Math. Systems Theory 15, 315–321 (1982)

Mathematical
Systems Theory

On Reducing the Number of States in a PDA[1]

Jonathan Goldstine,* John K. Price,*[2] and Detlef Wotschke**

*Computer Science Department, The Pennsylvania State University, University Park, Pennsylvania, U.S.A.

**Fachbereich Informatik, Johann Wolfgang Goethe-Universität, Frankfurt, West Germany

Abstract. A transformation is presented which converts any pushdown automaton (PDA) M_0 with n_0 states and p_0 stack symbols into an equivalent PDA M with n states and $\lceil n_0/n \rceil^2 p_0 + 1$ stack symbols for any desired value of n, $1 \leqslant n < n_0$. This transformation preserves realtime behavior but not determinism. The transformation is proved to be the best possible one in the following sense: for each choice of the parameters n_0 and p_0, there is a realtime PDA M_0 such that any equivalent PDA M (whether realtime or not) having n states must have at least $\lceil (n_0/n)^2 p_0 \rceil$ stack symbols. Furthermore, the loss of deterministic behavior cannot be avoided, since for each choice of n_0 and p_0, there is a deterministic PDA M_0 such that no equivalent PDA M with fewer states can be deterministic.

I. Introduction

It is well known that there is an algorithm to minimize the number of states in a deterministic finite automaton. Since each move of a pushdown automaton (PDA) depends on the top stack symbol as well as on the state and the input symbol, the product of the number of states and the number of stack symbols in a PDA is in a sense analogous to the number of states in a finite automaton, and it might be desirable to minimize this product. However, an argument by Gruska [6] can be modified to show that there is no algorithm to accomplish this. On the other hand, minimizing the number of states in a PDA is trivial: since each PDA is equivalent to a context-free grammar (CFG), it easily follows that each PDA is equivalent to a one-state PDA. (We assume that acceptance is by empty stack, so

[1]This research was supported in part by the National Science Foundation under Grants MCS76-10076 and MCS76-10076A01.

[2]*Current address*: Bell Laboratories, Holmdel, New Jersey, U.S.A.

0025-5661/82/0015-0315 $01.40

that a separate accepting state is unnecessary.) This construction replaces stack symbols by (state, stack symbol, state)-triples, and hence multiplies the number of stack symbols by approximately n_0^2, where n_0 is the original number of states. Thus, although one cannot effectively minimize the state-stack symbol-product for a PDA, one can collapse the finite-state control to a single state, but only at a large cost in the size of the stack alphabet. This suggests the question, can one reduce the size of the finite-state control less drastically at a lesser cost?

The present paper shows that a natural generalization of the "triple" construction for converting a PDA to a CFG will reduce the number of states in any PDA from n_0 to n for any desired n, $1 \leqslant n < n_0$, at the cost of increasing the size of the stack alphabet from p_0 to $\lceil n_0/n \rceil^2 p_0 + 1$. Like the triple construction, it preserves realtime behavior but not determinism. Furthermore, this construction is more or less the best possible, in the sense that an expansion in the stack alphabet to size $\lceil (n_0/n)^2 p_0 \rceil$ is sometimes unavoidable even if realtime behavior need not be preserved, and also in the sense that no general state-reduction procedure can preserve determinism.

Section II describes this generalized triple construction. Section III shows that the construction cannot be improved.

II. The Generalized Triple Construction

We adopt the usual notation $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ for a PDA (see, e.g., [8]), with the following modifications:

1. Since M accepts by empty stack, we omit F.

2. To facilitate comparisons between PDAs and CFGs, we shall write the move

$$(q, B_1 B_2 \ldots B_m) \in \delta(p, x, A)$$

in the form

$$pA \to xqB_1 B_2 \ldots B_m. \qquad (*)$$

Here, p and q are states in Q; x is in $\Sigma \cup \{e\}$ (e is the empty string); $A, B_1, B_2, \ldots, B_m$ are stack symbols in Γ; $m \geqslant 0$; and Q, Σ, Γ are assumed to be pairwise disjoint.

Thus, $pA \to xqB_1 \ldots B_m$ denotes the move that changes state from p to q while consuming x from the input stream, and popping A and pushing $B_1 \ldots B_m$ on the stack. (B_1 is the new top stack symbol if $m \geqslant 1$.) Then M is *realtime* if $pA \to xq\beta$, $\beta \in \Gamma^*$, implies $x \neq e$. And M is *deterministic* if, for each $p \in Q$, $A \in \Gamma$, $a \in \Sigma$, the set of moves

$$\{pA \to aq\beta \mid q \in Q, \beta \in \Gamma^*\} \cup \{pA \to q\beta \mid q \in Q, \beta \in \Gamma^*\}$$

contains at most one move of M.

Note that the language $L(M)$ defined by M by empty stack may be obtained by interpreting the moves (*) as context-sensitive productions, adding the productions

$$q \to e, q \in Q,$$

and taking $q_0 Z_0$ as the axiom for a grammar:

$$L(M) = \{w \in \Sigma^* | q_0 Z_0 \overset{*}{\Rightarrow} w\}.$$

Obviously, if M has only one state, then the states can be omitted from (*), resulting in context-free productions which generate $L(M)$ from Z_0. So a one-state PDA is essentially identical to a CFG, and leftmost derivations of the CFG correspond to computations of the PDA. From this point of view, the usual "triple" construction [3] for converting a PDA to a CFG is merely a construction for reducing the number of states in a PDA to a single state * as follows.

Triple Construction

Replace each move

$$pA \to xp_0 B_1 \dots B_m, \qquad m \geqslant 0,$$

by the $(\#Q)^m$ moves

$$*\langle pAp_m \rangle \to x*\langle p_0 B_1 p_1 \rangle \dots \langle p_{m-1} B_m p_m \rangle, (p_1, \dots, p_m) \in Q^m;$$

and if $pA = q_0 Z_0$, then also add the moves

$$*Z_0 \to x*\langle p_0 B_1 p_1 \rangle \dots \langle p_{m-1} B_m p_m \rangle, (p_1, \dots, p_m) \in Q^m.$$

The new set of stack symbols consists of the starting symbol Z_0 and $\#(Q \times \Gamma \times Q)$ new symbols $\langle pAq \rangle$, one for each triple in $Q \times \Gamma \times Q$. It is a routine matter to verify by induction on the length of the derivations that

$$q_0 Z_0 \overset{+}{\Rightarrow} wp_0 B_1 \dots B_m$$

for the first PDA if and only if there exists $(p_1, \dots, p_m) \in Q^m$ such that

$$*Z_0 \overset{+}{\Rightarrow} w*\langle p_0 B_1 p_1 \rangle \dots \langle p_{m-1} B_m p_m \rangle$$

for the second PDA, so that both PDAs accept the same language.

This construction can be generalized as follows. Suppose that $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ is a PDA, and suppose that $Q \subseteq Q' \times Q''$ for some sets Q' and Q'' having n and k elements respectively. Let $q_0 = (q_0', q_0'') \in Q' \times Q''$.

Generalized Triple Construction

Replace each move

$$(p', p'')A \to x(p_0', p_0'')B_1 \dots B_m, \qquad m \geqslant 0,$$

by the k^m moves

$$p'\langle p''Ap_m''\rangle \to xp_0'\langle p_0''B_1p_1''\rangle \dots \langle p_{m-1}''B_m p_m''\rangle, (p_1'', \dots, p_m'') \in (Q'')^m,$$

and if $(p', p'')A = (q_0', q_0'')Z_0$ then also add the moves

$$q_0'Z_0 \to xp_0'\langle p_0''B_1p_1''\rangle \dots \langle p_{m-1}''B_m p_m''\rangle, (p_1'', \dots, p_m'') \in (Q'')^m.$$

Thus, when $m = 0$, the popping move $(p', p'')A \to x(p_0', p_0'')$ is replaced by the popping move $p'\langle p''Ap_0''\rangle \to xp_0'$. The new set of states is Q' with initial state q_0', and the new set of stack symbols is

$$\Gamma' = \{Z_0\} \cup \{\langle p''Aq''\rangle | p'', q'' \in Q'', A \in \Gamma\}$$

with initial stack symbol Z_0. Note that the Generalized Triple Construction is essentially the Triple Construction when $n = 1$ and produces essentially no change when $k = 1$. Also, note that the construction transforms realtime PDAs to realtime PDAs, but in general transforms deterministic PDAs to PDAs that are not deterministic.

Lemma. *If* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ *is a* PDA *with* $Q \subseteq Q' \times Q''$ *and* $M' = (Q', \Sigma, \Gamma', \delta', q_0', Z_0)$ *is the* PDA *produced from* M *by the Generalized Triple Construction, then* M' *is equivalent to* M, *i.e.*, $L(M') = L(M)$.

Proof. Just as in the case of the Triple Construction, one may prove by induction on the length of derivations that

$$(q_0', q_0'')Z_0 \overset{+}{\Rightarrow} w(p_0', p_0'')B_1 \dots B_m \text{ in } M$$

if and only if there exists $(p_1'', \dots, p_m'') \in (Q'')^m$ such that

$$q_0'Z_0 \overset{+}{\Rightarrow} wp_0'\langle p_0''B_1p_1''\rangle \dots \langle p_{m-1}''B_m p_m''\rangle \text{ in } M'.$$

Taking $m = 0$, the lemma follows. □

Theorem 1. *For every* PDA M_0 *with* n_0 *states and* p_0 *stack symbols and for every integer* n *in the range* $1 \leqslant n < n_0$, *there is an equivalent PDA* M *with* n *states and* $\lceil n_0/n \rceil^2 p_0 + 1$ *stack symbols such that* M *is realtime if* M_0 *is.*

Proof. Let Q' be a set of size n and Q'' be a set of size $k = \lceil n_0/n \rceil$. Then $Q' \times Q''$ has size $nk \geqslant n_0$, so the state set of M_0 can be identified with a subset of

$Q' \times Q''$. The Generalized Triple Construction now produces a PDA M satisfying the theorem. In particular, M has n states and $kp_0k + 1 = \lceil n_0/n \rceil^2 p_0 + 1$ stack symbols. □

III. Can the Generalized Triple Construction be Improved?

The Generalized Triple Construction reduces the number of states in a PDA from n_0 to n at the cost of increasing the size of the stack alphabet from p_0 to $\lceil n_0/n \rceil^2 p_0 + 1$. It preserves realtime behavior but not determinism. There are two questions one might raise in seeking to improve this construction.

—Is there is a general construction to lower the number of states of a PDA that increases the stack alphabet to a size substantially smaller than $\lceil n_0/n \rceil^2 p_0 + 1$, perhaps at the cost of destroying realtime behavior?

—Is there a general construction to lower the number of states of a PDA while preserving nondeterminism, perhaps at the cost of a larger increase in the size of the stack alphabet?

The answer to both questions is no, as the two theorems in this section show.

Theorem 2. *For every pair of positive integers n_0 and p_0, there is a* PDA M_0 *with n_0 states and p_0 stack symbols such that every equivalent* PDA M *with n states has at least $\lceil (n_0/n)^2 p_0 \rceil$ stack symbols. Furthermore, the PDA M_0 is realtime and deterministic.*

Proof. In [5], it is shown that, for every pair of positive integers n_0 and p_0, there is a realtime deterministic PDA M_0 with n_0 states and p_0 stack symbols such that every equivalent CFG G has at least $n_0^2 p_0$ self-embedding variables. (The variable A is self-embedding if $A \overset{*}{\Rightarrow} uAv$, $uv \neq e$, for some terminal strings u and v.) Let M be a PDA with n states and p stack symbols that is equivalent to M_0. The Triple Construction converts M to an equivalent CFG G with at most $n^2 p$ self-embedding variables, since the one additional starting symbol is not self embedding. Hence, $n^2 p \geqslant n_0^2 p_0$, so $p \geqslant (n_0/n)^2 p_0$, and the theorem follows. □

Lemmas 11.6.1 and 11.6.3 in Harrison [7] prove that, for each $n_0 \geqslant 1$, there is a deterministic PDA with n_0 states and five stack symbols such that every equivalent deterministic PDA has at least as many states. The next theorem extends this result to deterministic PDAs with just one stack symbol.

Theorem 3. *For every pair of positive integers n_0 and p_0, there is a deterministic PDA M_0 with n_0 states and p_0 stack symbols such that every PDA M equivalent to M_0 having fewer than n_0 states is not deterministic. Furthermore, the PDA M_0 is realtime.*

Proof. It obviously suffices to prove the theorem for the case in which $p_0 = 1$, for one can then add additional, useless stack symbols to M_0 to increase p_0 to any desired value. Thus, what is needed is a language that can be recognized by a deterministic "one-counter" automaton M_0 (i.e., a deterministic PDA with one stack symbol) with n_0 states, but not by any deterministic PDA with fewer states.

Intuitively, to a first approximation such a language is

$$L_0 = \{a^m c_i b_i^m | m \geqslant 0, 0 \leqslant i < n_0\},$$

since, on encountering c_i, a deterministic finite-state control must remember i in order to insure that the remaining input symbols are b_i's. However, for this choice of L_0, one additional state would be needed to enforce the formatting restriction that every string lie in

$$R = a^* \cdot \{c_i | 0 \leqslant i < n_0\} \cdot \{b_i | 0 \leqslant i < n_0\}^*.$$

So in order to avoid this complication, the formatting restriction will be relaxed slightly, and the machine M_0 defined below will also accept some strings not in R, such as ab_0b_0 or ac_0c_1.

For any n_0, let $M_0 = (Q_0, \Sigma, \{Z_0\}, \delta_0, q_0, Z_0)$, where

$$Q_0 = \{q_i | 0 \leqslant i < n_0\},$$
$$\Sigma = \{a, b_i, c_i | 0 \leqslant i < n_0\},$$

and δ_0 consists of the moves

$$q_0 Z_0 \to a q_0 Z_0 Z_0,$$
$$q_0 Z_0 \to c_i q_i, 0 \leqslant i < n_0,$$
$$q_i Z_0 \to b_i q_i, 0 \leqslant i < n_0.$$

Then M_0 is a realtime deterministic PDA having n_0 states and one stack symbol, so it suffices to prove that any equivalent deterministic PDA has at least n_0 states. Let $M = (Q, \Sigma, \Gamma, \delta, p_0, Z_0)$ be such a PDA. For each $m \geqslant 0$ and k, $0 \leqslant k < n_0$, $a^m c_k b_k^m \in L(M_0) = L(M)$, so there is a computation in M of the form

$$p_0 Z_0 \overset{*}{\Rightarrow} a^m p_m Z_m \alpha_m \Rightarrow a^m c_k p_{mk} \beta_{mk} \alpha_m, p_{mk} \beta_{mk} \overset{*}{\Rightarrow} b_k^{r_{mk}} p'_{mk}, p'_{mk} \alpha_m \overset{*}{\Rightarrow} b_k^{s_{mk}}, \tag{*}$$

where $r_{mk} + s_{mk} = m$, for some $p_m, p_{mk}, p'_{mk} \in Q$, $Z_m \in \Gamma$, α_m, $\beta_{mk} \in \Gamma^*$, r_{mk}, $s_{mk} \geqslant 0$. Note that $p_m Z_m \alpha_m$ is determined by a^m since M is deterministic and the following move consumes an input symbol. Whenever two (m, k) pairs (m_1, k_1) and (m_2, k_2) have the same value (p, Z, p') for (p_m, Z_m, p'_{mk}),

$$p_0 Z_0 \overset{*}{\Rightarrow} a^{m_1} p Z \alpha_{m_1} \Rightarrow a^{m_1} c_{k_2} p_{m_2 k_2} \beta_{m_2 k_2} \alpha_{m_1}$$
$$\overset{*}{\Rightarrow} a^{m_1} c_{k_2} b_{k_2}^{r_{m_2 k_2}} p' \alpha_{m_1} \overset{*}{\Rightarrow} a^{m_1} c_{k_2} b_{k_2}^{r_{m_2 k_2}} b_{k_1}^{s_{m_1 k_1}} \tag{**}$$

If there were infinitely many (m, k) pairs for which $s_{mk} = 0$ then there would be two such pairs (m_1, k_1) and (m_2, k_2) with different m values having the same (p, Z, p') value. It would then follow from (**) that $m_1 = r_{m_2 k_2} + s_{m_1 k_1} = r_{m_2 k_2}$; while by (*), $m_2 = r_{m_2 k_2} + s_{m_2 k_2} = r_{m_2 k_2}$; so that $m_1 = m_2$, contrary to hypothesis.

Thus, for any large enough m, $s_{mk} > 0$ for all k, $0 \leqslant k < n_0$. For such a fixed m, suppose that two different k values have the same value p' for p'_{mk}. Since m is fixed, the values p and Z of p_m and Z_m also coincide, so by (**), $a^{m_1} c_{k_2} b'^{m_2 k_2}_{k_2} b^{s_{m_1 k_1}}_{k_1}$ is in $L(M) = L(M_0)$. Since $s_{m_1 k_1} > 0$, this is only possible if $k_1 = k_2$, contrary to hypothesis. Thus, the states p_{mk} must be distinct for each of the n_0 different values of k, and so M has at least n_0 states. □

As a final observation, note that the Generalized Triple Construction converts a PDA with n_0 states and p_0 stack symbols to a PDA with $n < n_0$ states and $\lceil n_0/n \rceil^2 p_0 + 1$ stack symbols, and Theorem 2 proves that at least $\lceil (n_0/n)^2 p_0 \rceil$ stack symbols are sometimes necessary. There is a slight gap between these two bounds. Thus, it remains an open question whether a very slight improvement in the Generalized Triple Construction is possible.

References

1. A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation, and Compiling, Vol. I: Parsing.* Prentice-Hall: Englewood Cliffs, 1972.
2. Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars, in Y. Bar-Hillel, *Language and Information.* Reading: Addison-Wesley, 116–150, 1964.
3. S. Ginsburg. *The Mathematical Theory of Context-Free Languages.* New York: McGraw-Hill, 1966.
4. S. Ginsburg and S. A. Greibach. Deterministic context-free languages, *Information and Control* 9, 563–582, 1966.
5. J. Goldstine, J. K. Price, and D. Wotschke. A pushdown automaton or a context-free grammar—which is more economical?, *Theoretical Computer Science* 18, 33–40, 1982.
6. J. Gruska. Complexity and unambiguity of context-free grammars and languages, *Information and Control 18*, 502–519, 1971.
7. M. A. Harrison. *Introduction to Formal Language Theory.* Reading: Addison-Wesley, 1978.
8. J. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Reading: Addison-Wesley, 1979.
9. M. O. Rabin and D. Scott. Finite automata and their decision problems, in E. F. Moore (ed.). *Sequential Machines: Selected Papers.* Reading: Addison-Wesley, 63–91, 1964.
10. M. P. Schutzenberger. On context-free languages and push-down automata, *Information and Control 6*, 246–264, 1963.

Received October 6, 1981, and in final form May 10, 1982.