KAI SALOMAA

D. WOOD

SHENG YU

Pumping and pushdown machines

Informatique théorique et applications, tome 28, nº 3-4 (1994), p. 221-232

<http://www.numdam.org/item?id=ITA_1994__28_3-4_221_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 28, n° 3-4, 1994, p. 221 à 232)

PUMPING AND PUSHDOWN MACHINES (*)

by Kai Salomaa [1], D. Wood [1] and Sheng Yu [1]

Abstract. – *We present a new technique for proving that certain languages are not accepted by any pushdown machine of a specific type. We define the technique by example. Essentially, we combine context-free pumping lemmas with the Floydian view of pushdown machines as transition systems or finite-state machines, to give machine-based pumping lemmas. We give two applications and explications of the technique.*

The first application is to reprove the well-known results that the languages $E = \{a^i b^i, a^i b^{2i} : i \geqq 1\}$ *and* $\{xx^R : x \in \{a, b\}^*\}$ *are not deterministic context-free languages. As the second application, we prove that the context-free language* $\{ab^{i_1} ab^{i_2} a \ldots ab^{i_t} : t \geqq 2$ *and* $i_t = i_j$ *for some* j, $1 \leqq j < t\}$ *has a nondeterminism-degree complexity* $\Omega(\log n)$. *Therefore, there are at least two levels in the suspected infinite hierarchy of context-free languages with respect to this complexity measure.*

1. INTRODUCTION

We explore a beautiful idea originated by Bob Floyd about 20 years ago; namely, treating machines, in particular pushdown machines, as being made up of a finite number of devices that form transition systems, and treating instruction (or transition) sequences as first-class objects. The idea has resurfaced twice since Kurki-Suonio [6] investigated their basic properties in 1975. First, Jonathan Goldstine, in a sequence of three articles [2, 3, 4] rediscovered Floyd's idea and he applied it to AFA theory and to the theory of pushdown and Turing machines. Second, Floyd and Beigel have written a text [1] on the language of machines that develops the ideas rigorously and also applies them consistently to the standard collection of machines that are studied in a first theory of computation course. Wood [11] is also using this approach in the second edition of his text.

(*) This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada.

[1] Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada.

Based on the state diagram approach to pushdown machines, we can show that the sets of well-formed instruction sequences are context-free. Therefore we can apply any of the context-free pumping lemmas to them. We can then combine knowledge of the pumping positions of an instruction sequence with properties of the underlying pushdown machine to provide contradictions.

We believe that it is, in general, easier to prove pumping lemmas for context-free grammars than it is to prove pumping lemmas for pushdown machines. There have been few attempts to prove pumping lemmas for pushdown machines directly. One exception can be found in Floyd and Beigel's text [1]. As a result, we can apply pumping lemmas for context-free languages to instruction-sequence languages of pushdown machines to give system-dependent pumping lemmas for pushdown machines. We apply these system-dependent pumping lemmas to establish three results about pushdown languages.

The first application, in Section 4, is to reprove two well-known results; namely, the languages

$$\{a^i\, b^i : j = i \text{ or } j = 2i \text{ and } i \geqq 0\}$$

and

$$\{xx^R : x \in \{a, b\}^*\}$$

are not deterministic context-free languages. The second application, in Section 5, is to prove that the language

$$\{ab^{i_1}\, ab^{i_2}\, a \ldots ab^{i_t} : t \geqq 2 \text{ and } i_t = i_j \text{ for some } j,\ 1 \leqq j < t\}$$

has nondeterminism-degree complexity $\Omega(\log n)$.

We begin, in Section 2, by explicating the relationship between instruction sequences for pushdown machines and context-free languages. Then in Section 3 we recall two pumping lemmas and discuss system-dependent and system-independent pumping lemmas.

2. PRELIMINARIES

We give a nonstandard definition of a pushdown machine that is similar to the definition of Floyd (see the report of Kurki-Suonio [6] and the text of Floyd and Beigel [1]).

Given an alphabet Σ, we let Σ_λ denote $\Sigma \cup \{\lambda\}$ and Σ^{-1} denote $\{a^{-1} : a \in \Sigma\}$. We treat a^{-1} as the right inverse of a; thus, $aa^{-1} \equiv \lambda$.

Given the identities $aa^{-1} \equiv \lambda$, for all $a \in \Sigma$, a string $x \in (\Sigma \cup \Sigma^{-1})^*$ is **well formed** with **content** $y \in \Sigma^*$ if either $x \in \Sigma^*$ and $x = y$, or $x = uaa^{-1}v$, for some u and v in $(\Sigma \cup \Sigma^{-1})^*$, and uv is well formed with content y. Each well-formed string in $(\Sigma \cup \Sigma^{-1})^*$ corresponds to exactly one content string; thus, we denote the content of a well-formed string $x \in (\Sigma \cup \Sigma^{-1})^*$ by content (x). When the content of a string x is λ, we say that the string is **balanced**.

We need the notion of a restricted morphism, called a **projection**. Given two alphabets Σ and Γ such that $\Sigma \subseteq \Gamma$, a function π_Σ is a projection with respect to Γ if, for all $a \in \Sigma$, $\pi_\Sigma(a) = a$ and, for all $a \notin \Sigma$, $\pi_\Sigma(a) = \lambda$. Clearly, a projection is a morphism that preserves some symbols and erases others; thus, it has a natural inverse. An **inverse projection** $\pi_\Sigma^{-1} : \Sigma^* \to 2^{\Gamma^*}$ is defined by, for all $x \in \Sigma^*$,

$$\pi_\Sigma^{-1}(x) = \{y : y \in \Gamma^* \text{ and } \pi_\Sigma(y) = x\}.$$

It is well known that the families of regular languages and context-free languages are closed under projection and inverse projection.

We specify a **pushdown machine** M with a tuple $(Q, \Sigma, \Gamma, \delta, s, F, Z)$, where Q is a **state alphabet**, Σ is an **input alphabet**, Γ is a **pushdown alphabet**, $\delta \subseteq Q \times \Sigma_\lambda \Gamma^{-1} \Gamma^* \times Q$ is a **finite transition relation**, where $\Gamma^{-1} = \{a^{-1} : a \in \Gamma\}$, s is a **start state**, $F \subseteq Q$ is a set of **final states**, and $Z \in \Gamma$ is the **initial pushdown symbol**. The inverse pushdown symbol a^{-1} denotes the **pop operation** that pops the symbol a from the top of the pushdown, whereas a string $x \in \Gamma^*$ denotes the **push operation** that pushes the string x onto the top of the pushdown. Observe that we view M as a finite-state machine (or transition system) with input alphabet $\{x : (p, x, q) \in \delta\}$.

Given a pushdown machine $M = (Q, \Sigma, \Gamma, \delta, s, F, Z)$, we consider a tuple in δ to be an **instruction** and we define an instruction sequence as follows. A sequence $Z p_0 x_1 p_1 \cdots x_m p_m$ is an **instruction sequence** if $p_0 = s$ and $(p_i, x_{i+1}, p_{i+1}) \in \delta$, $0 \leq i < m$. It is a **well-formed instruction sequence** if, in addition, $\pi_{\Gamma \cup \Gamma^{-1}}(Z p_0 x_1 p_1 \cdots x_m p_m)$ is well formed.

We associate three languages with a pushdown machine $M = (Q, \Sigma, \Gamma, \delta, s, F, Z)$. First, we have the **instruction sequence language**, $ISL(M)$, that is defined by

$$ISL(M) = \{Z p_0 x_1 p_1 \cdots x_m p_m : Z p_0 x_1 p_1 \cdots x_m p_m \text{ is an instruction sequence and } p_m \in F\}.$$

Second, we have the **well-formed instruction sequence language**, $WFL(M)$, that is defined by

$$WFL(M) = \{Z\,p_0\,x_1\,p_1 \cdots x_m\,p_m : Z\,p_0\,x_1\,p_1 \cdots x_m\,p_m \text{ is a well-formed instruction sequence and } p_m \in F\}.$$

Lastly, we have the **(input) language**, $L(M)$, that is defined by

$$L(M) = \{x : Z\,p_0\,x_1\,p_1 \cdots x_m\,p_m \text{ is a well-formed instruction sequence, } p_m \in F, \text{ and } \pi_\Sigma(Z\,p_0\,x_1\,p_1 \cdots x_m\,p_m) = x\}.$$

The first language is regular since its sequences are given by paths in the state diagram of the machine. The second language is context-free since its sequences are given by paths in the state diagram of the machine that manipulate the pushdown correctly. The third language corresponds to one definition of acceptance by a pushdown machine, namely, acceptance by final state. (If we want to accept by final state and empty pushdown, then we must use balanced instruction sequences, based on balanced pushdown strings rather than on well-formed instruction sequences.)

We now express $WFL(M)$ and $L(M)$ in terms of $ISL(M)$ using the **Dyck language**, $D_{\Gamma \cup \Gamma^{-1}}$, over the alphabet $\Gamma \cup \Gamma^{-1}$, and the language $P_{\Gamma \cup \Gamma^{-1}}$ of prefixes of all strings in $D_{\Gamma \cup \Gamma^{-1}}$. The Dyck language, $D_{\Gamma \cup \Gamma^{-1}}$, is defined as the set of all balanced strings over $(\Gamma \cup \Gamma^{-1})^*$, whereas $P_{\Gamma \cup \Gamma^{-1}}$ is the set of all well-formed strings over $(\Gamma \cup \Gamma^{-1})^*$. We interpret a symbol as a left-labeled bracket and an inverse symbol as a right-labeled bracket. We can now express $WFL(M)$ is terms of $ISL(M)$ as follows:

$$WFL(M) = ISL(M) \cap \pi^{-1}_{\Gamma \cup \Gamma^{-1}}(P_{\Gamma \cup \Gamma^{-1}}).$$

In addition, we can express $L(M)$ in terms of $WFL(M)$ as follows:

$$L(M) = \pi_\Sigma(WFL(M)).$$

Note that these simple equations establish that every pushdown language is a context-free language. If we prefer to have pushdown machines accept strings by both final state and empty pushdown, then we must replace the Dyck prefix language with the Dyck language in the equation that expresses $WFL(M)$ is terms of $ISL(M)$.

3. PUSHDOWN MACHINES AND PUMPING

There are two varieties of pumping lemmas for language families. First, we have the system-dependent pumping lemmas that present a pumping lemma in terms of a class of language-description systems. For example, we can give a pumping lemma for context-free grammars that describes pumping for syntax trees, which are sufficiently tall. Similarly, we can give a pumping lemma for deterministic finite-state machines that describes pumping for computations that are sufficiently long.

Second, we have system-independent pumping lemmas that present pumping lemmas without any reference to a language-description system. The following pumping lemma for context-free languages has this form. We will use it in Section 4.

LEMMA 3.1: *For each context-free language L there is a constant $p \geqq 0$ such that, for all strings z in L of length at least p, there are strings u, v, w, x, and y such that the following four conditions hold:*

1. $z = uvwxy$.
2. $|vx| \geqq 1$.
3. $|vwx| \leqq p$.
4. *For all* $i \geqq 0$, $uv^i wx^i y \in L$.

We also recall the following pumping lemma for context-free languages (see Harrison's text [5] for more details). We use it in Section 5.

LEMMA 3.2: *For each context-free language L there is a constant $p \geqq 0$ such that, for each $z \in L$ and for any set S of distinguished positions in z, if $\#S \geqq p$, then there are strings $v_1, \ldots, v_5$ such that the following four conditions hold (where S_i is the set of distinguished positions that corresponds to the substring v_i, $1 \leqq i \leqq 5$):*

1. $z = v_1 v_2 v_3 v_4 v_5$.
2. *Either* $S_1, S_2, S_3 \neq \emptyset$ *or* $S_3, S_4, S_5 \neq \emptyset$.
3. $\#(S_2 \cup S_3 \cup S_4) \leqq p$.
4. *For all* $i \geqq 0$, $v_1 v_2^i v_3 v_4^i v_5 \in L$.

Since we can always transform system-dependent pumping lemmas into system-independent pumping lemmas, why do we make the distinction? The reason is simple. We can use system-dependent pumping lemmas to provide system-dependent information about sentences and their generation or acceptance. We will demonstrate this idea in both Sections 4 and 5.

4. APPLICATION I: TWO NONDETERMINISTIC CONTEXT-FREE LANGUAGES

We reprove the well-known results that the languages

$$E = \{a^i b^i, a^i b^{2i} : i \geqq 1\}$$

and

$$H = \{x x^R : x \in \{a, b\}^*\}$$

are not deterministic context-free languages. Since the family of deterministic context-free languages is defined by the class of deterministic pushdown machines that accept by final state, it is exactly the set of languages $L(M)$, where M is a deterministic pushdown machine. We begin by defining this subclass of pushdown machines.

A pushdown machine $(Q, \Sigma, \Gamma, \delta, s, F, Z)$ is deterministic if it satisfies the following two conditions.

1. For all $p \in Q$, for all $A \in \Sigma_\lambda$, and for all $B \in \Gamma$, there is at most one instruction of the form $(p, AB^{-1} x, q)$ in δ.

2. For all $p \in Q$ and for all $B \in \Gamma$, if there is an instruction of the form $(p, B^{-1} x, q)$ in δ, then for all $a \in \Sigma$, there is no instruction of the form $(p, a B^{-1} y, r)$ in δ.

Without more ado we reprove the following theorem using our new technique.

THEOREM 4.1: *$E = \{a^i b^i, a^i b^{2i} : i \geqq 1\}$ is not a deterministic context-free language.*

Proof: We argue by contradiction. Assume that $E = L(M)$ for some deterministic pushdown machine $M = (Q, \Sigma, \Gamma, \delta, s, F, Z)$, where $\Sigma = \{a, b\}$. Since $WFL(M)$ is context-free and infinite, we can apply the context-free pumping lemma (Lemma 3.1) to it. Thus, there is a constant $p > 0$ such that, for all strings W in $WFL(M)$ of length at least p, we can decompose W into $uvxyz$, where $|vy| \geqq 1$ and $|vxy| \leqq p$, and, for all $i \geqq 0$, $uv^i xy^i z \in WFL(M)$.

We consider the pumping of the well-formed instruction sequence W whose input string is $w = \pi_\Sigma(W) = a^{2p} b^{4p}$. Now if v is empty, then y must be nonempty. Thus y can have one of four forms $\pi_\Sigma(y) = \lambda$, $\pi_\Sigma(y) \in a^+$, $\pi_\Sigma(y) \in b^+$, or $\pi_\Sigma(y) \in a^+ b^+$. The last three forms lead to an immediate contradiction when we consider the string $uv^2 xy^2 z$. The first possibility implies that y corresponds to a sequence of null-input instructions;

therefore, $uv^2 xy^2 z$ is well-formed and is an accepting sequence for w. Since M is deterministic and we have two distinct accepting sequences, we have obtained a contradiction. We can argue in a similar manner if y is empty and v is nonempty. Hence both v and y must be nonempty.

If v and y are both nonempty, then by similar arguments we can deduce that $\pi_\Sigma(v)$ and $\pi_\Sigma(y)$ are nonempty. If $\pi_\Sigma(vy) \in \lambda + a^+ + b^+$, $\pi_\Sigma(v) \in a^+ b^+$, or $\pi_\Sigma(y) \in a^+ b^+$, then $uv^2 xy^2 z$ provides a contradiction. Therefore, $\pi_\Sigma(v) \in a^+$ and $\pi_\Sigma(y) \in b^+$ is the only remaining possibility (the reverse assignment is impossible).

Now, letting $|v|_a = l$, we must have that $|y|_b = 2l$; otherwise, the input string uxz is outside E. Since $|vxy| \leqq p$, the number of b's in xy is at most p; therefore, the decomposition does not affect the rightmost $3p$ b's.

We now use this analysis in conjunction with the determinism of M to obtain a contradiction. Since M is deterministic, the well-formed instruction sequence W' of the string $w' = \pi_\Sigma(W') = a^{2p} b^{2p}$ must be a prefix of the well-formed instruction sequence W for w. Moreover, a corresponding decomposition of W' is $uvxyz'$, where z' is a prefix of z. Lastly, observe that uxz' is in $WFL(M)$, yet $|uxz'|_a \neq |uxz'|_b$ and $2|uxz'|_a \neq |uxz'|_b$. Thus we have obtained a contradiction and E is not a deterministic context-free language. $\square$

THEOREM 4.2: *$H = \{xx^R : x \in \{a, b\}^*\}$ is not a deterministic context-free language.*

Proof: We argue by contradiction. If H is deterministic context-free, then clearly the language $H' = H \cap (ba^+ b)^*$ is also deterministic context-free. Assume that $H' = L(M)$ for some deterministic pushdown machine $M = (Q, \Sigma, \Gamma, \delta, s, F, Z)$, where $\Sigma = \{a, b\}$. Since $WFL(M)$ is context-free and infinite, we can apply the context-free pumping lemma (Lemma 3.1) to it. Thus, there is a constant $p > 0$ such that, for all strings W in $WFL(M)$ of length at least p, we can decompose W into $uvxyz$, where $|vy| \geqq 1$ and $|vxy| \leqq p$, and, for all $i \geqq 0$, $uv^i xy^i z \in WFL(M)$.

We consider the pumping of the well-formed instruction sequence W whose input string $w = \pi_\Sigma(W) = ba^p bba^p bba^p bba^p b$. If $\pi_\Sigma(v)$ or $\pi_\Sigma(y)$ belongs to $\Sigma^+ - a^+$, then we obtain a contradiction by considering the string $\pi_\Sigma(uv^2 xy^2 z)$ which is not in H'. If $\pi_\Sigma(v) = \pi_\Sigma(y) = \lambda$, then W and $uv^2 xy^2 z$ are two different well-formed instruction sequences that correspond to the input string w, an impossibility since M is deterministic. Thus, necessarily $\pi_\Sigma(vy) \in a^+$.

We now consider the string $uv^2 xy^2 z$ and see that the only possible decomposition is that $\pi_\Sigma(v) = \pi_\Sigma(y) = a^k$, $1 \leqq k < p$, and the a's of v and of y belong to the second and the third substring $ba^p b$ of w, respectively. Let W' be the well-formed instruction sequence whose input string $w' = \pi_\Sigma(W') = ba^p bba^p b$. Note that $w' \in H'$. Since M is deterministic, it follows that W' is a prefix of W and $W' = uvx'$, where x' is a prefix of x. Since x' ends with a final state, it follows that $uv^2 x' \in WFL(M)$, which is a contradiction because $\pi_\Sigma(uv^2 x') = ba^p bba^{p+k} b$.

Hence H is not a deterministic context-free language. $\square$

5. APPLICATION II: A NEW LOWER BOUND

Nondeterminism, which is similar to time and space, is a resource that can and should be measured. Two measurements of nondeterminism for pushdown automata were introduced by Vermeir and Savitch [10] and investigated by Salomaa and Yu [7, 9]. Recently, a new measure of nondeterminism for pushdown machines, which is more natural and compatible with other complexity measures, has been introduced [8]. The basic unit of this measure is a nondeterminism computational step. A computational step of a pushdown machine is a **nondeterminism computational step** if it is one of at least two applicable instructions. An instruction is applicable if either it is defined for the current state, the current input symbol, and current top of pushdown symbol or it is a null-input instruction that is defined for the current state and current top of pushdown symbol.

Let M be a pushdown machine and $C = Z p_0 x_1 p_1 \cdots x_m p_m$ be a well-formed instruction sequence for M. The computational step of C defined by an instruction (p_i, x_{i+1}, p_{i+1}) in C, $0 \leqq i < m$, is a *nondeterministic computational step* if one of the following four possibilities holds:

1. $x_{i+1} = au^{-1}v$, $a \in \Sigma$, $u \in \Gamma$, $v \in \Gamma^*$ and there are $v' \in \Gamma^*$ and $p \in Q$ such that

(*a*) $(v', p) \neq (v, p_{i+1})$ and $(p_i, au^{-1}v', p) \in \delta$. (The machine can execute another instruction by reading the next input symbol a.)

(*b*) $(p_i, u^{-1}v', p) \in \delta$. (The machine can execute a null-input instruction instead of reading the next input symbol a.)

2. $x_{i+1} = u^{-1}v$, $u \in \Gamma$, $v \in \Gamma^*$, a is the first symbol of $\pi_\Sigma(x_{i+2} \cdots x_m)$ and there are $v' \in \Gamma^*$ and $p \in Q$ such that $(p_i, au^{-1}v', p) \in \delta$. (Instead

of executing a null-input instruction (p_i, x_{i+1}, p_{i+1}), the machine can read the next input symbol a.)

3. $x_{i+1} = u^{-1} v$, $u \in \Gamma$, $v \in \Gamma^*$ and there are $v' \in \Gamma^*$, $p \in Q$, and $(v', p) \neq (v, p_{i+1})$ such that $(p_i, u^{-1} v', p) \in \delta$. (The machine can execute another null-input instruction.)

Let $NCS(C)$ be the number of nondeterministic computational steps in C and define the **nondeterminism degree ND(x)** of a string x with respect to M as 0 if x is not in $L(M)$ and, otherwise, as

$$\inf \{NCS(C) : C \text{ is a well-formed instruction sequence of } M \text{ and } \pi_\Sigma(C) = x\}.$$

The (worst-case) nondeterminism degree of M is a function $N_M : \mathcal{N} \to \mathcal{N}$ that is defined by

$$N_M(n) = \max \{ND(x) : |x| = n \text{ and } x \in \Sigma^*\}.$$

Salomaa and Yu [8] called it the minmax nondeterminism measure.

Example 5.1: Let $M = (Q, \Sigma, \Gamma, \delta, s, F, Z)$ be a pushdown machine, where $Q = \{s, p, f\}$, $\Sigma = \{a, b\}$, $\Gamma = \{Z, A, B\}$, $F = \{f\}$, and δ consists of the following instructions:

1. $(s, CY^{-1}YX, s)$, for all $(C, X) \in \{(a, A), (b, B)\}$ and for all $Y \in \{Z, A, B\}$.
2. $(s, Y^{-1}Y, p)$, for all $Y \in \{Z, A, B\}$.
3. (p, CX^{-1}, p), for all $(C, X) \in \{(a, A), (b, B)\}$.
4. (p, Z^{-1}, f).

Obviously, M accepts the language $\{ww^R : w \in \Sigma^*\}$. It is straightforward to verify that the nondeterminism degree of M is $O(n)$.

We say that a context-free language has **nondeterminism-degree complexity** N if there is a pushdown machine M such that $L = L(M)$ and the nondeterminism degree of M is N. The two context-free languages

$$L = \{a^i b^j a^k | i = j \text{ or } j = k, i, j, k > 0\}$$

and

$$L^k, \text{ for an integer constant } k \geq 0$$

have constant nondeterminism-degree complexity.

Salomaa and Yu [8] conjectured that there is an infinite hierarchy of nondeterminism-degree complexity classes of context-free languages. The only result, however, has been the demonstration that there is a context-free language with nondeterminism-degree complexity $\omega(1)$. We now use the transformation of pumping lemmas for context-free languages into pumping lemmas for pushdown machines to improve this result. We show that there is a context-free language that has nondeterminism-degree complexity $\Omega(\log n)$.

THEOREM 5.1: *Let* $K = \{ab^{i_1}\, ab^{i_2}\, a \ldots ab^{i_t} : t \geqq 2$ *and* $i_t = i_j$ *for some* $j,\ 1 \leqq j < t\}$. *The nondeterminism-degree complexity of* K *is* $\Omega(\log n)$.

Proof: Let M be a pushdown machine that accepts K. Without loss of generality we assume that each nondeterministic step of a computation has exactly two choices. Since $WFL(M)$ is a context-free language, it satisfies the positional pumping lemma (Lemma 3.2) for context-free languages. Let p be the constant for $WFL(M)$ in the pumping lemma and let $m > p$ be a constant. Consider the following m strings:

$$w_k = ab^{m+1}\, ab^{m+2}\, a \ldots ab^{2m}\, ab^{m+k}, \qquad 1 \leqq k \leqq m.$$

It is clear that $w_k \in K$, $1 \leqq k \leqq m$. Choose w_i and w_j such that $i < j$, and let C_i and C_j be two well-formed instruction sequences of M for the acceptance of w_i and w_j, respectively. It is easy to see that $C_i \neq C_j$ since $w_i \neq w_j$. We now prove by contradiction that C_i is not a prefix of C_j.

We assume that C_i is a prefix of C_j and we apply the pumping lemma to C_j. We choose the first p appearances of b in the last group of bs as the distinguished positions in C_j for the pumping lemma. It is easy to verify that in a decomposition $C_j = uvxyz$ given by the pumping lemma, y must contain d of the p distinguished positions, for some $1 \leqq d \leqq p$, and v necessarily contains d appearances of bs from the jth group of bs (and no other bs). When stating that y (or v) contains an appearance of the symbol b in a specific group of bs, we mean that in the decomposition $w_j = \pi_\Sigma(u)\,\pi_\Sigma(v)\,\pi_\Sigma(x)\,\pi_\Sigma(y)\,\pi_\Sigma(z)$ the substring $\pi_\Sigma(y)$ (or $\pi_\Sigma(v)$) contains an appearance of the symbol b in this group.

Now $C_i = uvxyz'$, where z' is a proper prefix of z. Since z' ends with a final state, it follows that

$$uv^r\, xy^r\, z' = C_i[r] \in WFL(M),$$

for all $r \geqq 0$. Now,

$$\begin{aligned}&\pi_\Sigma (C_i [m+1]) \\ &\quad = ab^{m+1} \cdots ab^{m+j-1}\, ab^{m+j+md}\, ab^{m+j+1} \cdots ab^{2m}\, ab^{m+i+md}\end{aligned}$$

and since $i \neq j$, $\pi_\Sigma (C_i [m+1]) \notin K$. We have obtained a contradiction.

Thus the two instruction sequences of M on w_i and w_j differ at some point earlier than final instructions. Not that this statement holds for any i and j, $1 \leqq i, j \leqq m$, such that $i \neq j$. Therefore, we can consider the instruction sequences as a binary tree with the start state as the root and nondeterministic steps as branches. Since there are m frontier nodes, there is one path from the root to the frontier that passes through at least log m branch points. In other words, there is necessarily an i, $1 \leqq i \leqq m$, such that the computation of w_i takes at least log m nondeterministic steps. Let n be the length of w_i. Because $n \simeq m^2$, log m is $\Omega (\log n)$ and we have shown that the nondeterminism-degree complexity of K is $\Omega (\log n)$. □

6. CONCLUSIONS

We have combined context-free pumping lemmas and the Floydian view of pushdown machines to synthesize machine-specific pumping lemmas. Based on this approach we have obtained elegant and short proofs that two well-known context-free languages are not deterministic. Our experience is that these proofs are the simplest to understand. In the second edition of his text [11], the second author uses this approach. We believe that the approach shows great promise from both the pedagogic and research points of view. To demonstrate the second claim, we have established the first nontrivial lower bound for nondeterminism-degree complexity of a context-free language. We anticipate that the application of the Floyd-pumping synthesis will provide further insight and clarification of the conjectured nondeterminism-degree hierarchy.

REFERENCES

1. R. W. Floyd and R. Beigel, *The Language of Machines: An Introduction to Computability and Formal Language Theory*, Computer Science Press, San Francisco, CA, 1994.
2. J. Goldstine, Automata with data storage, In *Proceedings of the Conference on Theoretical Computer Science*, Waterloo, Canada, 1977, University of Waterloo, pp. 239-246.

3. J. Goldstine, A rational theory of AFLs, In Proceedings of the Sixth Colloquium on Automata, Languages, and Programming, Volume 71 of *Lecture Notes in Computer Science*, New York, NY, 1979, Springer-Verlag, pp. 271-281.
4. J. Goldstine, Formal languages and their relation to automata: What Hopcroft & Ullman didn't tell us, In R.V. Book, Editor, *Formal Language Theory: Perspectives and Open Problems*, New York, NY, 1980, Academic Press, pp. 109-140.
5. M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, Reading, MA, 1978.
6. R. Kurki-Suonio, *Describing automata in terms of languages associated with their peripheral devices*, Technical Report STAN-CS-75-493, Computer Science Department, Stanford University, Stanford, CA, 1975.
7. K. Salomaa and S. Yu, Degrees of nondeterminism for pushdown automata, In Proceedings of the 8th Fundamentals of Computation Theory Conference, Volume 529 of *Lecture Notes in Computer Science*, New York, NY, 1991. Springer-Verlag, pp. 380-389.
8. K. Salomaa and S. Yu, Limited nondeterminism for pushdown automata, *Bulletin of the European Association for Theoretical Computer Science*, 1993, *50*, pp. 186-193.
9. K. Salomaa and S. Yu, Measures of nondeterminism for pushdown automata, *Journal of Computer and System Sciences*, to appear.
10. D. Vermeir and W. Savitch, On the amount of nondeterminism in pushdown automata, *Fundamenta Informaticae*, 1981, *4*, pp. 401-418.
11. D. Wood, *Theory of Computation*, John Wiley & Sons, Inc., New York, NY, second edition, 1993, In preparation.

Note added in proof: What we have called the Floydian view of pushdown machines should probably be called the Chomsky-Schützenberger-Floyd view. Chomsky and Schützenberger used inverse pushdown symbols in an early formulation of pushdown machines (see, for example, N. Chomsky, Context-free grammars and pushdown storage, *MIT Quarterly Progress Report*, 1962, *65*, pp. 187-194).