Regularity and Related Problems for Deterministic Pushdown Automata

LESLIE G. VALIANT

Carnegie-Mellon University, Pittsburgh, Pennsylvania

ABSTRACT. It is shown that to decide whether the language accepted by an arbitrary deterministic pushdown automaton is LL(k), or whether it is accepted by some one-counter or finite-turn pushdown machine, must be at least as difficult as to decide whether it is regular. The regularity problem itself is analyzed in detail, and Stearns' decision procedure for this is improved by one level of exponentiation. Upper bounds, close to known lower bounds, are obtained for the succinctness with which a pushdown automaton, and various restrictions of it, can express equivalent finite-state machines

KEY WORDS AND PHRASES. pushdown automaton, deterministic, decision procedure, complexity, regularity, program schema

CR CATEGORIES. 5 22, 5.24

1. *Introduction*

Several important restricted classes of deterministic pushdown acceptors (dpda) have desirable properties that are not enjoyed by the class as a whole. For example, finite-state automata recognize languages with all the ideal closure properties, one-counter automata have a certain rigid periodic structure [12], while in simple machines [5] configurations can be related in an elegant algebraic manner. Furthermore, within each of these subclasses, as well as others like the finite-turn machines [11] or the LL(k) acceptors [8], we know how to decide equivalence.

For these reasons we may prefer to work within one of these subclasses whenever possible, and would like to be able to decide for an arbitrary dpda whether an equivalent machine belonging to one of these exists. (membership problem)

These questions can each be put in the form of a *containment problem* (U:V). This will denote the problem of deciding for an arbitrary machine in class U whether there exists a machine in class V recognizing the same language.

The decidability of each of the containment problems (dpda : V), where V is one of the subclasses so far mentioned, is currently open, except for the case of finite-state automata, for which a decision procedure of triple exponential time complexity has been given by Stearns [9]. In this paper we first show that if, as we conjecture, these other problems are decidable, then they must be at least as difficult to decide as regularity. We do this by exhibiting efficient reductions of the latter problem to each of the others. We then reformulate and improve the regularity analysis of [9], and obtain an exponentially faster algorithm. In the course of this we resolve the following problem. Stearns has shown that the function defined as the size of the largest reduced finite automaton

This work was supported by the Science Research Council of the United Kingdom.

Author's present address Centre for Computer Studies, The University of Leeds, Leeds LS2 9JT, United Kingdom.

equivalent to some dpda with given parameters of description, is bounded above by a triple exponential function. Meyer and Fischer [7] have given an example to show that there is a double exponential lower bound to this function. Our result is to reduce the upper bound to a double exponential level.

By a well-known translation the decidability of regularity leads immediately to a procedure for testing whether for an arbitrary one-location recursion program schema there exists an equivalent Ianov schema. As an application of the improved analysis it is shown in [10] that a nearly achievable upper bound can be derived, on the succinctness with which large flowchart schemas can be reexpressed as smaller recursive ones. This is a formal result that relates to our intuitive knowledge about the considerable brevity that can sometimes be gained by recursive notation.

2. *Definitions*

Our definition of a deterministic pushdown automaton can be regarded as a normal form into which a machine in any of the other customary formulations can be easily translated [10].

A dpda M is a one-tape one-way deterministic acceptor with a pushdown stack and a finite-state control for storage. It can be specified by a sextuple $(Q,\Gamma,\Sigma,\Delta,c_s,F)$ where Q, Γ, Σ are respectively finite sets of *states* $(s, \cdots)$, *stack symbols* $\{A, \cdots\}$, and *input symbols* $\{a, \cdots\}$, and Δ, c_s, F, are as defined below. Typically we denote words from Γ^* and Σ^* by w and α respectively, and their lengths by $|w|$, $|\alpha|$. A *configuration* c is a pair (s, w) from $Q \times (\Gamma^+ \cup \Omega)$, where Ω is a special empty stack symbol. The *height* of c is $|w|$. A *mode*, designated either a *reading mode* or an ϵ-*mode*, is a pair from $Q \times (\Gamma \cup \Omega)$. Δ is a set of *transitions*, each of the form $(s, A) \xrightarrow{\pi} (s', w)$, where $\pi \in \Sigma \cup \{\epsilon\}$, such that (i) if (s, A) is a reading mode then for each $a \in \Sigma$, it has a unique transition with $\pi = a$, but none with $\pi = \epsilon$, and (ii) if (s, A) is an ϵ-mode then it has just one transition, and in this $\pi = \epsilon$.

The automaton M makes a *move* $(s, wA) \xrightarrow{\pi} (s', ww')$ if and only if there is some transition $(s, A) \xrightarrow{\pi} (s', w')$. If $\pi \in \Sigma$ then this symbol is considered to have been read.

A *derivation* $(c \rightarrow c')$ is a sequence of such moves through successive configurations starting from c and finishing at c'. It is said to read the word α if α is the concatenation of the symbols read by the constituent moves. It is then denoted by $c \xrightarrow{\alpha} c'$. There are no infinitely long ϵ-derivations, i.e. every input can be read in a finite number of moves. This assumption is necessary only in Section 4.

The set F of *accepting modes* is a set of reading modes. A word α is *accepted* from configuration c if and only if for some c' with mode (i.e. state, top stack symbol) belonging to F, there is a derivation from c to c' that reads α. Configurations c, c' are *distinguished* by α if α is accepted from one but not from the other. c and c' are *equivalent* $(c \equiv c')$ if no string distinguishes them.

There is a unique *starting configuration* c_s which is assumed, for simplicity, to be of height one. A configuration c is *reachable* in M if there is some derivation $c_s \rightarrow c$. The *language* accepted by M is the set of words accepted from c_s, and is denoted by $L(M)$. The class of languages recognized by a class U of automata we denote by $\mathfrak{L}(U)$, e.g. when *Fsa* denotes the class of finite state machines, $\mathfrak{L}(Fsa)$ will be the class of regular sets.

The size of M is described by means of the parameters q (number of states), t (number of stack symbols), p (number of input symbols), and h (maximum length of stack words appearing in the transition rules). The total machine description is bounded above by the product of these parameters.

M is described as *realtime* if it has no ϵ-moves. The symbols \$, $\$_1$ will denote input symbols which do not belong to the Σ being discussed.

For classifying the size of a function $X(x_1, \cdots, x_m)$ the following notation is useful. X is of *order* E^n if n is the largest integer such that

$$\overline{\lim_{y \to \infty}} [\log^{n+1} X(y, \cdots, y)/\log y] > 0.$$

It is of order $E^n(Y(x_1, \cdots, x_m))$ iff $\exists\ k, k' > 0$ such that for all sufficiently large $x_1, \cdots, x_m$, $\exp^n(kY) < X < \exp^n(k'Y)$. (Here $\exp^0(x) = x$, $\exp^n(x) = 2^{\exp^{n-1}(x)}$ for $n > 0$, and $\log^n$ is defined similarly.)

3. *Relative Complexity*

We shall define an ordering on decision problems. $P_1 \geq P_2$ will indicate that it is at least as difficult to decide problem P_1 as it is to decide P_2. This notion of relative complexity will attempt to embrace all the usual measures such as time and space, and to be machine-independent for most purposes, except when applied to very easy problems. Thus, for example, if some measure for P_2 for some reasonable machine model is expressed as a multinomial in the parameters of the tested automata, the corresponding measure for P_1 will have dominant terms at least as great, except possibly for multiplicative constant factors.

Let V be a class of machines, the parameters of each member of which can be described by a vector $\mathbf{x}$.

Definition. A transformation $\tau : V \to V$ is *efficient* if it can be carried out by a multitape Turing machine in linear time, and increases each parameter at most linearly.

Definition. $P_1 \geq P_2$ for V iff there is an efficient transformation τ such that P_2 can be decided for $M \in V$ by applying P_1 to $\tau(M)$.

In our applications to dpda, all the transformations we need are very simple transductions, provided that the descriptions are in a suitable lexicographic order, and we shall omit giving proofs of their efficiency. Furthermore, it will be the case that the representation of the outputs can be made the same as that of the inputs and therefore that the ordering $\geq$ will be transitive. The important point is that to decide any global property of these automata, the operations necessary are at least as difficult as these transductions and will therefore not be dominated by them. We also observe that any dpda specified in the more customary formulation of [1] can be translated into our normal form, with only linear growth in each parameter, by an algorithm that requires only linear time on a random access computer [13].

In Section 4 we shall denote the class of dpda by D, and prove results of the form $(D : U) \geq (D : V)$. From the definitions above this will clearly mean that if, in a given time, $(D : U)$ can be decided for any dpda with parameters less than $\mathbf{x}$, then for some $k > 0$, $(D : V)$ can also be decided in about the same time for any dpda with parameters less than $k\mathbf{x}$.

4. *Reductions*

The first theorem is for a class of containment problems. It will imply that it is at least as difficult to decide whether an arbitrary deterministic *language* (specified by a dpda) is simple (Korenjak and Hopcroft [5]), LL(k) (Lewis and Stearns [6]), or real-time strict (Harrison and Havel [4]) as it is to determine whether one is regular. We shall require the following observations.

Definition. $L \subset \Sigma^*$ has the *prefix property* if and only if $\alpha \in L$, $\alpha\beta \in L \Rightarrow \beta = \epsilon$.

Definition. For $L \subset \Sigma^*$, $\text{sinit}(L) = \Sigma^* - L\Sigma^+$.

LEMMA 1. *If L has the prefix property, then L regular $\Leftrightarrow$ sinit(L) regular.*

PROOF. ($\Rightarrow$) Immediate, since sinit(L) is defined by regularity preserving operations.

($\Leftarrow$) If L has the prefix property then it consists of just those words in sinit(L) that are not proper prefixes of other words in sinit(L). A finite-state automaton for L can therefore be obtained from one for sinit(L) by simply redefining the accepting states. The new accepting states are just the old ones, but with those states from which further old accepting states can be reached, removed. □

THEOREM 1. *For any class $U \subset D$ such that (a) U is realtime, (b) acceptance only occurs in empty stack configurations, and (c) $L\{\$\} \in \mathcal{L}(U)$ for any regular set L, $(D : U) \geq (D : Fsa)$.*

PROOF. We first specify a translation that modifies each $M \in D$ to an $M' \in D$ such that $L(M') = \text{sinit}(L(M)\{\$_1\})$. The state set of M' will be $Q \cup \{s_a, s_r, s_b\}$, where Q is the state set of M, and the accepting modes of M' will be just the modes that don't have state s_r. The transformation scans the transitions of M, and for each mode (s, A) of M it adds the transition $(s, A) \xrightarrow{\$_1} (s_a, A)$ or $(s, A) \xrightarrow{\$_1} (s_b, A)$ according to whether (s, A) is an accepting mode of M or not. For each $A \in \Gamma \cup \Omega$ and $a \in \Sigma$ it also adds the transitions $(s_a, A) \xrightarrow{a} (s_r, A)$, $(s_b, A) \xrightarrow{a} (s_b, A)$, and $(s_r, A) \xrightarrow{a} (s_r, A)$.

It can be easily verified that if the transitions and the set of accepting modes are stored in some lexicographic order on separate tapes of a multitape Turing machine, then this transformation can be carried out in linear time, and increases the parameters at most linearly.

A second efficient translation from M' to $M'' \in D$, such that $L(M'') = L(M')\{\$\}$, can be specified in a similar fashion. Furthermore, the two translations carried out consecutively can be regarded as a single efficient transformation.

Clearly if $L(M)$ is regular, then so is $L(M')$, and therefore $L(M'') \in \mathcal{L}(U)$ by assumption (c).

If $L(M)$ is not regular then, since $L(M)\{\$\}$ has the prefix property, by Lemma 1 $L(M')$ is not regular either. Any recognizer for it therefore needs to undergo unbounded stack growth. But from the definition of sinit, every word that is a prefix of some word in $L(M')$ is also in $L(M')$. Therefore in a recognizer for $L(M'')$ we require acceptance to occur on \$ input from configurations of unbounded heights. Therefore $L(M'')$ cannot satisfy both conditions (a) and (b).

We conclude that $L(M)$ can be tested for regularity by transforming M to M'', and testing $L(M'')$ for inclusion in U. □

The class of deterministic one-counter automata [12], the restriction of D to the case $t = 1$, we denote by C.

THEOREM 2. $(D : C) \geq (D : Fsa)$

PROOF. We specify an efficient transformation that maps each $M \in D$ to an $M' \in D$ where $L(M') = \bigcup_{n=1}^{\infty} [L(M)\{\$\}]^n \$_1^n$. M' will simulate M directly between successive occurrences of \$, and keep count of the number of \$ symbols already read. The count is in the form of a string A^n kept at the bottom of the stack, where A is a new symbol. When an accepting mode of M is reached in the simulation and a \$ symbol follows, the stack of M' is emptied down to the A^n segment, a new A is added, and the starting configuration of M is restored on the top of this. The last stage of the computation is to check the $\$_1$'s against the A's.

That this transformation is efficient can be easily verified. It remains to prove that $L(M') \in \mathcal{L}(C)$ iff $L(M)$ is regular.

Clearly if $L(M)$ is regular, then using a finite-state recognizer for it, a one-counter machine can be constructed to recognize $L(M')$, in the above-described manner. Conversely, suppose $L(M)$ is not regular. Then to recognize $L(M)$ starting from any configuration of a one-counter automaton, this automaton will have to empty its stack in the course of recognizing at least some of the input strings. Otherwise regularity would be implied. However an acceptor for $L(M')$ cannot be allowed to empty its stack between every pair of successive \$ symbols in the input, for then it cannot uniquely remember the number of strings from $L(M)\{\$\}$ already scanned. □

A pushdown acceptor is described as finite-turn [2] if in the set of all derivations from the starting configuration, there is a bound on the number of times the direction of the stack movement can change. Even in the nondeterministic case, a machine can be tested for the finite-turn property in polynomial time [2, 10]. Here we consider the containment problem for the deterministic finite-turn class T [11].

THEOREM 3. $(D : T) \geq (D : Fsa)$.

PROOF. It can be easily verified that there is an efficient transformation mapping an arbitrary $M \in D$ to an M' recognizing the language $[L(M)\{\$\}]^*$.

If $L(M)$ is regular, then $L(M')$ can be recognized by a finite-state automaton, which is, trivially, a finite-turn dpda.

Conversely, suppose $L(M)$ is not regular, and is recognized from some configuration of a dpda. Clearly a turn has to be made in the course of reading some input string, for otherwise regularity would be implied. It follows that when a string from $[L(M)\ \{\$\}]^*$ has been read by a recognizer for $L(M')$, for some continuation of the input string a turn will have to be made before the next \$ is read. For at this stage the recognizer could, with only minor changes, be modified to accept just $L(M)\ \{\$\}$. It follows that to recognize some string from $[L(M)\ \{\$\}]^n$, at least n turns will have to be made, and therefore that $L(M')$ cannot belong to $\mathfrak{L}(T)$. □

We note that many variations on the above results can also be obtained. In particular we observe that the above problems, as well as the problem of equivalence, remain equally difficult if D is replaced by the class of dpda accepting by empty stack. The efficient transformation that effectively adds an endmarker to the language recognized, ensures this in each case. It follows that by imposing such grammatical restrictions on the deterministic languages as the strict deterministic [3] or the LR(0) conditions, we are not making any of these problems substantially more tractable.

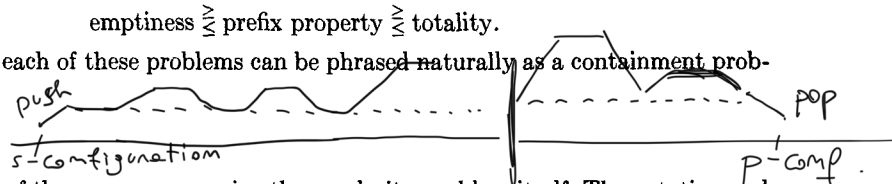
In conclusion we mention that many of the important decision problems that are known to be easy to decide for the whole class D, can be related to each other directly by means of appropriate reductions. In [10] it is proved that

$$\text{regularity} \geq \text{finiteness} \geq \text{emptiness}$$

and that

$$\text{emptiness} \gtrless \text{prefix property} \gtrless \text{totality}.$$

We observe that each of these problems can be phrased naturally as a containment problem.



5. *Derivations*

In the remainder of the paper we examine the regularity problem itself. The notation and basic ideas we describe are similar to, and derived from, those of Stearns [9]. There are however several modifications that enable more precise results to be obtained, and contribute to a different style of exposition.

We first introduce some terminology for describing the geometry of stack movements. All the following definitions are assumed to be with respect to some particular derivation $c \xrightarrow{\alpha} c'$.

Definitions. c_1 is a *stacking (s-) configuration* in the derivation iff it is not followed by any configuration of height $\leq |c_1|$. It is a *popping (p-)configuration* iff it is not preceded by any configuration of height $\leq |c_1|$. A derivation is a *stacking derivation* $(c \uparrow (\alpha)\ c')$ iff c is an s-configuration in it, a *popping derivation* $(c \downarrow (\alpha)\ c')$ iff c' is a p-configuration in it. An *index set* N is an ordered sequence of nonnegative integers. It induces *segments* $w_{i,j}$ in a stack w, where $w_{i,j}$ is the substring of w from the $(i+1)$-st letter to the jth, and $i, j \in N$. It also induces s- or p-configurations in a derivation, namely those of heights $i+1$, and i respectively, if defined, for each $i \in N$.

By definition, in any derivation there can be at most one s-configuration, and one p-configuration, of any given height. If $i < j$ then the string $\alpha_{i,j}$ will denote that substring of α read in the part of the derivation from the s-configuration of height $i+1$ to the one of height $j+1$. Notice that if these two have the same mode, then by iterating the substring $\alpha_{i,j}$ in the input k times, we obtain configurations like c', but with the stack segment $w_{i,j}$ iterated k times in the stack. If $i < j$ then $\alpha_{j,i}$ will denote that substring of α read from the p-configuration of height j to the one of height i. Where defined, it will be convenient to say that in the α derivation "the string $\alpha_{j,i}$ *pops* the segment $w_{i,j}$."

Definition. $c \downarrow (\alpha)\ c'$ is a *j-derivation* with respect to N iff fewer than j segments of w induced by pairs of integers consecutive in N, are popped by nonnull substrings of α.

Definition. The segment w' is *l-invisible* in $(s, ww'w'')$ with respect to N iff for any s' and any l-derivation $(s, ww'w'') \downarrow (\alpha) (s', ww')$, it is the case that $(s', ww') \downarrow (\epsilon) (s', w)$.

In other words, the existence of the segment w' can only be detected in the configurations by derivations that pop at least l disjoint segments of w'' induced by N, by nonnull input strings.

Definition. w is *null-transparent* iff for all $s \in Q$,

$$(s, w) \downarrow (\epsilon) (s', \Omega) \Rightarrow (s', w) \downarrow (\epsilon) (s', \Omega).$$

A null-transparent segment w therefore has the property that if $(s, w) \downarrow (\epsilon) (s', \Omega)$ then for all $m \geq 1$, $(s, w^m) \downarrow (\epsilon) (s', \Omega)$. Thus if one copy of w in a stack w_1w^m is ϵ-popped, then so are all the rest, and the final state reached will be independent of m. Since any input string α, a substring of which pops w^m in (s, w_1w^m), will have to ϵ-pop at least one of the copies of w if $|\alpha| < m$, we obtain the following.

LEMMA 2. *If w is null-transparent then for all s and w_1, (s, w_1w^m) and $(s, w_1w^{m'})$ cannot be distinguished by any string α where $|\alpha| < min(m, m')$.* □

6. *Main Lemma*

Using inductive arguments we now derive some sufficient conditions for the existence of null-transparent and l-invisible segments.

LEMMA 3. *For a configuration c with stack w and index set N of $\bar{N}$ elements all less than $|w|$,*

(i) $\bar{N} > q! \Rightarrow$ *some induced segment is null-transparent.*

(ii) $\bar{N} > 2(lq)^q \Rightarrow$ *some induced segment is l-invisible in c.*

PROOF. We prove both parts by means of the inductive assertion $\mathfrak{A}(P_m, N_m)$, where $P_m \subset Q$, $N_m \subset N$ and $m = 0, 1, 2 \cdots$. $\mathfrak{A}(P_m, N_m)$ is defined to hold if and only if

$$i, j \in N_m,\ s \in P_m \Rightarrow (s, w_{i,j}) \downarrow (\epsilon) (s, \Omega).$$

We define P_0 to be empty, and N_0 to be N. $\mathfrak{A}(P_0, N_0)$ will then trivially hold.

We shall show that if $\bar{N}_m$ is sufficiently large, and $\mathfrak{A}(P_m, N_m)$ holds, then *either* N_m already induces the required segment, *or* else we can find P_{m+1}, N_{m-1} such that $\mathfrak{A}(P_{m+1}, N_{m+1})$ also holds, where $P_m \nsubseteq P_{m+1}$ and $\bar{N}_{m+1}$ is greater than some given function of $\bar{N}_m$. From this it follows that if N is initially large enough, the inductive process can continue for up to q steps, i.e. until P_m exhausts Q, and hence that the required segment will be found.

PROOF OF (i). We assume that (P_m, N_m) is true, and that e and f, the smallest and largest elements of N_m respectively, are distinct. If $w_{e,f}$ is null-transparent the result is proved. Otherwise, by definition, there exist s, s' such that (a) $(s, w_{e,f}) \downarrow (\epsilon) (s', \Omega)$, but (b) not $(s', w_{e,f}) \downarrow (\epsilon) (s', \Omega)$.

Consider the p-configurations induced in derivation (a) by N_m. Clearly no state occurring in one of these configurations can belong to P_m, for, by the inductive assertion, it would have to be the state s', which is impossible since $s' \in P_m$ would contradict (b). Let s'' be one of the states occurring most frequently in these configurations, and let N_{m+1} index these occurrences. If $P_{m+1} = P_m \cup \{s''\}$ then $P_m \nsubseteq P_{m+1}$. Furthermore, by induction, P_m will be of size m, and therefore $\bar{N}_{m+1} \geq \bar{N}_m/(q - m)$. By construction $\mathfrak{A}(P_{m+1}, N_{m+1})$ clearly holds. Also, if $\bar{N} = \bar{N}_0 > q!$, the induction can continue until $m = q$, without $\bar{N}_m$ becoming less than two.

PROOF OF (ii). We assume that $\mathfrak{A}(P_m, N_m)$ is true and that e and f, now the smallest and second smallest elements of N_m respectively, are distinct. If $w_{e,f}$ is l-invisible in c the result is proved. Otherwise, by definition, there is some α-derivation that is an l-derivation with respect to N_m, rendering it visible, i.e. (a) $c \downarrow (\alpha) (s', w_{0,f})$, but (b) not $(s', w_{0,f}) \downarrow (\epsilon) (s', w_{0,e})$.

From the derivation (a) extract the ϵ-subderivation that pops the most segments in-

duced by N_m, and let N'_{m+1} be the set inducing these segments. Then $\bar{N}'_{m+1} \geq (\bar{N}_m - 1)/l$. None of the p-configurations induced in (a) by N'_{m+1} can possess a state belonging to P_m, for that would contradict (b). Let s'' be the most frequently occurring state and let N_{m+1} index these occurrences. If $P_{m+1} = P_m \cup \{s''\}$, then, as before, $P_m \nsubseteq P_{m+1}$, $\bar{N}_{m+1} \geq \bar{N}'_{m+1}/(q - m)$ and $\mathfrak{A}(P_{m+1}, N_{m+1})$ will hold. Now $\bar{N} = \bar{N}_0 > 2(lq)^2$ will guarantee that the induction can continue, if necessary, until Q is exhausted. □

Note 1. The bound in (i) is obtained in [9] by the same argument. It can be shown to be optimal by looking at segments that are ϵ-popped from all states, and thereby perform permutation operations on them. Taking representatives $g_1, g_2, \cdots, g_n$ of each of the $q! - 1$ different nonidentity permutations, the stack segment consisting of the concatenation of $g_1, g_1^{-1}g_2, g_2^{-1}g_3, \cdots, g_{n-1}^{-1}g_n$ has the required properties. Furthermore it can be realized with a stack alphabet of just two symbols.

Note 2. In the regularity theorem we shall be interested in l-invisibility when l is exponential in q. It is here that we gain our most significant improvement over [9], by having a bound of order $(lq)^q$ instead of one of q^{l+q}. We can show that if l is of larger order than q, then our bound is optimal in the following sense. A dpda can be derived from the proof of Lemma 3(ii), with the property that for some configuration and index set of size $(l/q)^q$, no l-invisible segment can be found.

7. *The Regularity Theorem*

Using Lemma 3, we now show how Stearns' construction can be improved by an exponential in the general case. There are also significant improvements for various important special cases, that yield bounds which are provably optimal to within multiplicative constants in the exponent.

THEOREM 4. *If M is a dpda in normal form with q states, t stack symbols, and stack words of length at most h in the transition rules, and if $L(M)$ is regular, then $L(M)$ is recognized by some finite-state automaton with fewer than $X(q, t, h)$ states, where X is a double exponential (E^2) function.*

PROOF. We shall prove that there is a function $Y(q, t, h)$ of order E^1 such that, if any reachable configuration of M has height greater than Y, then *either* we can cut out a segment from its stack to obtain a smaller equivalent reachable configuration, *or else* there exist input strings δ_1, δ_2 such that the configurations reached after inputs of $\delta_1\delta_2^m$ for $m = 1, 2, \cdots$ are all pairwise inequivalent. Thus if $L(M)$ is regular the first possibility must always hold for reachable configurations larger than Y. Consequently M can only be using up to qt^Y pairwise distinguishable configurations in recognizing $L(M)$. This gives the required result.

To prove the existence of Y we consider an arbitrary derivation $c_s \xrightarrow{\alpha} c = (s, w)$, where $|w| = n > Y(q, t, h)$. We let N be the set of integers indexing one of the most frequently occurring modes among the s-configurations defined by this derivation. Clearly $\bar{N} \geq n/qth$. If this is large enough then, by Lemma 3(ii), we can find a segment $w_{i,j}$ in w that is $(qq!)$-invisible in c w.r.t. N. Furthermore, by the choice of N, the configuration $c' = (s, w_{0,i}w_{j,n})$ is reachable from c_s via $\alpha_{0,i}\alpha_{j,n}$. We shall prove that if $L(M)$ is regular, then $c \equiv c'$. For simplicity, we shall not modify the indexing of the stack in the translation from c to c' (e.g. as in Figure 1, we shall still index the top of c' by n).

Suppose $c \not\equiv c'$, and let β be a shortest string distinguishing them. Then by the construction of c', for some γ, η such that $\beta = \gamma\eta$, $c \downarrow (\gamma)\ (s', w_{0,j})$, where this is not a $(qq!)$-derivation. It follows that we can pick an $N' \subset N$ of more than $q!$ integers between j and n with the properties that (a) no segment of w induced by N' is popped by an ϵ subderivation of γ, and (b) the elements of N' all index p-configurations with the same state.

We now pick a null-transparent segment $w_{k,m}$ induced by N' in $w_{j,n}$, as guaranteed by Lemma 3(i). We introduce new configurations c_1, c_2, c_1', c_2' such that:

$$c \downarrow (\gamma_{n,m})\ c_2,\ c_2 \downarrow (\gamma_{m,k})\ c_1,\ c' \downarrow (\gamma_{n,m})\ c_2',\ c_2' \downarrow (\gamma_{m,k})\ c_1'.$$

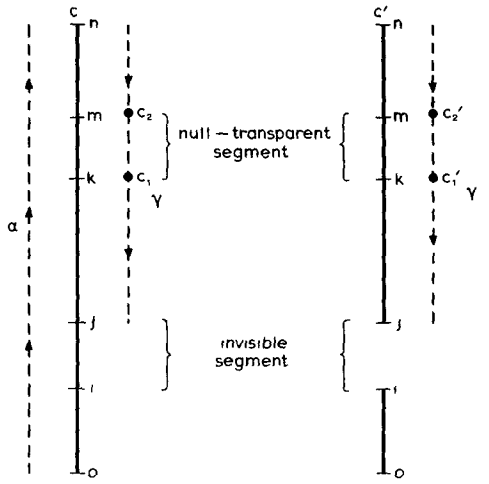


FIG. 1 Main construction

Since $\gamma\eta$ is a minimal distinguishing string for c, c', $\gamma_{m,j}\eta$ must be a minimal one for c_2, c_2', and $\gamma_{k,j}\eta$ for c_1, c_1'. But property (a) of N' ensures that $\gamma_{m,k} \neq \epsilon$, and therefore that $|\gamma_{k,j}| \neq |\gamma_{m,j}|$. Hence it is impossible that both $c_1 \equiv c_2$ and $c_1' \equiv c_2'$.

Without loss of generality we assume that $c_1 \not\equiv c_2$. We introduce a family of configurations $\{{}^r c \mid r \geq 0\}$ such that $c_s \uparrow (\alpha_{0,k}\alpha_{k,m}^r)\, {}^r c$. By the choice of N, N' the top parts of the stacks of these will consist of iterations of the null-transparent word. If we let $\xi_r = \alpha_{m,n}\gamma_{n,m}\gamma_{m,k}^r$ for $r \geq 0$, then ${}^{r+1}c \xrightarrow{\xi_r} c_2$ and ${}^r c \xrightarrow{\xi_r} c_1$ for $r \geq 0$.

Thus $c_1 \not\equiv c_2 \Rightarrow {}^r c \not\equiv {}^{r+1}c$ for any r. From this, and the null-transparency of $w_{k,m}$, we can now deduce that $r \geq 0$, $l > 0 \Rightarrow {}^r c \not\equiv {}^{l+r}c$.

For if we assume otherwise for some l, r, and consider the effect on the configurations ${}^{xl+r}c$ of inputs $\alpha_{k,m}^l$ and $\alpha_{k,m}$ for successive values of x of 0, 1, 2 $\cdots$, then we are led to deduce that for all $x \geq 0$, ${}^{xl+r}c \equiv {}^r c$ and ${}^{xl+r+1}c \equiv {}^{r+1}c$.

However, by Lemma 2, any string distinguishing ${}^r c$ and ${}^{r+1}c$ will fail to distinguish ${}^{xl+r}c$ and ${}^{xl+r+1}c$ for sufficiently large x, contrary to the previous statement.

To summarize, the assumption $c \not\equiv c'$ has forced us to the conclusion that the family $\{{}^r c\}$ (or a corresponding one constructed from c') consists of reachable but pairwise inequivalent configurations. Thus for regularity it must be that $c \equiv c'$, which is the result we want.

From Lemma 3 we recall that for the above construction it is sufficient that $\bar{N} > 2(q^2q!)^q$, and therefore that $Y(q, t, h) > 2qth(q^2q!)^q$. It follows that $X(q, t, h)$ can be bounded above by a function of order $E^2(q^2 \log q + \log t + \log h)$. □

COROLLARY 1. *If* $q = 1$, $X \leq E^1(ht \log t)$.

COROLLARY 2. *If there are no* ϵ*-moves then the bounds of Lemma* 3 *are* 1 *and* $l + 1$ *respectively, and hence* $X \leq E^1(hq^2t \log t)$.

COROLLARY 3. *For one-counter machines (i.e. with* $t = 1$, $h = 2$*),* $X \leq E^1(q \log q)^4$.

COROLLARY 4. *If all* ϵ*-moves leave states unchanged, then the bounds in Lemma* 2 *are* 1 *and* $2\, l^{\min(q,t)}$ *respectively. This gives* $X \leq E^2(t \log q + \log h)$.

Examples to show that the bounds of Corollaries 1, 2, and 3 can be achieved to within multiplicative constants in the exponents are given in [10]. That a double exponential order can be achieved by some dpda has been shown by Meyer and Fischer [7]. Their example, which also satisfies the conditions of Corollary 4, has $q = t$ and $h = 2$, and achieves order $E^2(q)$.

Corollary 3 is outlined in [10], and obtained independently in [12]. We note that the structure of the configurations $\{{}^r c\}$ in Theorem 4 is very similar to the notion of propriety described in [12].

The proof of Corollary 4 rests on a revision of Lemma 3(ii), and is implicit in [10, Ch. 8], where it is applied to program schemas. From this corollary an almost achievable upper bound can be derived for the succinctness with which large Ianov schemas can be reexpressed as equivalent small recursive ones [10].

8. *Complexity of Regularity Test*

From the above analysis we can deduce that for an arbitrary dpda M, the regularity of $L(M)$ can be tested in double exponential time. The decision procedure is to construct the candidate finite-state automaton, say M', specified in the proof of the regularity theorem, and to test whether it is equivalent to the given dpda. Since this last test requires only polynomial time in terms of the machine sizes, and since M' is of E^2 size, it only remains to prove that M' can be constructed in E^2 time.

The states of M' are to correspond to configurations of M of height no more than Y. To construct M' from M we introduce new transitions to replace all the old moves which took the stack above the Y level. To determine these we need to compute, for each configuration of height just greater than Y, the $(qq!)$-invisible segments specified in the proof of Theorem 4. It is easily verified that this can be done in E^2 time altogether, by first finding for each configuration a derivation reaching it, and computing the transition table of the popping derivations for each of the induced segments.

For the realtime and one-counter cases, we get procedures of single exponential time complexity in a similar manner. If in the realtime case we have the additional restriction that acceptance can only occur in empty stack configurations then, as observed for various cases elsewhere [4, 5], we can get a polynomial time test. It is easily verified that such a machine accepts a regular language if and only if no accepting derivation of it goes through a configuration of height greater than hq^2t. This property, however, can be easily decided by testing the machine, in appropriately augmented form, for emptiness.

9. *Conclusion*

By imposing various restrictions on deterministic pushdown automata we obtain machines with such diverse characteristics as the finite-state machines, one-counter automata, and the finite-turn and simple deterministic pushdown automata. These all have the major advantage that their structure is much better understood than that of the unrestricted machines.

We have investigated the problem of deciding whether for an arbitrary dpda there exists an equivalent one within one of these classes. We have improved Stearns' regularity test exponentially to one of double exponential space and time complexity. In the process we have obtained a near optimal upper bound on the succinctness with which a dpda can represent an equivalent large finite automaton.

The other problems we have shown must be at least as difficult to decide as regularity. Since we know of no nontrivial lower bound for the complexity of any of these, we interpret the force of these results, at this time, to be the warning that it may be difficult to find decision procedures for these open problems. In spite of this, we suggest that their pursuit may be worthwhile in serving as a focus for the further investigation of the structure of deterministic pushdown automata.

REFERENCES

1. GINSBURG, S , AND GREIBACH, S A. Deterministic context-free languages. *Inform and Contr 9* (1966), 620–668
2. GINSBURG, S., AND SPANIER, E Finite-turn pushdown automata *SIAM J. Control 4* (1966), 423–434
3. HARRISON, M. A , AND HAVEL, I M. Strict deterministic grammars. *J. Comput Syst Sci 7* (1973) 237–277.
4. HARRISON, M A., AND HAVEL, I. M Real time strict deterministic languages. *SIAM J Computing 1* (1972) 333–349

5 KORENJAK, A. J , AND HOPCROFT, J E Simple deterministic languages. IEEE 7th Symp. on Switching and Automata Theory, Berkeley, Calif., 1966, pp. 36–46.
6. LEWIS, P. M II, AND STEARNS, R. E. Syntax-directed transduction. *J. ACM 15*, 3 (July 1968), 465–488
7. MEYER, A. R , AND FISCHER, M J Economy of description by automata, grammars, and formal systems IEEE 12th Symp on Switching and Automata Theory, 1971, 188–191.
8. ROSENKRANTZ, D J , AND STEARNS, R. E. Properties of deterministic topdown grammars. *Inform. and Contr 17* (1970) 226–255
9. STEARNS, R. E. A regularity test for pushdown machines. *Inform and Cont. 11* (1967), 323–340
10. VALIANT, L. G Decision procedures for families of deterministic pushdown automata. Ph D. Th , Computer Centre Rep No. 7, U of Warwick, Coventry, England, 1973
11. VALIANT, L. G. The equivalence problem for deterministic finite-turn pushdown automata *Inform. and Contr 25* (1974), 123–133
12. VALIANT, L G., AND PATERSON, M S. Deterministic one-counter automata. *J. Comput. Syst. Sci.* (to appear).
13 VALIANT, L. G. Manuscript.

RECEIVED JANUARY 1974; REVISED MAY 1974

2-EXP? lower-bound on the size of a DFA equivalent to a DPDA recognizing a regular language.

Can use a stack of height k to count in binary $0, \ldots, 2^k - 1$.

Thus one can build a DPDA recognizing $L = \{a^{2^k}\}$ of size k. (real-time)