Theoretical Computer Science 1 (1976) 297–316. © North-Holland Publishing Company

THE INCLUSION PROBLEM FOR SIMPLE LANGUAGES*

EMILY P. FRIEDMAN

Department of System Science, University of California, Los Angeles, Calif. 90024, USA

Communicated by M. Nivat
Received October 1974
Revised February 1975

Abstract. A deterministic pushdown acceptor is called a simple machine when it is restricted to have only one state, operate in real-time, and accept by empty store. While the existence of an effective procedure for deciding equivalence of languages accepted by these simple machines is well-known, it is shown that this family is powerful enough to have an undecidable inclusion problem. It follows that the inclusion problems for the $LL(k)$ languages and the free monadic recursion schemes that do not use an identity function are also undecidable.

1. Introduction

One of the most important questions in language theory is determining whether the languages accepted by two different machines in some given class are the same. This question is called the equivalence problem, and it has been investigated for a wide variety of machine classifications [3,8,9,11]. Recently, however, its relation to another question has drawn a great deal of interest. This question, known as the inclusion problem, is that of deciding whether one language is a subset of another. The most obvious connection between the equivalence and inclusion problems can be explained as follows: The decidability of the inclusion problem for some family of languages implies the decidability of the equivalence problem for the same family. This is determined by the fact that $L_1 = L_2$ iff $L_1 \subseteq L_2$ and $L_2 \subseteq L_1$, for any two languages L_1 and L_2. Thus, any algorithm that decides inclusion for some family of languages can also be used to decide equivalence. The question naturally arises as to whether the converse holds. Does the decidability of the equivalence problem for some family of languages implies decidability of the inclusion problem for that same family?

Until recently, all "natural" families of languages studied had been found to have their equivalence and inclusion problems both be decidable or both be unde-

* The results reported are a portion of the author's Ph. D. dissertation written at Harvard University under the supervision of Professor Ronald V. Book. The research was supported in part by the National Science Foundation under Grant NSF GJ-30409 and Grant NSF GJ-803.

cidable. For example, the regular sets [8] and the bounded languages [6] both have decidable equivalence and inclusion problems, whereas the context-free languages [8] and the languages accepted by Turing machines [8] both have undecidable equivalence and inclusion problems. In 1971, Bird exhibited the first case that fit into neither of these previous patterns [3]. He found that the languages accepted by two-tape Rabin and Scott machines [10] possess a decidable equivalence problem and an undecidable inclusion problem. Valiant's thesis [12] explored this question further, finding three other families exhibiting this unusual feature of having a decidable equivalence problem and an undecidable inclusion problem: the languages accepted by finite-turn deterministic pushdown automata, one-counter deterministic pushdown automata, and nonsingular automata.

In this paper we investigate another subclass of deterministic pushdown automata — simple machines. These devices are basically deterministic pushdown automata that have only one state and operate in real-time. In Section 2, we outline the fundamental properties of these machines.

Simple machines were first investigated by Korenjak and Hopcroft [9] in an attempt to find a subfamily of the deterministic languages that characterizes a fairly large class of programming languages that can be rapidly parsed. Although these languages cannot describe all of the structures existing in a language such as ALGOL, they can describe many other non-regular features. An interesting property of the languages accepted by simple machines is that they possess a decidable equivalence problem [9]. In their paper, Korenjak and Hopcroft posed the inclusion problem as the major open question relating to simple languages remaining to be solved [9]. A significant new result, Theorem 3.1, shows that these "simple languages" indeed have an undecidable inclusion problem. Because the simple languages are a proper subfamily of the *LL* languages (in fact, they are the simple *LL*(1) languages described in [1]), the undecidability of the inclusion problem for *LL* languages is an immediate result. Therefore, we add two new families of languages to the growing list of those with decidable equivalence but undecidable inclusion problems. The nature of the feature or features contributing to this strange classification for families of languages remains to be discovered.

2. Fundamental properties

A *simple machine* is a very restricted form of a pushdown automaton. A pushdown automaton is called simple if it has only *one state* and operates in *real-time* (no e-moves). Because these machines possess only one state, no information can be stored in its finite-state control. Therefore, a simple machine can alternately be thought of as a stateless device [12]. For this reason, there is no necessity for including a state set in the definition of simple machines. We use the formal definition that follows.

Definition 2.1. (a) A *simple machine* is a 4-tuple $M = (\Sigma, \Gamma, \delta, Z_0)$, where

Σ is a finite *input* alphabet,

Γ is a finite *pushdown* alphabet,

$\delta : \Sigma \times \Gamma \to \Gamma^*$ is the partial *transition function*,

$Z_0 \in \Gamma$ is the *initial pushdown symbol*.

(b) A *configuration* of M is a pair (w, α), where

$w \in \Sigma^*$ is the portion of the input tape remaining to be read (i.e., not yet passed under the read-head),

$\alpha \in \Gamma^*$ is the current contents of the pushdown store, where the top of the store is the rightmost symbol of α.

We define the operator $\vdash_M$ (or $\vdash$ where the machine M is clearly implied by the context) on configurations of M as follows:

For all $a \in \Sigma$, $w \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \beta \in \Gamma^*$,

$$(aw, \alpha Z) \vdash_M (w, \alpha\beta) \quad \text{iff } \delta(a, Z) = \beta.$$

Each occurrence of $(aw, \alpha Z) \vdash_M (w, \alpha\beta)$ is called a move of the simple machine M.

(c) A computation of M is a sequency of configurations $c_1, \ldots, c_n$, $n \geqslant 1$, where $c_i \vdash_M c_{i+1}$, for all $1 \leqslant i \leqslant n-1$. we let $\vdash_M^*$ denote the transitive reflexive closure of $\vdash_M$, and $\vdash_M^+$ the transitive closure of $\vdash_M$.

For $t \geqslant 0$, we define $\vdash_M^t$ as follows:

(i) $c \vdash_M^0 c$ for all configurations c;

(ii) $c \vdash_M^t d$ iff there is a computation $c_0, \ldots, c_t$, where $c = c_0$, $d = c_t$.

Thus, $c \vdash_M^* d$ iff there is some $t \geqslant 0$ such that $c \vdash_M^t d$, and $c \vdash_M^+ d$ iff there is some $t \geqslant 1$ such that $c \vdash_M^t d$.

After the entire tape has been processed, the resulting configuration of the simple machine determines whether the input tape is accepted or rejected. For this paper, we are concerned with acceptance of an input tape by empty pushdown store. Accordingly, we define the language accepted by simple machine M to be

$$N(M) = \{w \in \Sigma^* \mid (w, Z_0) \vdash_M^* (e, e)\}.$$

A language L is said to be *simple* if $L = N(M)$ for some simple machine M. We can now define S as the family of simple languages.

A simple machine is defined so that once the pushdown store is empty, no further input can be processed. Thus, if $L \in S$, then L is prefix-free[1]. Hence, S corresponds precisely to the family of languages defined by Harrison and Havel as the real-time strict deterministic languages of degree 1 (see[7]).

Recall that simple machines are defined so that no e-moves are allowed; the partial transition function is only defined on an input symbol and a pushdown symbol.

[1] A language L is prefix-free if no proper prefix of a word in L is also a member of L. That is, if $x \in L$, and $xy \in L$, then $y = e$.

Valiant shows that we can relax the real-time restriction on our machines and not enlarge the family of languages accepted [12]. That is, we could redefine the transition function for simple machines in order to allow for e-moves. By this, we mean that if $M = (\Sigma, \Gamma, \delta, Z_0)$ is a simple machine, then the partial transition function δ is such that $\delta : (\Sigma \cup \{e\}) \times \Gamma \to \Gamma^*$. The requirement that simple machines remain deterministic includes the restriction that for each $Z \in \Gamma$, either (i) $\delta(e, Z) = \emptyset$, or (ii) for all $a \in \Sigma$, $\delta(a, Z) = \emptyset$.

The family of languages accepted by these "less restricted" simple machines (that allow e-moves) is precisely S. For convenience, then, we retain our earlier definition for simple machines without e-moves. This does not really affect any of the results that we prove.

3. The inclusion problem

Korenjak and Hopcroft [9] first investigated the properties of the languages of S in an attempt to find a characterization for a fairly large class of programming languages that can be parsed rapidly. Their main theorem shows that the equivalence problem for S is decidable. The major open question in their paper is whether or not the inclusion problem for languages accepted by simple machines is decidable. Recall that this problem was posed before the discovery of any natural family of languages having a decidable equivalence problem and an undecidable inclusion problem [3]. We intend to show that the question of inclusion is indeed undecidable. The inspiration for the proof comes from the work of Valiant [12]. He shows that the inclusion problem for a family of languages accepted by a subclass of real-time pushdown automata (i. e., nonsingular automata) is undecidable. He was not able to extend his proof to include the one-state restriction, or even for LL languages.

We shall now prove the major theorem of this paper.

Theorem 3.1. *The inclusion problem for S is undecidable.*

Before we prove Theorem 3.1 we first give some definitions.

Definition 3.2. We use the following as our definition for a Turing machine.

A *Turing machine* is a 6-tuple $M = (K, \Sigma, \Gamma, \delta, q_0, F)$, where

K = the set of *states*,

Γ = the set of *tape* symbols. Let $B \in \Gamma$ denote the *blank* tape symbol,

Σ = the set of *input* symbols. Also, $B \notin \Sigma$,

$q_0 \in K$ = the *starting state*,

$F \subseteq K$ = the set of *final states*,

$\delta : K \times \Gamma \to K \times (\Gamma - \{B\}) \subseteq \{L, R\}$ is the partial *transition function*.

A *configuration* of the Turing machine M is denoted by the string $\alpha q \beta$, where

$q \in K$ is the *current state* of M,

$\alpha\beta \in (\Gamma - \{B\})^*$ is the non-blank portion of the tape.

The tape head is situated so that the next tape symbol that will be read is the initial symbol of β, if $\beta \neq e$. For the case where $\beta = e$, the next symbol to be read is a blank, B.

We define the relation $\vdash_M$ (or $\vdash$ when M is understood from the context) on configurations of M as follows:

For all $a, b, c \in \Gamma - \{B\}$, $\alpha, \beta, \gamma, \mu \in (\Gamma - \{B\})^*$,

(1) if $\delta(q, a) = (p, b, R)$ and $\beta = a\gamma$, then $\alpha q \beta = \alpha q a \gamma \vdash_M \alpha b p \gamma$;

(2) if $\delta(q, a) = (p, b, L)$ and $\beta = a\gamma$, $\alpha = \mu c$, then $\alpha q \beta = \mu c q a \gamma \vdash_M \mu p c b \gamma$;

(3) if $\delta(q, \beta) = (p, a, R)$ and $\beta = e$, then $\alpha q \beta = \alpha q \vdash_M \alpha a p$;

(4) if $\delta(q, B) = (p, a, L)$ and $\beta = e$, $\alpha = \mu c$, then $\alpha q \beta = \mu c q \vdash_M \mu p c a$.

Each occurrence of $\alpha q \beta \vdash_M \alpha' q' \beta'$ is called a *move* of the Turing machine. Moves (1) and (3) advance the read head one tape square to the right, whereas moves (2) and (4) move the read head one square to the left. We let $\vdash_M^*$ denote the transitive reflexive closure of $\vdash_M$.

An input tape is accepted if the Turing machine ever gets into some configuration where the current state is a member of F. Thus, the language accepted by Turing machine M is defined to be

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash_M^* \alpha q \beta, \text{ for some } \alpha, \beta \in (\Gamma - \{B\})^*, \ q \in F\}.$$

Definition 3.3. A Turing machine $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ is said to *halt* on input tape $w \in \Sigma^*$ if there exists some $\alpha, \beta \in \Gamma^*$, $q \in K$ such that $q_0 w \vdash_M^* \alpha q \beta$, and no next move is possible from configuration $\alpha q \beta$.

The *halting problem for Turing machines* is as follows:

Given any Turing machine $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ and input tape $w \in \Sigma^*$, does Turing machine M halt on input w?

It is well-known that the halting problem for Turing machines is undecidable [3]. That is, there is no algorithm that can decide whether a Turing machine halts for some given input tape.

Definition 3.4. Let Σ be a finite set of symbols containing at least two elements, and let $\mathcal{J}$ be a finite non-empty sequence of ordered pairs of strings in Σ^+. For example,

$$\mathcal{J} = (x_1, y_1), \ldots, (x_n, y_n),$$

where for $i = 1, \ldots, n$, $x_i, y_i \in \Sigma^+$. $\mathcal{J}$ is called an *instance* of the *Modified Correspondence Problem.*

The sequence of indices $i_1, \ldots, i_t$ with $t \geq 1$ is a *solution for* $\mathcal{J}$ if $x_1 x_{i_1} \ldots x_{i_t} = y_1 y_{i_1} \ldots y_{i_t}$.

The *Modified Correspondence Problem* is defined as follows:

Given any instance of the Modified Correspondence Problem, δ, does there exist a solution for δ?

The Modified Correspondence Problem is undecidable [8].

The rest of this section is devoted to the proof of Theorem 3.1.

Given a deterministic Turing machine M and an input tape w, we construct two simple machines, M_1 and M_2, such that $N(M_1) \nsubseteq N(M_2)$ iff M halts and accepts w. Thus, we show that if the inclusion problem for S were solvable, we then could solve complement of the halting problem for Turing machines. In fact, showing this means that the inclusion problem for S is *not* semi-decidable. (Recall that the halting problem is semi-decidable, but the complement of the halting problem is not semi-decidable.)

Let $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ be any Turing machine, and $w \in \Sigma^*$ be any input tape to M. Without loss of generality, assume that $F = \{q_f\}$, and for each $a \in \Gamma$, $\delta(q_f, a)$ is undefined. Also, assume that for all $q \in K - \{q_f\}$, $a \in \Gamma$, that $\delta(q, a)$ is defined. Hence, Turing machine M halts on input tape w iff M accepts w. We now show how to construct simple machines M_1 and M_2 from the definition of Turing machine M and input tape w. The reduction that follows is essentially similar to the proof of Hopcroft and Ullman of the undecidability of the Modified Correspondence Problem [8].

First, we construct two finite lists, X and Y, of non-empty strings. Only the first two pairs are numbered, as the numbering of the other parts is irrelevant. Recall that B is the blank, and $\#$ is a new symbol not connected to M.

	LIST X	LIST Y
Pair 1:	$\#$	$\#q_0w\#$
Pair 2:	$q_f\#\#$	$\#$
	$\#$	$\#$

For each $d_1, d_2, d_3 \in \Gamma - \{\mathrm{B}\}$, $p \in K$, $q \in K - \{q_f\}$,

d_1	d_1	
qd_1	d_2p	if $\delta(q, d_1) = (p, d_2, \mathrm{R})$,
d_3qd_1	pd_3d_2	if $\delta(q, d_1) = (p, d_2, \mathrm{L})$,
$q\#$	$d_2p\#$	if $\delta(q, \mathrm{B}) = (p, d_2, \mathrm{R})$,
$d_3q\#$	$pd_3d_2\#$	if $\delta(q, \mathrm{B}) = (p, d_2, \mathrm{L})$,
$d_1q_fd_2$	q_f	
$d_1q_f\#$	$q_f\#$	
$\#q_fd_2$	$\#q_f$	

We see that X and Y each consists of n strings, for some $n \geq 2$. Each string is composed of symbols from $K \cup \{\#\} \cup (\Gamma - \{\mathrm{B}\})$. We denote X and Y by lists $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$, respectively.

Hopcroft and Ullman [8] show that for all $i \in \{0, ..., k\}$, $q_i \in K$, $\alpha_i, \beta_i \in (\Gamma - \{B\})^*$, if

$$q_0 w \vdash_M \alpha_1 q_1 \beta_1 \vdash_M \cdots \vdash_M \alpha_k q_k \beta_k,$$

then there exist integers $r \geqslant 0$, $3 \leqslant i_1, ..., i_r \leqslant n$, such that

$$x_1 x_{i_1} \cdots x_{i_r} = \# q_0 w \# \alpha_1 q_1 \beta_1 \# \cdots \# \alpha_{k-1} q_{k-1} \beta_{k-1} \#,$$

$$y_1 y_{i_1} \cdots y_{i_r} = \# q_0 w \# \alpha_1 q_1 \beta_1 \# \cdots \# \alpha_{k-1} q_{k-1} \beta_{k-1} \# \alpha_k q_k \beta_k \#.$$

If $q_k = q_f$, then by choosing pairs of strings (x_j, y_j), where x_j contains q_f and pairs from $\{(\#,\#)\} \cup \{(d, d) \mid d \in \Gamma - \{B\}\}$, we obtain

$$x_1 x_{i_1} \cdots x_{i_r} x_{i_{r+1}} \cdots x_{i_t} x_2 = y_1 y_{i_1} \cdots y_{i_r} y_{i_{r+1}} \cdots y_{i_t} y_2.$$

In addition, if there exist some $s \geqslant 0$, $3 \leqslant j_1, ..., j_s \leqslant n$ such that

$$x_1 x_{j_1} \cdots x_{j_s} x_2 = y_1 y_{j_1} \cdots y_{j_s} y_2,$$

then $r \leqslant s$ and $j_1 = i_1$, $j_2 = i_2$, ..., $j_r = i_r$.

Therefore, we make the following claim that is proved in [8].

Claim 1. *M halts and accepts input tape w iff there exist $t \geqslant 0$, $3 \leqslant i_1, ..., i_t \leqslant n$ such that*

$$x_1 x_{i_1} \cdots x_{i_t} x_2 = y_1 y_{i_1} \cdots y_{i_t} y_2.$$

Notice that for $i \geqslant 3$, each x_i and y_i contain the same number of occurrences of letters denoting states of M — either 0 or 1; that x_1 contains no state; that y_1 contains 1 state. Thus, in $\bar{x} = x_1 x_{i_1} \cdots x_{i_t}$ and $\bar{y} = y_1 y_{i_1} \cdots y_{i_t}$, there is one more occurrence of a letter denoting a state of M in $\bar{y}$ than in $\bar{x}$. Also, note that x_i ends in $\#$ iff y_i ends in $\#$.

At this point, we use an example to illustrate the construction of simple machines M_1 and M_2 from a given Turing machine and input tape.

Let M be a Turing machine that is given a string of 1's as its input, and appends another 1 to the right of this input tape and then halts (and accepts), and let $w = e$ be the input tape to M. Thus,

$$M = (\{q_0, q_f\}, \{1\}, \{1, B\}, \delta, q_0, \{q_f\})$$

with

$$\delta(q_0, 1) = (q_0, 1, R), \qquad \delta(q_0, B) = (q_f, 1, R).$$

Lists X and Y each consists of 9 non-empty strings constructed from the alphabet $\{\#, 1, q_0, q_f\}$.

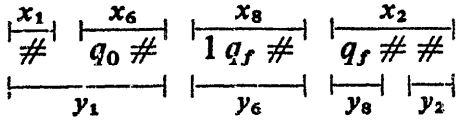
	LIST X	LIST Y
Pair 1:	$\#$	$\#q_0\#$
Pair 2:	$q_f\#\#$	$\#$
Pair 3:	1	1

Pair 4:	$\#$	$\#$
Pair 5:	$q_0 1$	$1 q_0$
Pair 6:	$q_0 \#$	$1 q_f \#$
Pair 7:	$1 q_f 1$	q_f
Pair 8:	$1 q_f \#$	$q_f \#$
Pair 9:	$\# q_f 1$	$\# q_f$

The computation of M on the blank tape is

$$q_0 \vdash_M 1 q_f.$$

Hence,



For each integer i, $1 \leqslant i \leqslant n$, let f_i be a new symbol not in $\Delta = (\Gamma - \{B\}) \cup K \cup \{\#\}$. f_i should be considered an encoding for integer i. In addition, let $\not{\text{c}}$ and \$ be two new symbols not in $\Delta \cup \{f_1, \ldots, f_n\}$. Define the homomorphism $h: \Delta^* \to (\Delta \cup \{\not{\text{c}}\})^*$ determined by defining $h(d) = d\not{\text{c}}$ for all $d \in \Delta$. This homomorphism will only be used to describe the languages accepted by M_1 and M_2. The definitions of the two machines are now given.

We first construct the simple machine M_1, and make the claim that

$$N(M_1) = \{f_2 f_{i_t} \cdots f_{i_1} f_1 h(y_1 y_{i_1} \cdots y_{i_t} y_2) \$ \mid t \geqslant 0,\ 3 \leqslant i_1, \ldots, i_t \leqslant n\}.$$

(Note: the case where $t = 0$ in the definition of the above set, defines the string $f_2 f_1 h(y_1 y_2)$ \$.)

Informally, the mechanism of M_1 is such that it reads along the encodings of integers (i.e., reads f's) and pushes the associated y's onto the pushdown store. For example, when M_1 reads f_4, it pushes y_4 onto the pushdown store. Then, as M_1 reads symbols from $\Delta \cup \{\not{\text{c}}\}$, it pops when the symbol at the top of the pushdown store matches the input symbol, checking that a $\not{\text{c}}$ appears as every other symbol. M_1 accepts the tape if \$ is read when the pushdown store has only Z_1 as its contents.

A formal definition of M_1 follows:

Let

$$M_1 = (\Delta \cup \{\not{\text{c}}\} \cup \{\$\} \cup \{f_1, \ldots, f_n\}, \delta_1, \Gamma_1, Z_0)$$

where

$$\Gamma_1 = \{Z_0, Z_1, C\} \cup \{[z] \mid z \in \Delta^+, |z| \leqslant \max\{|y_i|\}\} \cup \{[z\not{\text{c}}] \mid z \in \Delta^*, |z| < \{|y_i|\}\}, \qquad \Gamma_1 \cap \Delta = \emptyset,$$

(note that $[z]$ (and also $[z\not{\text{c}}]$) is a *single* pushdown symbol).

I. Insure that the initial portion of an accepted input tape is a string of the form $f_2 \alpha f_1$, where $\alpha \in \{f_3, ..., f_1\}^*$.

(a) $\delta_1(f_2, Z_0) = Z_1[y_2^R]\, C$,

(b) $\delta_1(f_i, C) = [y_i^R]\, C$, for each $i = 3, ..., n$,

(c) $\delta(f_1, C) = [y_1^R]$.

II. The next portion of an accepted input tape must be a string $h(y_1 \beta y_2)$, where $\beta \in \{y_3, ..., y_n\}^*$. Also, if the first portion of the input tape is $f_2 \alpha f_1$ (see part I above), then we insure that

(i) $\alpha = e$ implies that $\beta = e$,

(ii) $\alpha = f_{i_j} ... f_{i_1}$, for $j \geqslant 1, 3 \leqslant i_1, ..., i_j \leqslant n$ implies that $\beta = y_{i_1} ... y_{i_j}$.

For each $d \in \Delta$, $z \in \Delta^*$, $|z| < \max\{|y_i|\}$, we have

(d) $\delta_1(d, [zd]) = [z\not\epsilon]$, check for matching symbol,

(e) $\delta_1(\not\epsilon, [z\not\epsilon]) = [z]$, if $z \neq e$, } check that every other symbol

(f) $\delta_1(\not\epsilon, [\not\epsilon]) = e$, } is a $\not\epsilon$.

III. The final symbol in an accepted input tape is a \$.

(g) $\delta_1(\$, Z_1) = e$,

Note that δ_1 is only partially defined.

Facts about machine M_1. These facts are obvious from the construction above, so no proofs are provided.

(1) If $(a, Z_0) \vdash (e, \alpha)$ is a valid computation of M_1, then by definition (a) of transition function δ_1, it is clear that $a = f_2$ and $\alpha = Z_1[y_2^R]C$.

(2) If $(a, C) \vdash (e, \alpha)$ is a computation of M_1, with $\alpha \notin \Gamma_1^* C$, then from part (c) of the definition of δ_1, we have $a = f_1$, $\alpha = [y_1^R]$.

(3) If $(v, C) \vdash (e, \alpha C)$ is a computation of M_1 for $t \geqslant 1$, then by part (b) of δ_1, we have $v = f_{i_t} ... f_{i_1}$, $\alpha = [y_{i_t}^R] ... [y_{i_1}^R]$, where $3 \leqslant i_1, ..., i_t \leqslant n$.

(4) If $(v, z^R) \vdash^* (e, e)$ is a valid computation of M_1, then parts (d), (e), and (f) of the definition of δ_1 insure that $v = h(z)$.

(5) If $(v, Z_1) \vdash^* (e, e)$ is a computation of M_1, then part (g) of δ_1 requires that $v = \$$.

From the above facts, we make the following claim (without proof) about the structure of the language accepted by M_1.

Claim 2.

$$N(M_1) = \{f_2 f_{i_t} ... f_{i_1} f_1 h(y_1 y_{i_1} ... y_{i_t} y_2)\$ \mid t \geqslant 0,\ 3 \leqslant i_1, ..., i_t \leqslant n\}.$$

We now construct the simple machine M_2, and make the claim that

$$N(M_2) = \{f_2 f_{i_t} \dots f_{i_1} f_1 h(y_1 y_{i_1} \dots y_{i_t} y_2)\$ \mid t \geqslant 0,\ 3 \leqslant i_1, \dots, i_t \leqslant n,\ x_1 x_{i_1} \dots x_{i_t} x_2 \neq y_1 y_{i_1} \dots y_{i_t} y_2\}$$
$$\cup \text{ \{some strings that irrelevant to our arguments that follow\}}.$$

Informally, the machine M_2 starts off in a manner similar to M_1, by reading the f's and pushing the associated x's onto the pushdown store. Next, as M_2 reads symbols from $\Delta \cup \{¢\}$, it pops whenever the input symbol matches the symbol at the top of the pushdown store, again, checking that a ¢ appears as every other symbol. Unlike M_1, M_2 rejects the input tape if the \$ is read when the pushdown store consists only of Z_1. M_2 empties the pushdown store and accepts the input tape whenever (1) the top of the store does not match the input symbol read, (2) when \$ is encountered and the pushdown store does not consist only of Z_1, or (3) when the store contains only Z_1 and the input tape is not the string \$. The proof that M_2 is actually capable of accepting an input tape under all three of these conditions constitutes the major portion of the proof that follows the construction.

One portion of the proof shows that simple machine M_2 accepts all strings of the form

$$f_2 f_{i_t} \cdots f_{i_1} f_1 h(y_1 y_{i_1} \cdots y_{i_t} y_2)\$,$$

where $t \geqslant 0$, $3 \leqslant i_1, \dots, i_t \leqslant n$, $y_1 y_{i_1} \cdots y_{i_t} y_2 \neq x_1 x_{i_1} \cdots x_{i_t} x_2$ and

$$|y_1 y_{i_1} \cdots y_{i_t} y_2| \geqslant |x_1 x_{i_1} \cdots x_{i_t} x_2|.$$

As soon as a mismatched symbol is encountered, we are insured that there are at least as many input symbols remaining to be read as there are symbols on the pushdown store. At this point, the ¢ markers play a key role in encoding the fact that a mismatch has occurred and that the entire pushdown store must be popped. In fact, it is the use of these ¢ markers that gives rise to the "irrelevant strings" mentioned in Claim 3 below.

The second (and most difficult) part of the proof that follows is to show that M_2 accepts *all* strings of the form

$$f_2 f_{i_t} \cdots f_{i_1} f_1 h(y_1 y_{i_1} \cdots y_{i_t} y_2)\$$$

where $t \geqslant 0$, $3 \leqslant i_1, \dots, i_t \leqslant n$, $y_1 y_{i_1} \cdots y_{i_t} y_2 \neq x_1 x_{i_1} \cdots x_{i_t} x_2$. This includes the case where $|y_1 y_{i_1} \cdots y_{i_t} y_2| \leqslant |x_1 x_{i_1} \cdots x_{i_t} x_2|$.

We prove in Claim 3 that as soon as a mismatch occurs, there are still enough input symbols remaining to be read in order to pop the pushdown store and thus accept the tape. Lists X and Y have been deviously constructed to enable the proof to be carried through.

The construction of simple machine M_2 appears below. Let

$$M_2 = (\Delta \cup \{¢\} \cup \{\$\} \cup \{f_1, \dots, f_n\}, \Gamma_2, \delta_2, Z_0),$$

where

$$\Gamma_2 = \{Z_0, Z_1, A, C, D, J\} \cup \{[z] \mid z \in \Delta^+, |z| \leqslant \max\{|x_i|\}\} \cup \{[z¢] \mid z \in \Delta^*, |z| < \max\{|x_i|\}\}$$

(as in M_1, $[z]$ (and also $[z¢]$) is a *single* pushdown symbol) and the transition function δ_2 is

I. Insure that the initial portion of an accepted input tape is a string of the form $f_2\alpha f_1$, where $\alpha \in \{f_3, ..., f_n\}^*$.

(a) $\delta_2(f_2, Z_0) = Z_1[x_2^R]C$,
(b) $\delta_2(f_i, C) = [x_i^R]C$, for $i = 3, ..., n$,
(c) $\delta_2(f_1, C) = [x_1^R]$.

II. As in part II of the definition of δ_1 (machine M_1), we match input symbols against the pushdown store symbols, checking that a ¢ is every other symbol in the input tape.

For $d \in \Delta$, $z \in \Delta^*$, $|z| < \max\{|x_i|\}$,

(d) $\delta_2(d, [zd]) = [z¢]$, check matching symbols,
(e) $\delta_2(¢, [¢]) = e$,
(f) $\delta_2(¢, [z¢]) = [z]$, if $z \neq e$, } check that every other symbol is a ¢.

III. Reject the input tape if the final symbol is \$ and the pushdown store contents is Z_1.

(g) $\delta_2(\$, Z_1) = J$.

IV. If there is a mismatch between the input symbol and the top of the pushdown store, use the pushdown store symbol D as a means to encode the fact so that we can accept the input tape.

For $d_1, d_2 \in \Delta$, $d_1 \neq d_2$, and $z, \bar{z} \in \Delta^*$, $|\bar{z}| \leqslant \max\{|x_i|\}$, $|z| < \max\{|x_i|\}$,

(h) $\delta_2(d_1, [zd_2]) = e$, check for mismatched symbols and pop the pushdown,
(i) $\delta_2(¢, [\bar{z}]) = D, \bar{z} \neq e$, use ¢ to then encode D on pushdown,
(j) $\delta_2(d_1, D) = e$.

V. If the bottom of the pushdown store is encountered (Z_1) when a symbol other than \$ is read, use pushdown symbol A to encode that the input tape will be eventually accepted when the \$ is finally read.

For $d_3 \in \Delta \cup \{¢\}$,

(k) $\delta_2(d_3, Z_1) = A$, use A to encode the fact that we will eventually accept the tape,
(l) $\delta_2(d_3, A) = A$, process all symbols until \$,
(m) $\delta_2(\$, A) = e$, accept the input tape.

Facts about machine M_2. No proofs are given, as the facts follow directly from the construction above.

(1) If $(a, Z_0) \vdash (e\ \alpha)$ is a computation of M_2, then by part (a) of the definition of δ_2, $a = f_2$ and $\alpha = Z_1[x_2^R]C$.

(2) If $(a, C) \vdash (e, \alpha)$ is a computation of M_2, with $\alpha \notin \Gamma_2^* C$, the part (c) of δ_2 insures that $a = f_1$ and $\alpha = [x_1^R]$.

(3) If $(v, C) \overset{t}{\vdash} (e, \alpha C)$ is a computation of M_2 for $t \geqslant 1$, then by definition (b), we have $v = f_{i_t} \dots f_{i_1}$, $\alpha [x_{i_t}^R] \dots [x_{i_1}^R]$, where $3 \leqslant i_1, \dots, i_t \leqslant n$.

(4) For any $u, v \in \Delta^*$, $|u| \leqslant \max\{|x_i|\}$, $|u| \leqslant |v|$, then either

(i) u is a prefix of v, $v = u\bar{v}$ for some $\bar{v} \in \Delta^*$, and $(h(v), [u^R]) \overset{*}{\vdash} (h(\bar{v}), e)$ by parts (d), (e), and (f), or

(ii) u is not a prefix of v. Then $u = \bar{u} d_1 v_1$ and $v = \bar{u} d_2 v_2$ for $\bar{u}, v_1, v_2 \in \Delta^*$, $d_1, d_2 \in \Delta$, $d_1 \neq d_2$, *and* $(h(v), [u^R]) \overset{*}{\vdash} (\not\in h(v_2), e)$ by parts (d), (f), and (h) of the definition of δ_2.

(5) For any $v \in \Delta^*$, if $|v| = j \geqslant 1$, then by parts (i) and (j) for the definition of δ_2, we have $(h(v), [x_{i_1}^R] \dots [x_{i_j}^R] D) \overset{*}{\vdash} (e, D)$.

(6) For any $v \in \Delta^+$, if $|v| \geqslant 1$, then by parts (k), (l), and (m) of the definition of δ_2, we have $(h(v)\,\$, Z_1 D) \overset{*}{\vdash} (\$, A) \vdash (e, e)$.

We make he following claim about the structure of the language accepted by M_2.

Claim 3. *$N(M_2)$ includes all of the elements in $N(M_1)$, except those (if any) of the form $f_2 f_{i_t} \dots f_{i_1} f_1 h(x_1 x_{i_1} \dots x_{i_t} x_2)\,\$$, where $t \geqslant 0$, and $3 \leqslant i_1, \dots, i_t \leqslant n$.*

In other words,

$$N(M_2) = \{f_2 f_{i_t} \dots f_{i_1} f_1 h(y_1 y_{i_1} \dots y_{i_t} y_2)\,\$ \mid t \geqslant 0,\ 3 \leqslant i_j, \dots, i_t \leqslant n,\ x_1 x_{i_1} \dots x_{i_t} x_2 \neq y_1 y_{i_1} \dots y_{i_t} y_2\} \cup \text{JUNK},$$

where JUNK is just a set of strings irrelevant to this argument.

We do not prove which strings are contained in JUNK. It is only necessary to insist that $\text{JUNK} \cap N(M_1) = \emptyset$.

Proof of Claim 3. Suppose that $z \in L(M_1)$. Then by Claim 1, z is of the form $f_2 f_{i_t} \dots f_{i_1} f_1 h(y_1 y_{i_1} \dots y_{i_t} y_2)\,\$$, for some $t \geqslant 0$, $3 \leqslant i_1, \dots, i_t \leqslant n$. We examine the two possibilities for z, where

$$h(y_1 y_{i_1} \dots y_{i_t} y_2) = h(x_1 x_{i_1} \dots x_{i_t} x_2)$$

and

$$h(y_1 y_{i_1} \dots y_{i_t} y_2) \neq h(x_1 x_{i_1} \dots x_{i_t} x_2).$$

We show that no element satisfying the first condition belongs to $N(M_2)$, whereas all elements satisfying the second are contained in $N(M_2)$.

Case 1: First, consider the case where $z = f_2 f_{i_t} \dots f_{i_1} f_1 h(x_1 x_{i_1} \dots x_{i_t} x_2)\$$, for some $t \geqslant 0$, $3 \leqslant i_1, \dots, i_t \leqslant n$. Then,

$$(z, Z_0) = (f_2 f_{i_t} \dots f_{i_1} f_1 h(x_1 x_{i_1} \dots x_{i_t} x_2)\$, Z_0) \vdash^*$$

$$\vdash^* (h(x_1 x_{i_1} \dots x_{i_t} x_2)\$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_1}^R] [x_1^R]) \quad \text{by facts (1)–(3),}$$

$$\vdash^* (\$, Z_1) \quad \text{by fact (4)(i),}$$

$$\vdash^* (e, J) \quad \text{by part (g) of the definition of } \delta_2.$$

Hence, $z \notin N(M_2)$.

Case 2: Now, consider the case where $z = f_2 f_{i_t} \dots f_{i_1} f_1 h(y_1 y_{i_1} \dots y_{i_t} y_2)\$$, for some $t \geqslant 0$, $3 \leqslant i_1, \dots, i_t \leqslant n$, where $h(x_1 x_{i_1} \dots x_{i_t} x_2) \neq h(y_1 y_{i_1} \dots y_{i_t} y_2)$. By the construction of the homomorphism h, we know that this implies that $x_1 x_{i_1} \dots x_{i_t} x_2 \neq y_1 y_{i_1} \dots y_{i_t} y_2$.

Observe that the lists X and Y are such that there exists an integer $0 \leqslant j \leqslant t$ that satisfies (*) below for some $k \geqslant 0$.

$$(*) \qquad \begin{cases} x_1 x_{i_1} \dots x_{i_j} = \# \alpha_0 q_0 \beta_0 \# \alpha_1 q_1 \beta_1 \# \dots \# \alpha_{k-1} q_{k-1} \beta_{k-1} \#, \\ y_1 y_{i_1} \dots y_{i_j} = \# \alpha_0 q_0 \beta_0 \# \alpha_1 q_1 \beta_1 \# \dots \# \alpha_{k-1} q_{k-1} \beta_{k-1} \# \alpha_k q_k \beta_k \#, \end{cases}$$

where $\alpha_0 q_0 \beta_0 = q_0 w$ and $\alpha_i, \beta_i \in (\Gamma - \{B\})^*$, $q_i \in K$ for $0 \leqslant i \leqslant k$. (Recall that w is the input tape to Turing machine M.) Again, we point out that for all $i \geqslant 3$, each x_i and y_i contains the same number of occurrences of states of M; that x_1 contains no state; and that y_1 contains the single state, q_0. Also, x_i ends in $\#$ iff y_i ends in $\#$. Equations (*) are trivially satisfied for $j = 0$. In this instance,

$$x_1 = \# \quad \text{and} \quad y_1 = \# q_0 w \# = \# \alpha_0 q_0 \beta_0 \#.$$

There may, however, be several such integers $\leqslant t$ that satisfy (*) above. Let p be the maximal such integer. Thus, for some $\alpha, \beta \in (\Gamma - \{B\})^*$,

$$q \in K, \qquad y_1 y_{i_1} \dots y_{i_p} = x_1 x_{i_1} \dots x_{i_p} \alpha q \beta \#.$$

Suppose simple machine M_2 is given the string z as its input tape. Then,

$$(z, Z_0) = (f_2 f_{i_t} \dots f_{i_1} f_1 h(y_1 y_{i_1} \dots y_{i_t} y_2)\$, Z_0)$$

$$\vdash^* (h(y_1 y_{i_1} \dots y_{i_t} y_2)\$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_1}^R] [x_1^R]) \quad \text{by facts (1)–(3).}$$

$$= (h(x_1 x_{i_1} \dots x_{i_p}) h(\alpha q \beta \#) h(y_{i_{p+1}} \dots y_{i_t} y_2)\$, Z_1 [x_2^R] [x_{i_t}^R] \dots$$

$$\dots [x_{i_t}^R] [x_1^R])$$

since $y_1 y_{i_1} \dots y_{i_p} = x_1 x_{i_1} \dots x_{i_p} \alpha q \beta \#$,

$$\vdash^* (h(\alpha q \beta \#) h(y_{i_{p+1}} \dots y_{i_t} y_2)\$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_{p+1}}^R])$$

by fact (4) (i).

Since p is maximal, a mismatch will occur while M_2 reads $h(\alpha q\beta\#)$. That is, some symbol in $h(\alpha q\beta\#)$ will not match the top of the pushdown store. The main thrust of what follows is to show that when this occurs, there are still enough input symbols remaining to be read so that the whole pushdown store can be emptied and the input tape accepted.

Subcase 2.1: Let us first suppose that $x_{i_{p+1}}...x_{i_t}x_2$ is a proper prefix of $\alpha q\beta\#$. Then $\alpha q\beta\# = x_{i_{p+1}} \dots x_{i_t}x_2 v$ for some $v \neq e$. Hence,

$$(z, Z_0) \vdash^* (h(\alpha q\beta\#)h(y_{i_{p+1}}...y_{i_t}y_2)\$, Z_1[x_2^R]\,[x_{i_t}^R]...[x_{i_{p+1}}^R]) \quad \text{by the argument above,}$$

$$= (h(x_{i_{p+1}}...x_{i_t}x_2v)h(y_{i_{p+1}}...y_{i_t}y_2)\$, Z_1[x_2^R]\,[x_{i_t}^R]...[x_{i_{p+1}}^R])$$

$$\vdash (h(v)h(y_{i_{p+1}}...y_{i_t}y_2)\$, Z_1) \quad \text{by fact (4)(i),}$$

$$\vdash (\$, A) \quad \text{by parts (k) and (l) of the definition of } \delta_2,$$

$$\vdash (e, e) \quad \text{by part (m) of the definition of } \delta_2.$$

So for this subcase, $z \in N(M_2)$, as desired.

Subcase 2.2: Now suppose that $x_{i_{p+1}} \dots x_{i_t}x_2$ is not a proper prefix of $\alpha q\beta\#$. Since p is maximal, we also know that $\alpha q\beta\#$ is not a prefix of $x_{i_{p+1}} \dots x_{i_t}x_2$. Therefore, there exist τ, ν, $\psi \in \Delta^*$, $d_1, d_2 \in \Delta$, $d_1 \neq d_2$, such that

$$\alpha q\beta\# = \tau d_1 \nu, \qquad x_{i_{p+1}} \dots x_{i_t}x_2 = \tau d_2 \psi.$$

But $\tau = x_{i_{p+1}} \dots x_{i_{p-q}}\tau_1$ for some $p \leqslant p+q \leqslant t$, $\tau_1 \in \Delta^*$, where

$$\tau_1 d_2 \text{ is a prefix of } \begin{cases} x_{i_{p+q+1}} & \text{if } t > p+q \text{ (then } x_{i_{p+q+1}} = \tau_1 d_2 \alpha \text{ for some } \alpha \in \Delta^*), \\ x_2 & \text{if } t = p+q \text{ (then } x_2 = \tau_1 d_2 \alpha \text{ for some } \alpha \in \Delta^*), \end{cases}$$

and $\alpha q\beta\# = x_{i_{p+1}} \dots x_{i_{p+q}}\tau_1 d_1 \nu$.

Therefore,

$$(z, Z_0) \vdash^* (h(\alpha q\beta\#)\, h(y_{i_{p+1}} \dots y_{i_t}y_2)\, \$, Z_1\, [x_2^R]\, [x_{i_t}^R] \dots [x_{i_{p+1}}^R])$$

$$= (h(x_{i_{p+1}} \dots x_{i_{p+q}}\tau_1 d_1 \nu)\, h(y_{i_{p+1}} \dots y_{i_t}y_2)\, \$, Z_1\, [x_2^R]\, [x_{i_t}^R] \dots [x_{i_{p+1}}^R])$$

$$\vdash^* \begin{cases} (h(\tau_1 d_1)\, h(\nu y_{i_{p+1}} \dots y_{i_t}y_2)\, \$, Z_1\, [x_2^R]\, [x_{i_t}^R] \dots [x_{i_{p+q+1}}^R]) & \text{if } t > p+q, \\ (h(\tau_1 d_1)\, h(\nu y_{i_{p+1}} \dots y_{i_t}y_2)\, \$, Z_1\, [x_2^R]) & \text{if } t = p+q, \end{cases}$$

$$\vdash^{*} \begin{cases} (h(\tau_1 d_1)\, h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_{p+q+2}}^R] [\alpha^R d_2\, \tau_1^R]) \\ \qquad \text{if } t > p+q+1, \\ (h(\tau_1 d_1)\, h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R] [\alpha^R d_2\, \tau_1^R]) \\ \qquad \text{if } t = p+q+1, \\ (h(\tau_1 d_1)\, h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [\alpha^R d_2\, \tau_1^R]) \quad \text{if } t = p+q, \end{cases}$$

$$\vdash^{*} \begin{cases} (d_1\, \not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_{p+q+2}}^R] [\alpha^R d_2]) \\ \qquad \text{if } t > p+q+1, \\ (d_1\, \not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R] [\alpha^R d_2]) \quad \text{if } t = p+q+1, \\ (d_1\, \not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [\alpha^R d_2]) \quad \text{if } t = p+q; \end{cases}$$

since $d_1 \neq d_2$, we have by part (h) of the definition of δ_2:

$$\vdash^{*} \begin{cases} (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R] [x_{i_t}^R] \dots [x_{i_{p+q+2}}^R]) & \text{if } t > p+q+1, \\ (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 [x_2^R]) & \text{if } t = p+q+1, \\ (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1) & \text{if } t = p+q, \end{cases}$$

$$= (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1 \gamma),$$

where

$$\gamma = \begin{cases} [x_2^R] [x_{i_t}^R] \dots [x_{i_{p+2}}^R] & \text{if } t > p+q+1, \\ [x_2^R] & \text{if } t = p+q+1, \\ e & \text{if } t = p+q. \end{cases}$$

So γ is a *proper* prefix of $[x_2^R] [x_{i_t}^R] \dots [x_{i_{p+1}}^R]$. Then $|\gamma| \leqslant t-p$.

Subcase 2.2.1: Assume that $\gamma = e$. Then we have

$$(\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1) = (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1)$$

$\vdash (h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, A)$ by part (k) of the definition of δ_2,

$\vdash^{*} (\$, A)$ by part (l) of the definition of δ_2,

$\vdash (e, e)$ by part (m) of the definition of δ_2.

In this subcase, $z \in N(M_2)$, as desired.

Subcase 2.2.2: Assume that $\gamma \neq e$. Then $\gamma = \hat{\gamma}[x_j^R]$ for some $2 \leqslant j \leqslant n$. Thus,

$$(\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2)\, \$, Z_1\, \gamma) = (\not\in h(\nu y_{i_{p+1}} \dots y_{i_t} y_2) \$, Z_1\, \hat{\gamma}[x_j^R])$$

$\vdash (h(\nu y_{i_{p+1}} \dots y_{i_t} y_2) \$, Z_1 \hat{\gamma} D)$ by definition (i) of δ_2.

$\vdash^{*} (\$, A)$ since $|y_{t_{p+1}} \dots y_{t_t} y_2| \geq t-p+1$, ($|y_{t_j}| \geq 1$ for all $j \in \{p+1, \dots, t, 2\}$) and $|\hat{\gamma}| = |\gamma| - 1 < t-p$ (hence, $|\hat{\gamma}| < |\nu y_{t_{p+1}} \dots y_{t_t} y_2|$) *and* by facts (5) and (6),

$\vdash (e, e)$ by part (m) of the definition of δ_2.

Hence, we have $z \in N(M_2)$ for this subcase, also.

This proves Claim 3. □

We temporarily return to the example illustrated after Claim 1. Let M be the Turing machine we defined at that point with blank input tape, and let X and Y be the corresponding lists constructed from M. From these lists, we define simple machines M_1 and M_2 as outlined above.

Construct the simple machine M_1:

$$M_1 = (\{1, q_0, q_f, \#, \not{\epsilon}, \$\} \cup \{f_1, \dots, f_q\}, \delta_1, \Gamma_1, Z_0)$$

where

$$\Gamma_1 = \{Z_0, Z_1, C\} \cup \{[z] \mid z \in \{1, q_0, q_f, \#\}^+, |z| \leq 3\} \cup \{[z\not{\epsilon}] \mid z \in \{1, q_0, q_f, \#\}^*, |z| < 3\};$$

and

(a) $\delta_1(f_2, Z_0) = Z_1[\#]C$,

(b) $\delta_1(f_i, C) = [y_i^R]C$, for each $i = 3, \dots, 9$,

(c) $\delta_1(f_1, C) = [\# q_0 \#]$.

For each $d \in \{1, q_0, q_f, \#\}$, $z \in \{1, q_0, q_f, \#\}^*$, $|z| < 3$, we have

(d) $\delta_1(d, [zd]) = [z\not{\epsilon}]$,

(e) $\delta_1(\not{\epsilon}, [z\not{\epsilon}]) = [z]$, if $z \neq e$,

(f) $\delta_1(\not{\epsilon}, [\not{\epsilon}]) = e$,

(g) $\delta_1(\$, Z_1) = e$.

We illustrate the operation of M_1 on a simple input tape. Consider the string

$$f_2 f_3 f_1 h(y_1 y_3 y_2)\$ = f_2 f_3 f_1 h(\# q_0 \# 1 \#)\$.$$

Then,

$(f_2 f_3 f_1 h(\# q_0 \# 1 \#)\$, Z_0) \vdash (f_3 f_1 h(\# q_0 \# 1 \#)\$, Z_1[\#]C)$ part (a) of δ_1,

$\vdash (f_1 h(\# q_0 \# 1 \#)\$, Z_1[\#][1]C)$ part (b),

$\vdash (h(\# q_0 \# 1 \#)\$, Z[\#][1][\# q_0 \#])$ part (c),

$\vdash (\not{\epsilon} h(q_0 \# 1 \#)\$, Z_1[\#][1][\# q_0 \not{\epsilon}])$ part (d),

$\vdash (h(q_0 \# 1 \#)\$, Z_1[\#][1][\# q_0])$ part (e),

$\vdash (\not{\epsilon} h(\# 1 \#)\$, Z_1[\#][1][\# \not{\epsilon}])$ part (d),

$\vdash (h(\#1\#)\$, Z_1[\#]\,[1]\,[\#])$ part (e),
$\vdash (\not\!h(1\#)\$, Z_1[\#]\,[1]\,[\not\!])$ part (d),
$\vdash (h(1\#)\$, Z_1[\#]\,[1])$ part (f),
$\vdash (\not\!\#\not\!\$, Z_1[\#]\,[\not\!])$ part (d),
$\vdash (\#\not\!\$, Z_1[\#])$ part (f),
$\vdash (\not\!\$,\ Z_1[\not\!])$ part (d),
$\vdash (\$, Z_1)$ part (f),
$\vdash (e,e)$ part (g).

Hence, the string $f_2f_3f_1h(y_1y_3y_2)\$ \in N(M_1)$.

The construction of M_2 follows: Let

$$M_2 = (\{1, q_0, q_f, \#, \not\!, \$\} \cup \{f_1, \ldots, f_9\}, \delta_2, \Gamma_2, Z_0),$$

where

$$\Gamma_2 = \{Z_0, Z_1, A, C, D, J\}$$
$$\cup \{[z] \mid z \in \{1, q_0, q_f, \#\}^+, |z| \leqslant 3\}$$
$$\cup \{[z\not\!] \mid z \in \{1, q_0\, q_f, \#\}^*, |z| < 3\};$$

and

(a) $\delta_2(f_2, Z_0) = Z_1[\#\#q_f]C$,
(b) $\delta_2(f_i, C) = [x_i^R]C$, for $i = 3, \ldots, 9$,
(c) $\delta_2(f_1, C) = [\#]$.

For each $d \in \{1, q_0, q_f, \#\}$, $z \in \{1, q_0, q_f, \#\}^*$, $|z| < 3$, we have

(d) $\delta_2(d, [zd]) = [z\not\!]$,
(e) $\delta_2(\not\!, [\not\!]) = e$,
(f) $\delta_2(\not\!, [z\not\!]) = [z]$, if $z \neq e$,
(g) $\delta_2(\$, Z_1) = J$.

For each $d_1, d_2 \in \{1, q_0, q_f, \#\}$, $d_1 \neq d_2$, and $z, \bar{z} \in \{1, q_0, q_f, \#\}^*$, $|z| < 3$, $|\bar{z}| \leqslant 3$,

(h) $\delta_2(d_1, [zd_2]) = e$,
(i) $\delta_2(\not\!, [z]) = D$, $\bar{z} \neq e$,
(j) $\delta_2(d_1, D) = e$.

For each $d_3 \in \{1, q_0, q_f, \#, \not\!\}$,

(k) $\delta_2(d_3, Z_1) = A$,
(l) $\delta_2(d_3, A) = A$,
(m) $\delta_2(\$, A) = e$.

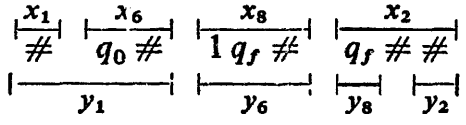
We illustrate the operation of M_2 on two sample input tapes.

Sample 1: Consider $y_1y_3y_2 = \# q_0 \# 1 \#$, $x_1x_3x_2 = \# 1q_f \#\#$. This is to be the same input tape as the one illustrated for machine M_1.

$$
\begin{aligned}
(f_2f_3f_1h(y_1y_3y_2)\$,Z_0) &= (f_2f_3f_1h(\#q_0\#1\#)\$,Z_0)\\
&\vdash (f_3f_1h(\#q_0\#1\#)\$,Z_1[\#\#q_f]C) && \text{part (a) of } \delta_2,\\
&\vdash (f_1h(\#q_0\#1\#)\$,Z_1[\#\#q_f][1]C) && \text{part (b)},\\
&\vdash (h(\#q_0\#1\#)\$,Z_1[\#\#q_f][1][\#]) && \text{part (c)},\\
&\vdash (\not{c}h(q_0\#1\#)\$,Z_1[\#\#q_f][1][\not{c}]) && \text{part (d)},\\
&\vdash (h(q_0\#1\#)\$,Z_1[\#\#q_f][1]) && \text{part (e)},\\
&\vdash (\not{c}h(\#1\#)\$,Z_1[\#\#q_f]) && \text{part (h)},\\
&\vdash (h(\#1\#)\$,Z_1D) && \text{part (i)},\\
&\vdash (\not{c}1\not{c}\#\not{c}\$,Z_1) && \text{part (j)},\\
&\vdash (1\not{c}\#\not{c}\$,A) && \text{part (k)},\\
&\overset{*}{\vdash} (\$,A) && \text{part (l)},\\
&\vdash (e,e).
\end{aligned}
$$

So, $f_2f_3f_1h(y_1y_3y_2)\$ \in N(M_2)$, and $f_2f_3f_1h(y_1y_3y_2)\$ \in N(M_1)$, where $y_1y_3y_2 = \#q_0\#1\#\not= \#1q_f\#\# = x_1x_3x_2$.

Sample 2: Consider



Then

$$
\begin{aligned}
&(f_2f_8f_6f_1h(y_1y_6y_8y_2)\$,Z_0) =\\
&= (f_2f_8f_6f_1h(\#q_0\#1q_f\#q_f\#\#)\$,Z_0)\\
&\vdash (f_8f_6f_1h(\#q_0\#1q_f\#q_f\#\#)\$,Z_1[\#\#q_f]C) && \text{part (a) of } \delta_2,\\
&\vdash (f_6f_1h(\#q_0\#1q_f\#q_f\#\#)\$,Z_1[\#\#q_f][\#q_f1]C) && \text{part (b)},\\
&\vdash (f_1h(\#q_0\#1q_f\#q_f\#\#)\$,Z_1[\#\#q_f][\#q_f1][\#q_0]C) && \text{part (b)},\\
&\vdash (h(\#q_0\#1q_f\#q_f\#\#)\$,Z_1[\#\#q_f][\#q_f1][\#q_0][\#]) && \text{part (c)},\\
&\overset{*}{\vdash} (h(q_0\#1q_f\#q_f\#\#)\$,Z_1[\#\#q_f][\#q_f1][\#q_0]) && \text{fact (4)(i)},\\
&\overset{*}{\vdash} (h(1q_f\#q_f\#\#)\$,Z_1[\#\#q_f][\#q_f1]) && \text{fact (4)(i)},\\
&\overset{*}{\vdash} (h(q_f\#\#)\$,Z_1[\#\#q_f]) && \text{fact (4)(i)},\\
&\overset{*}{\vdash} (\$,Z_1) && \text{fact (4)(i)},\\
&\vdash (e,J) && \text{part (g)}.
\end{aligned}
$$

Thus,

$$
\begin{aligned}
f_2f_8f_6f_1h(\#q_0\#1q_f\#q_f\#\#)\$ &= f_2f_8f_6f_1h(y_1y_6y_8y_2)\$\\
&= f_2f_8f_6f_1h(x_1x_6x_8x_2)\$ \notin N(M_2).
\end{aligned}
$$

We now return to the proof of Theorem 3.1.

Claim 4 below follows immediately from Claims 2 and 3.

Claim 4. *$N(M_1) \subseteq N(M_2)$ iff*

$$N(M_1) \cap \{f_2 f_{i_t} \dots f_{i_1} f_1 h(x_1 x_{i_1} \dots x_{i_t} x_2)\$ \mid t \geqslant 0, 3 \leqslant i_1, \dots, i_t \leqslant n\} = \emptyset$$

iff there is no (possibly empty) sequence of integers $i_1, \dots, i_t$, $t \geqslant 0$, with $3 \leqslant i_1, \dots, i_t \leqslant n$, such that $x_1 x_{i_1} \dots x_{i_t} x_2 = y_1 y_{i_1} \dots y_{i_t} y_2$.

From Claims 1 and 4, we obtain

Claim 5. *$N(M_1) \not\subseteq N(M_2)$ iff Turing machine M halts and accepts tape w.*

Since we can construct two simple machines from any given Turing machine and any given input tape, we have reduced the halting problem for Turing machines to the complement of the inclusion problem for simple machines. Hence, the inclusion problem for simple machines is undecidable. In fact, as pointed out earlier, we have shown that the inclusion problem is not even semi-decidable.

This proves Theorem 3.1. □

4. Single-turn restriction

We consider the family of simple machines with a restriction of the movement of the pushdown store.

Definion 4.1. A simple machine $M = (\Sigma, \Gamma, \delta, Z_0)$ is said to be a *1-turn simple machine* if for all $a, b \in \Sigma$, $v, w \in \Sigma^*$, $Z_0, Z_1, Z_2 \in \Gamma$, $\alpha, \beta, \gamma \in \Gamma^*$,

$$\text{if } (vZ_0) \overset{*}{\underset{M}{\vdash}} (awb, \alpha Z_1) \underset{M}{\vdash} (wb, \alpha) \overset{*}{\underset{M}{\vdash}} (b, \beta Z_2) \underset{M}{\vdash} (e, \beta\gamma), \text{ then } |\gamma| \leqslant 1.$$

Thus, M is a 1-turn simple machine if once it pops a symbol from the pushdown store, it never pushes more than one symbol at a time (each move) back onto the store.

In the proof of Theorem 3.1, simple machines M_1 and M_2 are constructed so that they both are 1-turn simple machines. Thus, we have the following.

Corollary 4.2. *If M_1 and M_2 are both 1-turn simple machines, then it is not semi-decidable whether $N(M_1) \subseteq N(M_2)$.*

Consider the $LL(k)$ grammars defined in [11]. Rosenkrantz and Stearns [11] have shown that if G_1 and G_2 are both $LL(k)$ grammars, then $L(G_1) = L(G_2)$ is decidable. The decidability of whether $L(G_1) \subseteq L(G_2)$ was posed as an open question. For $k = 0$, $L(G_1)$ and $L(G_2)$ must each have at most one member. Thus, for $k = 0$, we can easily decide if $L(G_1) \subseteq L(G_2)$. But it can be shown that the family of languages generated by $LL(1)$ grammars without e-rules (productions of the form $Z \to e$, for non-terminal Z) is precisely the family of languages accepted by

simple machines [11]. These grammars are also known as the simple $LL(k)$ grammars defined in Aho and Ullman [1]. By translating such a grammar into Greibach normal form, we obtain a grammar that is easily translatable into the coresponding simple machine [9]. The next result follows immediately from Theorem 3.1.

Corollary 4.3. *If G_1 and G_2 are both $LL(k)$ grammars, $k \geqslant 1$, then it is not semi-decidable whether $L(G_1) \subseteq L(G_2)$.*

5. Concluding remarks

Theorem 3.1 and its corollaries answer three of the open questions that had been posed in language theory. The new techniques employed would appear to have applications in several other areas. In fact, we are now presented with a means of showing how the inclusion problem for free monadic recursion schemes [2] (or, equivalently, deBakker–Scott schemes [4]) that do not use the identity function is undecidable. This is most easily accomplished by using these schemes to encode the operations of simple machines [5]. It is now hoped that the methods presented herein will prove useful in additional areas.

References

[1] A. V. Aho and J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1: Parsing (Prentice-Hall, Englewood Cliffs, N.J., 1972).

[2] E. Ashcroft, Z. Manna and A. Pnueli, Decibale properties of monadic functional schemas, J. ACM 20(3) (1973) 489–499.

[3] M. Bird, The equivalence problem for deterministic two-tape automata, J. Comput. System Sci. 7 (1973) 218–236.

[4] J. W. DeBakker and D. Scott, A theory of programs, unpublished memo, Vienna (August 1969).

[5] E. P. Friedman, Equivalence problems in monadic recursion schemas, 14th Annual Symp. on Switching and Automata Theory, University of Iowa, Iowa City (October 1973) 26–33.

[6] S. Ginsburg, The Mathematical Theory of Context-free Languages (McGraw-Hill, New York, 1966).

[7] M. A. Harrison and I. M. Havel, Real-time strict deterministic languages, SIAM J. Comput. 1 (1972) 333–349.

[8] J. E. Hopcroft and J. D. Ullman, Formal Languages and Their Relation to Automata (Addison-Wesley, Reading, Mass., 1968).

[9] A. J. Korenjak and J. E. Hopcroft, Simple deterministic languages, IEEE Conf. Record of 7th Annual Symp. on Switching and Automata Theory, 36–46.

[10] M. O. Rabin and D. Scott, Finite automata and their decision problems, in: E. F. Moore, ed., Sequential Machines: Selected Papers (Addison-Wesley, Reading, Mass., 1964) 63–91.

[11] D. J. Rosenkrantz and R. E. Stearns, Properties of deterministic top-down grammars, Information and Control 17(3) (1970) 226–256.

[12] L. G. Valiant, Decision procedures for families of deterministic pushdown automata, Ph.D. Thesis, Department of Computer Science, University of Warwick, Coventry (July 1973).