficiently detailed for initial research, has not been accompanied by mathematically rigorous definitions. This paper provides the missing formalism. Initially, a still more general model of computation, the Self-Modifying Automaton (SMA), is defined. A variety of constraints are then used to define subclasses of SMAs. SMFAs comprise one of these subclasses.

The main results presented in this paper are that the general SMA model has arbitrary computational power (that is, arbitrary languages are SMA-decidable); all

SMFA languages are recursively enumerable; and a relatively constrained subclass of SMFAs is Turing-powerful.

2 SMAs

Throughout this document, for any alphabet X, let $X_\lambda = X \cup \{\lambda\}$. Equivalently, $X_\lambda = \{w \in X^* \text{ such that } |w| \leq 1\}$.

Definition 2.1 (SMA basis)

An *SMA basis* is a tuple $B = \langle \Sigma, Q_0, R, A \rangle$, where Σ, Q_0, and R are pairwise disjoint finite sets of symbols, A is a set, and Q_0 and A are nonempty. Σ is called the *input alphabet*, Q_0 the set of *predefined states*, R the set of *registers*, and A the set of *modification actions*, of B.

The domain of *states* over B is $Q = Q_0 \cup (R \times \mathbb{N})$. That is, a state is either a predefined state, or an ordered pair $\langle r, k \rangle$ of a register r with a nonnegative integer k. The intent is that $\langle r, k \rangle$ is the k^{th} state created via register r.

The domain of *transitions* over B is $D = Q \times \Sigma_\lambda \times Q \times A$. That is, a transition is a tuple $\langle q, s, q', a \rangle$, where $q, q' \in Q$ are the source and destination states of the transition, $s \in \Sigma_\lambda$ is the input read, and $a \in A$ is the modification action taken.

The domain of transitions between predefined states over B is $D_0 = Q_0 \times \Sigma_\lambda \times Q_0 \times A$. Note that $D_0 \subseteq D$.

The function $\mathit{states} : \mathcal{P}(D) \to \mathcal{P}(Q)$ is defined by

$$\mathit{states}(X) \quad = \quad \{q \mid \exists s, p, a \text{ such that } \langle q, s, p, a \rangle \in X \text{ or } \langle p, s, q, a \rangle \in X\}$$

That is, if X is a set of transitions, then $\mathit{states}(X)$ is the set of every state that is the source or destination of some transition in X. (Functions are total unless otherwise stated.)

The domain of *configurations* over B is $C = Q \times (R \to \mathbb{N}) \times \mathcal{P}(D) \times \Sigma^*$. That is, a configuration is a tuple $\langle q, \rho, \delta, w \rangle$, where $q \in Q$ is the current state, $\rho : R \to \mathbb{N}$ is a function mapping each register to a nonnegative integer (the "current value" of that register), $\delta \in \mathcal{P}(D)$ is the current set of transitions, and w is the remaining input. The *null register function* over B is the unique function $\rho_0 : R \to \{0\}$; that is, $\rho_0(r) = 0$ for all $r \in R$.

Given any configuration $c = \langle q, \rho, \delta, w \rangle \in C$ and any transition $d = \langle p, s, p', a \rangle \in D$, d is *allowed* from c iff $q = p$, $d \in \delta$, and s is a prefix of w. A pair $\langle c, d \rangle \in C \times D$ is *allowable*, sometimes denoted by the predicate $\mathit{allow}(c, d)$, iff d is allowed from c. The domain of all allowable pairs is denoted $\mathit{allow}(C, D)$.

The domain of *modification functions* over B is $\mathcal{A} = \mathit{allow}(C, D) \to \mathcal{P}(D)$. The intent is that, when taking an allowed transition $d \in D$ from a configuration $c \in C$, a modification function $\alpha \in \mathcal{A}$ determines a set $\alpha(c, d)$ of transitions to be added to or deleted from the transition set of c.

The *null modification function* over B is the unique function $\alpha_0 \in \mathcal{A}$ such that $\alpha_0(c,d) = \{\}$ for all allowable $\langle c, d\rangle$. $\Box$

Lemma 2.2 (Allowable pairs)
If B is an SMA basis, then there exists an allowable pair over B. $\Box$

Proof. Suppose $B = \langle \Sigma, Q_0, R, A\rangle$ is an SMA basis. Since Q_0 and A must be nonempty, suppose $q \in Q_0$ and $a \in A$ are elements of these sets. Let $d = \langle q, \lambda, q, a\rangle$, and $c = \langle q, \rho_0, \{d\}, \lambda\rangle$. (Recall that ρ_0 is the null register function over B.) Then $d \in D$, $c \in C$, and $\langle c, d\rangle$ is an allowable pair. $\Box$

Definition 2.3 (SMA)
A *self-modifying automaton* (SMA) with basis $B = \langle \Sigma, Q_0, R, A\rangle$ is a tuple

$$M \quad = \quad \langle \Sigma, Q_0, R, A, S, F, \delta_0, \alpha, \alpha'\rangle$$

such that $S \in Q_0$, $F \subseteq Q_0$, $\delta_0 \subseteq D_0$, and $\alpha, \alpha' \in \mathcal{A}$. S is called the *start state*, F the set of *final states*, δ_0 the *initial set of transitions*, α the *addition function*, and α' the *deletion function*, of M.

The initial configuration of M on input $w \in \Sigma^*$ is $c_w = \langle S, \rho_0, \delta_0, w\rangle \in C$.

The *step function* for M is a partial function $step : allow(C, D) \to C$ as follows. Suppose $\langle c, d\rangle$ is an allowable pair; then $step(c, d) = \langle q', \rho', \delta', w'\rangle$ whenever all these values are well-defined as follows.

$$\begin{aligned} c &= \langle q, \rho, \delta, w\rangle \\ d &= \langle q, s, q', a\rangle \\ w &= sw' \\ \delta' &= (\delta - \alpha'(c,d)) \cup \alpha(c,d) \\ \forall r \in R, \quad \rho'(r) &= max(\{\rho(r)\} \cup \{k \mid \langle r, k\rangle \in states(\alpha(c,d))\}) \end{aligned}$$

(The value of $\rho'(r)$ is undefined whenever the set of integers in its definition has no upper bound; see Lemma 2.4, below.)

The binary computation step relation $\vdash_M$ on C is defined as $c \vdash_M c'$ iff there exists d such that $step(c, d) = c'$. The reflexive transitive closure of $\vdash_M$ is denoted $\vdash_M^*$. For configuration $c \in C$, c is *reachable* iff there exists some $w \in \Sigma^*$ such that $c_w \vdash_M^* c$. (Recall that c_w is the initial configuration of M on input w.) For configurations $c, c' \in C$, c' is *reachable from* c iff $c \vdash_M^* c'$.

A configuration $\langle q, \rho, \delta, w\rangle$ is *accepting* iff $q \in F$ and $w = \lambda$. Suppose $w \in \Sigma^*$. Then w is *accepted* by M iff there exists an accepting configuration that is reachable from c_w.

The *language accepted* by M is $L(M) = \{w \mid w \text{ is accepted by } M\}$. $\Box$

Lemma 2.4 (SMA Step functions)

Suppose M is an SMA with addition function α, and $\langle c, d\rangle$ an allowable pair on M. Then $step(c, d)$ is defined iff $states(\alpha(c, d))$ is finite.

Further, $step$ is a total function iff for *all* allowable $\langle c, d\rangle$, $states(\alpha(c, d))$ is finite. $\Box$

Proof. Suppose $M = \langle \Sigma, Q_0, R, A, S, F, \delta_0, \alpha, \alpha'\rangle$ is an SMA, and $\langle c, d\rangle$ an allowable pair. Let $X = states(\alpha(c, d))$. Since Q_0 must be finite, X is infinite iff $X - Q_0 = X \cap (R \times \mathbb{N})$ is infinite. Since R is finite, $X \cap (R \times \mathbb{N})$ is infinite iff there exists some $r \in R$ such that $X \cap (\{r\} \times \mathbb{N})$ is infinite. But $step(c, d)$ is defined iff all sets $X \cap (\{r\} \times \mathbb{N})$ are finite. Therefore $step(c, d)$ is defined iff X is finite.

The second half of the lemma follows immediately. $\Box$

In effect, the lemma says that only finitely many states may be added by any single computation step. Note, however, that an infinite number of *transitions* may be added with impunity, provided that they all together involve only a finite set of states.

One could, of course, generalize the definitions to allow for adding infinite numbers of states, by amending the domain of register values from $\mathbb{N}$ to $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$. Since even an infinite subset of $\mathbb{N}_\infty$ has a least upper bound in $\mathbb{N}_\infty$, the step function is total. However, this complication is entirely unnecessary, because (1) the model is already arbitrarily powerful without it (Theorem 5.1), and (2) in the special case that will be of principal concern here, the question of infinite state sets simply cannot arise (Corollary 4.6).

Definition 2.5 (Well-formed configuration)

Suppose M is an SMA, and $c = \langle q, \rho, \delta, w\rangle \in C$. The set of *valid states* in c is

$$Q_0 \cup \{\langle r, k\rangle \mid r \in R \text{ and } 0 \leq k \leq \rho(r)\}$$

c is *well-formed* iff all $p \in states(\delta)$ are valid in c. $\Box$

Note that c well-formed implies $states(\delta)$ finite.

Lemma 2.6 (Well-formed configurations)

Suppose M is an SMA. Then every reachable configuration of M is well-formed. $\Box$

Proof. Suppose M is an SMA, and $c = \langle q, \rho, \delta, w\rangle$ is a reachable configuration of M. Proceed by induction on the (finite) number of computation steps needed to reach c.

Suppose c is an initial configuration. Then δ is the initial set of transitions of M. There are no states of the form $\langle r, k\rangle$ in $states(\delta)$, so c is well-formed.

Suppose c is not an initial configuration, and the theorem holds for all configurations reachable in fewer steps than c. Since c is reachable, there must exist an allowable pair $\langle c', d' \rangle$ such that $step(c', d') = c$ and c' is reachable in fewer steps than c. Let $c' = \langle q', \rho', \delta', w' \rangle$, and let α be the addition function of M. Then $\delta \subseteq \delta' \cup \alpha(c', d')$.

Suppose further that $\langle r, k \rangle \in states(\delta)$; either $\langle r, k \rangle \in states(\delta')$ or $\langle r, k \rangle \in states(\alpha(c', d'))$ (or both). If $\langle r, k \rangle \in states(\delta')$, then by inductive hypothesis, $k \leq \rho'(r)$; and by definition of the step function, $\rho'(r) \leq \rho(r)$. On the other hand, if $\langle r, k \rangle \in states(\alpha(c', d'))$, then again by definition of the step function, $k \leq \rho(r)$. □

3 Classes of modification functions

Most distinctions between classes of SMAs will be based on the nature of their modification functions. (Even the distinction between finite and nonfinite SMAs will be based partly on modification functions; see Definition 4.5.)

Definition 3.1 (Separability by action)
Suppose $B = \langle \Sigma, Q_0, R, A \rangle$ is an SMA basis, $\alpha \in \mathcal{A}$ a modification function over B, and $X \subseteq \mathcal{A}$ a class of modification functions over B. Then α is *separable by action into* X iff there exists a function $f : A \to X$ such that

$$\alpha(c, \langle p, s, q, a \rangle) = (f(a))(c, \langle p, s, q, a \rangle)$$

□

Note that the decomposition need not be unique: Given B, α, and X, there may be more than one f satisfying the condition.

Separation by action allows complex modification functions to be understood in terms of classes of simpler functions.

3.1 ρ-separable

Before applying the principle of separation by action, one must have a previously defined class of modification functions. The following definition provides such a class.

Definition 3.2 (ρ-relative generator)
Suppose $B = \langle \Sigma, Q_0, R, A \rangle$ is an SMA basis. Let $g : (R \to \mathbb{N}) \to (Q \to Q)$, $h : (R \to \mathbb{N}) \to (D \to D)$, and $G : \mathcal{P}(D) \to \mathcal{A}$ be the functions

$$\begin{aligned}
(g(\rho))(q) &= \begin{cases} q & \text{if } q \in Q_0 \\ \langle r, \rho(r) + k \rangle & \text{if } q = \langle r, k \rangle \end{cases} \\
(h(\rho))(\langle p, s, q, a \rangle) &= \langle (g(\rho))(p), s, (g(\rho))(q), a \rangle \\
(G(Z))(\langle p, \rho, \delta, w \rangle, d) &= \{(h(\rho))(z) \mid z \in Z\}
\end{aligned}$$

Then G is the *ρ-relative generator* for B.

A modification function is *ρ-relative* iff it has the form $G(Z)$ for some $Z \subseteq D$. For any $k \in \mathbb{N}$, $G(Z)$ is ρ-relative *with depth k* iff $states(Z) \subseteq Q_0 \cup (R \times \{j \mid j \leq k\})$. $\Box$

Theorem 3.3 (ρ-relative generator)

Suppose $B = \langle \Sigma, Q_0, R, A \rangle$ is an SMA basis, G is the ρ-relative generator for B, $X, Y \subseteq D$, and $\langle c, d \rangle$ is an allowable pair. Then $G(X)(c, d) \subseteq G(Y)(c, d)$ iff $X \subseteq Y$, and $G(X)(c, d) = G(Y)(c, d)$ iff $X = Y$. $\Box$

Proof. Suppose B etc. as in the theorem. Let $c = \langle q, \rho, \delta, w \rangle$, and let functions g and h be as in Definition 3.2. We have:

$$\begin{array}{rcl} G(X)(c,d) & = & \{h(\rho)(x) \mid x \in X\} \\ G(Y)(c,d) & = & \{h(\rho)(y) \mid y \in Y\} \end{array}$$

Note that $g(\rho) : Q \to Q$ and $h(\rho) : D \to D$ are injections (i.e., one-to-one functions). The desired results follow immediately. $\Box$

Definition 3.4 (ρ-separable, ρ-normal)

Suppose B is an SMA basis, and $\alpha \in \mathcal{A}$ is a modification function over B. Then α is *ρ-separable* iff α is separable by action into the class of functions in $\mathcal{A}$ that are ρ-relative. Further, α is *ρ-normal* iff α is separable by action into the class of functions in $\mathcal{A}$ that are ρ-relative with depth 1. $\Box$

3.2 Self-separable

Non-ρ-separable modification functions are rarely of interest. The most prominent exception is the class of self-separable modification functions.

Definition 3.5 (Self-separable)

Suppose B is an SMA basis, and $\alpha \in \mathcal{A}$. Let $X \subset \mathcal{A}$ be the class $X = \{\alpha_0, \alpha_I\}$, where $\alpha_I(c, d) = d$ for all allowable $\langle c, d \rangle$. Then α is *self-separable* iff it is separable by action into X. $\Box$

3.3 Finite-order

Whereas separation by action isolates each individual action, the concept of finite-order modification function addresses the way different actions relate to each other. For convenience, let $action(d)$ denote the modification action a of any transition $d = \langle q, s, p, a \rangle \in D$.

Definition 3.6 (Order of a modification function)

Suppose $B = \langle \Sigma, Q_0, R, A \rangle$ is an SMA basis, and $\alpha \in \mathcal{A}$. Let $\triangleright$ be the following binary relation on A. $a \triangleright a'$ iff there exist $c \in C$ and $d, d' \in D$ such that $action(d) = a$, $action(d') = a'$, and $d' \in \alpha(c, d)$.

Suppose $a \in A$. For every $k \in \mathbb{N}$, *a has order k in α* iff there is no $a' \in A$ such that $a \triangleright^{k+1} a'$; in other words, there does not exist a sequence of actions $a_0, \cdots a_k \in A$ such that $a \triangleright a_0$ and, for all $0 \leq j < k$, $a_j \triangleright a_{j+1}$.

α has order k iff all actions $a \in A$ have order k in α.

$a \in A$ is *unordered* in α iff there exists $k \geq 1$ such that $a \triangleright^k a$. α is unordered iff some $a \in A$ is unordered in α.

$a \in A$ has *infinite order* in α iff a is ordered (i.e. not unordered) but does not have finite order in α. α has infinite order iff α is ordered but does not have finite order. $\square$

As defined here, if a has order k in α, then a also has order j in α for every $j > k$. Similarly, if α has order k then α has order j for every $j > k$.

Note that $a \in A$ has order zero in α iff there is no $a' \in A$ such that $a \triangleright a'$. α has order zero iff it is the null modification function, $\alpha = \alpha_0$.

The modification function in the proof of Theorem 5.1 has infinite order. The modification function in the proof of Theorem 5.3 is unordered. See also Corollary 4.7.

3.4 Notation

Because ρ-normal modification functions are so common in the theory of SMFAs, it is worthwile to establish a convenient notation for them. The following is a generalization of the notation used elsewhere for SMFA actions [RS93, RS94, RS95].

Convention 3.7 (Notation)

Suppose $B = \langle \Sigma, Q_0, R, A \rangle$ is an SMA basis, and $\alpha \in \mathcal{A}$ is ρ-normal. Let $f : A \to \mathcal{A}$ separate α by action into ρ-relative modification functions; let G be the ρ-relative generator for B, and let $g : A \to \mathcal{P}(D)$ be the unique function such that $G \circ g = f$. The following notation may be used to represent modification actions.

- A non-zeroth-order modification action $a \in A$ with finite $g(a) = \{d_1, \cdots d_n\} \subseteq D$ may be represented by

  $\underline{\text{verb}}\ \{\underline{d_1}, \cdots \underline{d_n}\}$

  where $\underline{\text{verb}}$ is either **add** or **delete**, depending on the usage of α, and the $\underline{d_k}$ are the representations of the transitions d_k according to the following rules. If $n = 1$, the enclosing set braces $\{\}$ may be omitted.

- A zeroth-order transition $d = \langle p, s, q, a \rangle$ is represented by $\underline{p} \xrightarrow{s} \underline{q}$, where $\underline{p}$ and $\underline{q}$ are the representations of p and q.

- A higher-order transition $d = \langle p, s, q, a\rangle$ is represented by $\underline{p} \xrightarrow{s/\underline{a}} \underline{q}$, where $\underline{p}$, $\underline{q}$, and $\underline{a}$ are the representations of p, q, and a.
- A predefined state is represented by itself.
- For any register r, state $\langle r, 0\rangle$ is represented by old_r, and $\langle r, 1\rangle$ by new_r. If the machine is single-register, these representations may be shortened to *old* and *new*.

□

For example, suppose $d \in D$ is the transition $d = \langle\langle r, 1\rangle, s, \langle r, 0\rangle, a\rangle$, and $g(a) = \{d\}$. Note that a is an unordered action (having neither finite nor infinite order). Then a could be represented as

$$\text{add } new_r \xrightarrow{s/a} old_r$$

or even (if you want to be difficult),

$$\text{add } new_r \xrightarrow{s/\text{add } new_r \xrightarrow{s/a} old_r} old_r$$

4 Classes of SMAs

Definition 4.1 (Elementary SMA classes)
Suppose M is an SMA with addition function α and deletion function α'. Let φ be any adjective on modification functions (such as *ρ-normal*, or *finite-order*). Them *M has φ addition* iff α is φ; *M has φ deletion* iff α' is φ; and *M is φ* iff α and α' are both φ. □

For example, M is ρ-normal iff α and α' are both ρ-normal.

Definition 4.2 (classes of deletion)
Suppose M is an SMA with deletion function α'. Then M is *with self-delete* iff α' is self-separable. Further, M is *without deletion* iff α' has order zero. □

When an SMA without deletion is written as a tuple, the deletion function is usually left off, so that the automaton is an 8-tuple $M = \langle\Sigma, Q_0, R, A, S, F, \delta_0, \alpha\rangle$ rather than a 9-tuple $M = \langle\Sigma, Q_0, R, A, S, F, \delta_0, \alpha, \alpha'\rangle$.

Definition 4.3 (without λ-transitions)
Suppose M is an SMA with initial set of transitions δ_0 and addition function α. Let $T \subseteq D$ be the following set of transitions.

$$T = \bigcup_{allow(c,d)} \alpha(c, d)$$

Then M is *without λ-transitions* iff for all transitions $\langle p, s, p', a\rangle \in (\delta_0 \cup T)$, $s \neq \lambda$. □

Definition 4.4 (Single-addition)
Suppose M is an SMA, and α the addition function of M. Then M is *single-addition* iff for all allowable $\langle c, d\rangle$, $|\alpha(c,d)| \leq 1$. □

Definition 4.5 (SMFA)
Suppose M is an SMA with modification action set A, addition function α, and deletion function α'. Then M is *finite* iff A is finite and α and α' are both computable. A finite SMA is called an *SMFA* (self-modifying finite automaton).

(Here, a modification function β is considered computable iff there exists a Turing machine that, given any allowable pair $\langle c, d\rangle$, enumerates the elements of the set $\beta(c,d)$ and halts in finite time.) □

Corollary 4.6 (SMFA step functions)
Suppose M is an SMFA. Then the step function of M is a total function. □

Proof. Suppose M etc. as in the corollary.
Let $\langle c, d\rangle$ be any allowable pair. The definition of SMFA requires that $\alpha(c,d)$ can be enumerated in finite time; therefore $\alpha(c,d)$ is finite, and $states(\alpha(c,d))$ is finite. Since this holds for all allowable $\langle c, d\rangle$, by Lemma 2.4 the step function of M is total. □

Corollary 4.7 (SMFA modification functions)
Suppose M is an SMFA with addition function α and deletion function α'. Then neither α nor α' has infinite order. □

Note that, although the modification functions cannot have infinite order, they could still be unordered; see for example the proof of Theorem 5.3.

Proof. Suppose M etc. as in the corollary.
Let $B = \langle \Sigma, Q_0, R, A\rangle$ be the basis of M; since M is an SMFA, A is finite. Suppose β is any modification function over B. If β is ordered, then since A is finite, every $a \in A$ must have finite order in β; and again because A is finite, there must be a finite upper bound on the order of actions in β, hence by definition, β has finite order. So β must be either unordered or finite-ordered. □

5 Computational power

Theorem 5.1 (SMAs)
Suppose L is a language. Then there exists a single-register ρ-normal SMA without deletion and without λ-transitions that accepts L. □

Note that the SMA in the following proof has, in general, infinite order.

Proof. Suppose Σ is a finite alphabet, and $L \subseteq \Sigma^*$. Consider the SMA without deletion $M = \langle \Sigma, \{q_0, q_1, q_f\}, \{r\}, \Sigma^*, q_0, F, \delta_0, \alpha \rangle$, where

$$F = \begin{cases} \{q_0, q_f\} & \text{if } \lambda \in L \\ \{q_f\} & \text{otherwise} \end{cases}$$

$$\begin{aligned} \delta_0 &= \{q_0 \xrightarrow{\sigma/\sigma} q_1 \mid \sigma \in \Sigma\} \ \cup\ \{q_0 \xrightarrow{\sigma/\lambda} q_f \mid \sigma \in L\} \\ \alpha(c, \langle q, s, q', a \rangle) &= (G(f(a)))(c, \langle q, s, q', a \rangle) \end{aligned}$$

with G the ρ-relative generator for the basis of M, and $f : A \to \mathcal{P}(D)$ defined as

$$f(v) = \begin{cases} \{\mathit{old} \xrightarrow{\sigma/v\sigma} \mathit{new} \mid \sigma \in \Sigma\} \ \cup\ \{\mathit{old} \xrightarrow{\sigma/\lambda} q_f \mid v\sigma \in L\} & \text{if } |v| > 1 \\ \{q_1 \xrightarrow{\sigma/v\sigma} \mathit{new} \mid \sigma \in \Sigma\} \ \cup\ \{q_1 \xrightarrow{\sigma/\lambda} q_f \mid v\sigma \in L\} & \text{if } |v| = 1 \\ \{\} & \text{if } v = \lambda \end{cases}$$

For each $v \in \Sigma^*$, $G(f(v)) \in \mathcal{A}$ is ρ-relative with depth 1; therefore α is ρ-normal, and M is ρ-normal. δ_0 contains no λ-transitions, nor does any $\alpha(c, d)$, therefore M is without λ-transitions. Inspection of M confirms that $L(M) = L$. □

Theorem 5.2 (SMFAs)
Suppose M is an SMFA. Then $L(M)$ is recursively enumerable. □

Proof. Suppose $M = \langle \Sigma, Q_0, R, A, S, F, \delta_0, \alpha, \alpha' \rangle$ is an SMFA with basis $B = \langle \Sigma, Q_0, R, A \rangle$. Given an input string $w \in \Sigma^*$, the following nondeterministic algorithm accepts w iff $w \in L(M)$. Note that since R is finite, any function $\rho : R \to \mathbb{N}$ can be represented in finite space, and its value for given $r \in R$ can be computed in finite time.

1. Let $c = c_w = \langle S, \rho_0, \delta_0, w \rangle$ be the current configuration.

2. If c is an accepting configuration, then accept.

3. If $\delta = \{\}$, then diverge. (c is not part of any accepting computation.)

4. Nondeterministically guess a transition $d = \langle p, s, p', a \rangle \in \delta$. If p is not the current state in c, then diverge. (This guess does not lead to acceptance.)

5. Let $c' = \mathit{step}(c, d)$. Since M is finite, the set of transitions in c must also be finite, and $\mathit{step}(c, d)$ must be defined; also, α and α' are computable; so *step* is computable.

6. Make c' the new value of c, and go back to step 2.

Since M is finite, the set δ of transitions in any reachable configuration is finite. Hence, every step of the algorithm can be accomplished in finite time and space. Hence $L(M)$ is recursively enumerable. □

Theorem 5.3 (ρ-normal SMFAs without deletion)
Suppose L is a recursively enumerable language. Then there exists a ρ-normal SMFA without deletion that accepts L. □

Note that the SMFA in the following proof is unordered.

Proof. Suppose L is a recursively eumerable languages. Then L is accepted by some deterministic Turing machine with two-way-infinite tape $M = \langle Q, Z, T, \delta, q_0 \rangle$, where

- Q is the set of states.
- Z is the tape alphabet, including the blank symbol $\#$, but not symbols L, R, H.
- $T \subseteq Z - \{\#\}$ is the input alphabet.
- $\delta : Q \times Z \rightarrow (Q \times Z \times \{L, R\}) \cup \{H\}$ is the transition function.
- $q_0 \in Q$ is the start state.

Here, L, R, H mean "move left" and "move right", and "halt".

Configurations are indexed by nonnegative integers $j \in \mathbb{N}$. Tape cells are indexed by integers $k \in \mathbb{Z}$. Let $z_{j,k} \in Z$ be the symbol at cell k, $p_j \in \mathbb{Z}$ the head position, and q_j the machine state, in configuration j. In the initial configuration, with input string $w = w_1 \cdots w_n$, $w_k \in T$,

$$\begin{aligned} p_0 &= 0 \\ z_{0,k} &= \begin{cases} w_k & \text{if } 1 \le k \le n \\ \# & \text{otherwise} \end{cases} \end{aligned}$$

An SMFA will now be constructed that simulates M, hence accepts L.

Each configuration j is represented by a path of λ-transitions with actions $a_{j,k}$, for $-j \le k \le n + 1 + j$, where

$$a_{j,k} = \begin{cases} \langle q_j, z_{j,k} \rangle & \text{if } k = p_j \\ z_{j,k} & \text{otherwise} \end{cases}$$

Traversing this path constructs a path representing the next configuration (except for the first and last transitions of the new path, which are constructed by predefined transitions using the special actions *begin* and *end*). The following set of $2|Q|+1$ registers is used in the construction:

$$R \;=\; \{r_0\} \cup \{r_{q,L} \mid q \in Q\} \cup \{r_{q,R} \mid q \in Q\}$$

When constructing configuration j from configuration $j-1$, the transitions for the old and new head positions (p_{j-1} and p_j) are connected through register $r_{q_j,d}$, where d is the direction moved by the head between configurations $j-1$ and j. All other consecutive pairs of transitions are connected through r_0.

In order to guarantee that all of the registers will be updated by every action of the machine, every action adds transitions

$$new_r \xrightarrow{\lambda} new_r \quad \forall r \in R$$

Additional transitions are added by various actions, as follows. Here, q_f is the final state, and q_L, q_R are other predefined states.

$$\begin{array}{llcll}
z: & old_0 & \xrightarrow{\lambda/z} & new_0 & \\
 & old_0 & \xrightarrow{\lambda/\langle q,z\rangle} & new_{q,L} & \forall q \in Q \\
 & old_{q,R} & \xrightarrow{\lambda/\langle q,z\rangle} & new_0 & \forall q \in Q \\
\langle q,z\rangle: & old_0 & \xrightarrow{\lambda/z'} & new_{q',R} & \text{if } \delta(q,z) = \langle q',z',R\rangle \\
 & old_{q',L} & \xrightarrow{\lambda/z'} & new_0 & \text{if } \delta(q,z) = \langle q',z',L\rangle \\
 & old_0 & \xrightarrow{\lambda} & q_f & \text{if } \delta(q,z) = H \\
begin: & q_L & \xrightarrow{\lambda/\#} & new_0 & \\
 & q_L & \xrightarrow{\lambda/\langle q,\#\rangle} & new_{q,L} & \forall q \in Q \\
end: & old_0 & \xrightarrow{\lambda/\#} & q_R & \\
begin': & q_L & \xrightarrow{\lambda/\langle q_0,\#\rangle} & new_0 &
\end{array}$$

The entire SMFA is shown in Figure 1. q_s is the start state, and q_f the final state. During computation, the entire input string must be read while in state q_1; otherwise, by Definition 2.3 the string will not be accepted. Traversing from q_s to q_2 creates the initial configuration path from q_L to q_R. Thereafter, traversing any loop from q_2 to q_2 creates another configuration path from q_L to q_R. There is no requirement that this loop always use the most recently added configuration path, but repeating an earlier configuration path only creates a redundant copy of some other existing path. The final state is reachable iff some accepting configuration path can be created, iff that configuration is reachable from the given initial configuration. $\square$

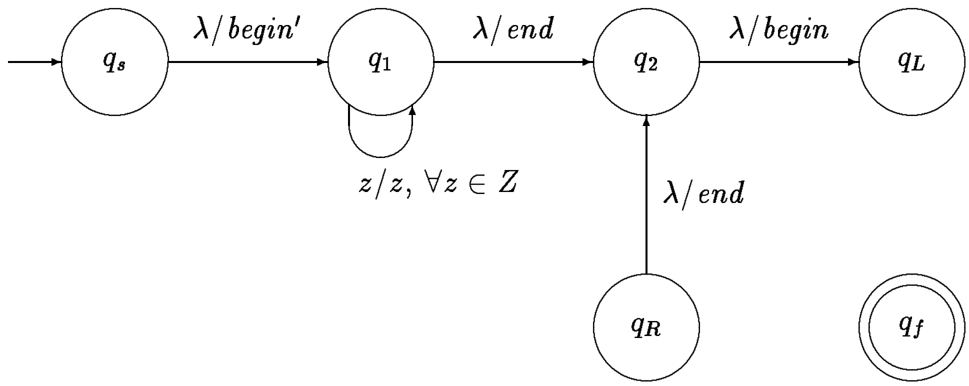

Figure 1: SMFA to simulate a DTM

Corollary 5.4 (ρ-normal SMFAs without deletion)
Any language L is recursively enumerable iff it is accepted by some ρ-normal SMFA without deletion. □

Proof. Immediate from Theorems 5.2 and 5.3. □

6 Conclusion

Formal definitions for Self-Modifying Automata (SMA's), Self-Modifying Finite Automata (SMFA's), and their properties have been presented here, supplementing and augmenting the definitions for SMFA's previously presented in [RS93, RS94, RS95]. The open SMFA problems previously presented may now be approached from additional angles, with additional techniques.

Basic results on the computational power of SMA's and SMFA's have also been presented. This includes the results that for an arbitrarily difficult language there exists a restricted form of SMA that accepts that language; every language accepted by an SMFA is recursively enumerable; and every recursively enumerable language is accepted by a restricted form of SMFA. The last two results together provide a new characterization of the recursively enumerable languages in terms of a restricted form of SMFA's.

References

[RS93] R. Rubinstein and J. Shutt. Self-modifying finite automata. Technical Report WPI-CS-TR-93-11, Worcester Polytechnic Institute, Worcester, MA, December 1993.

[RS94] R. Rubinstein and J. Shutt. Self-modifying finite automata. In B. Pehrson and I. Simon, editors, *Technology and Foundations: Information Processing '94 Vol. I: Proc. 13th IFIP World Computer Congress*, pages 493–498, Amsterdam, 1994. North-Holland.

[RS95] R. Rubinstein and J. Shutt. Self-modifying finite automata: An introduction. *Information Processing Letters*, 1995. To appear.