THE APPROXIMATION OF PROBABILISTIC TURING AUTOMATA BY PROBABILISTIC PUSHDOWN AUTOMATA *

by

Clarence A. Ellis

Department of Computer Science
University of Colorado
Boulder, Colorado 80302

Report #CU-CS-020-73 JUNE, 1973

* This work was supported by NSF Grant GJ-660.

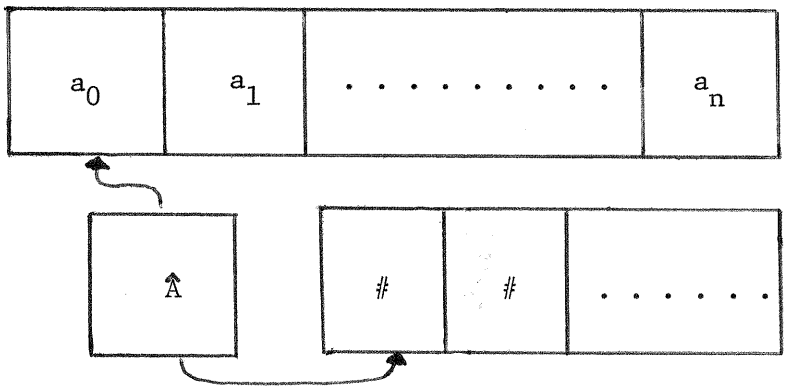
ABSTRACT

The concepts of Probabilistic Pushdown Automata and Probabilistic Turing Automata are defined. Then an algorithm is proposed which allows one to approximate the Turing Automaton by the Pushdown Automaton. The goodness of fit of this approximation is investigated, and an error bound is derived.

A Probabilistic Pushdown Storage Automaton (called a pushdown P automaton) consists of a finite state control unit, two one-way infinite tapes (an input tape and a storage tape), a read head for the input tape, and a read-write head for the storage tape. The machine operates by changing states at discrete time intervals. Along with this state change, it may read a symbol of the input tape, causing the tape to move left to the next symbol; and it may perform a stack operation (push or pop) on the storage tape. The new state and the string of symbols printed on the storage tape at time t + 1 depend probabilistically upon the old state and the symbols read on the two tapes at time t. This can be stated formally.

Definition: A Probabilistic Pushdown Storage Automaton $\hat{A}$ over an alphabet, T, of input symbols is a system (Q, M, S, Ξ) where Q is a finite set of states, $\{q_1, q_2 \ldots q_n\}$, S is a finite set of storage tape symbols, $\{s_1, s_2 \ldots s_m\}$, with $s_m = \#$, Ξ is an n-dimensional initial state vector, and M is a probabilistic transition function from $Q \times S \times (T \cup \{\lambda\})$ into $P(Q \times \overline{S}) \cup \{\lambda\}$ where $\overline{S} = S \cup \{\lambda, \sigma\}$.

A situation of $\hat{A}$ is a triple (q, s, a) with $q \in Q$, $s \in S$, $a \in T$. $\hat{A}$ is started in the situation $(q_0, \#, a_0)$ where $\xi(q_0) > 0$, and a_0 is the first symbol (not including #) of some string $x \in T^+$ which is present on the input tape. Pictorially, $\hat{A}$ has the following initial configuration, where the # denotes a blank square of a tape.



Then, using the terminology of Haines[2], $(q', s') \in M(q, s, a)$ will be written as $(q, s, a) \overset{p}{\rightarrow} (q', s')$ where p is the probability associated with the transition. Only three types of instructions are allowed:

(1) $(q, s, a_i) \overset{p}{\rightarrow} (q', s')$. This means if the automaton is in the state q, and scanning symbols s and a_i on the storage and input tapes respectively, then with probability p, a transition into state q' will occur, and the storage and input tapes will move one square to the left and then $s' \in S - \{\#\}$ will be printed on the storage tape one square to the right of s.

(2) $s' = \lambda$ implies the storage tape is left unchanged and unmoved, so $(q', s) \in M(q, s, a_i)$. The next situation is (q', s, a_{i+1}) assuming $1 \leq i < n$ where the input string is $x = a_1, a_2 \ldots a_n$.

(3) $s' = \sigma$ implies the symbol s is erased from the scanned square of the storage tape and the tape is moved one square to the right. If the string written on the storage tape at time t before the transition was $s_0, s_1 \ldots s_k, s$, with all $s_i \in S$, then the string at time t+1 after the transition is $s_0, s_1, \ldots, s_k$. $(q', s_k) \in M(q, s, a_i)$ and the new situation is (q', s_k, a_{i+1}). This is defined only if $k \geq 0$.

If $a = \lambda$ in any of these instructions, then the input tape is left unmoved and the transition is independent of the input symbol scanned, $(q, s, \lambda) \overset{p}{\rightarrow} (q', s')$ implies $(q', s') \in M(q, s, a)$ for all $a \in T$ and the next situation is (q', s', a).

A pushdown P automaton $\hat{A}$ terminates in acceptance if it makes a lambda transition, $M(q, s, a_n) \rightarrow \lambda$ such that $s = \#$ or such that the storage tape is empty after deleting s during this final transition (where a_n denotes the last symbol of the input string x). It is convenient to write $(q_f, \#, \#)$ to denote the situation after the automaton has halted in acceptance although $\#$ and q_f are fictitious (i.e., $\# \notin T$ and $q_f \notin Q$). Also, $\hat{A}$ terminates in nonacceptance if it is in a

situation (q, s, a) such that $M(q, s, a) = \phi$ or if the read head goes past the last input symbol a_n or the read-write head attempts to move to the square before the first square of the storage tape. A transition sequence for $x \in T^+$ is a sequence of situations $(q_0, s_0, a_0), (q_1, s_1, a_1), \ldots, (q_n, s_n, a_n)$ where $\hat{A}$ is started in an initial configuration with xy written on the input tape for some string $y \in T^*$, and a sequence of elementary transitions consistent with M occur, $(q_i, s_i, a_i) \overset{p_{i+1}}{\rightarrow} (q_{i+1}, s_{i+1})$, where p_{i+1} is the probability of the transition. The situation after the sequence of transitions must be (q_n, s_n, a_n) where $a_n = y_1$, the first terminal symbol of y. If further, $y = \lambda$ and (q_n, s_n, a_n) is the final situation, $(q_t, \#, \#)$, then the sequence is called an <u>accepting transition sequence</u>. The probability of a transition sequence is the product of the probabilities of the elementary transitions, $\bar{p}_k = \prod_{i=1}^{n} p_i$. The <u>probability of occurrence of x</u> is $\mu^*(x) = \sum_{k=1}^{m} \xi(q_0)\bar{p}_k$ where m is the number of transition sequences for x. The <u>probability of acceptance of x</u> is $\mu(x) = \sum_{k=1}^{\ell} \xi(q_0)\bar{p}_k$ where ℓ is the number of accepting transition sequences for x. The probabilistic transition function M can be specified by a set of transition items $(q, s, a) \rightarrow (q', s')$ and a probability p associated with each transition item. The criterion which we impose upon automata is that the total probability of leaving any state must sum to 1, noting that other normalizations are possible and in fact may be desirable in other applications [1].

<u>Definition</u>: A probabilistic pushdown storage automaton (Q, M, S, Ξ) over T is <u>normalized</u> iff Ξ is a stochastic vector and

$$\sum_{s \in S} \sum_{s' \in Su\{\lambda,\sigma\}} \sum_{a \in Tu\{\lambda\}} \sum_{q' \in Q} p[(q,s,a) \rightarrow (q',s')] = 1 \; \forall \; q \in Q.$$

Example: A normalized pushdown P automaton for the language $\{a^n b^n | n \geq 0\}$.

$\hat{A} = (Q, M, S, \Xi)$ over $T = \{a,b\}$

(1) $Q = \{q_0, q_1, q_2\}$

(2) $S = \{s,\#\}$

(3) M has instructions:

$(q_0, \#, a) \overset{1}{\rightarrow} (q_1, s)$

$(q_1, s, a) \overset{1/2}{\rightarrow} (q_1, s)$

$(q_1, s, b) \overset{1/2}{\rightarrow} (q_2, \sigma)$

$(q_2, s, b) \overset{1/2}{\rightarrow} (q_2, \sigma)$

$(q_2, s, b) \overset{1/2}{\rightarrow} (q_t, \sigma)$, i.e., $(q_2, s, b) \overset{1/2}{\rightarrow} \lambda$.

(4) $\Xi = (1, 0, 0)$

The model of Turing Automaton defined herein is derived from the 1-tape online turing machine[5]. A Probabilistic Turing Automaton (called a Turing P automaton) is basically a pushdown P automaton in which the storage tape is allowed to move left and right without erasing. More formally,

Definition: A Probabilistic Turing Automaton $\hat{A}$ over T is a system (Q, M, S, Ξ) where Q is a finite set of states, S is a finite set of storage tape symbols, M is a probabilistic transition function, $M: Q \times S \times T \overset{p}{\rightarrow} \mathcal{P}(Q \times S \times J) \cup \{\lambda\}$ where $J = \{-1, 0, 1\}$ and Ξ is the initial distribution vector.

An instruction of $\hat{A}$ may be written $(q, s, a) \overset{p}{\rightarrow} (q', s', j)$ where $j = 1$ indicates move the tape one square to the left and then print s', $j = -1$ implies print s' and then move to the right, and $j = 0$ implies print s' but do not move the tape.

All other definitions and restrictions are the same as for pushdown P automata. This version of Turing automaton differs from the usual in that it is probabilistic, it contains a move-before-write type of instruction, its storage

THE APPROXIMATION OF PROBABILISTIC TURING AUTOMATA BY PROBABILISTIC PUSHDOWN AUTOMATA *

by

Clarence A. Ellis

Department of Computer Science
University of Colorado
Boulder, Colorado 80302

Report #CU-CS-020-73 JUNE, 1973

* This work was supported by NSF Grant GJ-660.

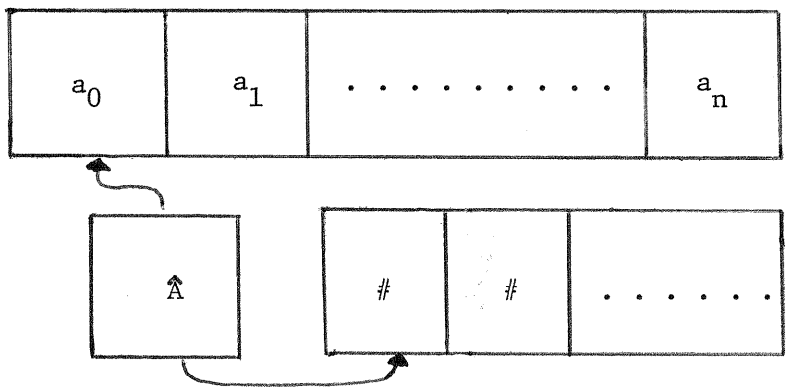
ABSTRACT

The concepts of Probabilistic Pushdown Automata and Probabilistic Turing Automata are defined. Then an algorithm is proposed which allows one to approximate the Turing Automaton by the Pushdown Automaton. The goodness of fit of this approximation is investigated, and an error bound is derived.

A Probabilistic Pushdown Storage Automaton (called a pushdown P automaton) consists of a finite state control unit, two one-way infinite tapes (an input tape and a storage tape), a read head for the input tape, and a read-write head for the storage tape. The machine operates by changing states at discrete time intervals. Along with this state change, it may read a symbol of the input tape, causing the tape to move left to the next symbol; and it may perform a stack operation (push or pop) on the storage tape. The new state and the string of symbols printed on the storage tape at time t + 1 depend probabilistically upon the old state and the symbols read on the two tapes at time t. This can be stated formally.

Definition: A Probabilistic Pushdown Storage Automaton $\hat{A}$ over an alphabet, T, of input symbols is a system (Q, M, S, Ξ) where Q is a finite set of states, $\{q_1, q_2 \ldots q_n\}$, S is a finite set of storage tape symbols, $\{s_1, s_2 \ldots s_m\}$, with $s_m = \#$, Ξ is an n-dimensional initial state vector, and M is a probabilistic transition function from $Q \times S \times (T \cup \{\lambda\})$ into $P(Q \times \overline{S}) \cup \{\lambda\}$ where $\overline{S} = S \cup \{\lambda, \sigma\}$.

A situation of $\hat{A}$ is a triple (q, s, a) with $q \in Q$, $s \in S$, $a \in T$. $\hat{A}$ is started in the situation $(q_0, \#, a_0)$ where $\xi(q_0) > 0$, and a_0 is the first symbol (not including #) of some string $x \in T^+$ which is present on the input tape. Pictorially, $\hat{A}$ has the following initial configuration, where the # denotes a blank square of a tape.



Then, using the terminology of Haines[2], $(q', s') \in M(q, s, a)$ will be written as $(q, s, a) \overset{p}{\rightarrow} (q', s')$ where p is the probability associated with the transition. Only three types of instructions are allowed:

(1) $(q, s, a_i) \overset{p}{\rightarrow} (q', s')$. This means if the automaton is in the state q, and scanning symbols s and a_i on the storage and input tapes respectively, then with probability p, a transition into state q' will occur, and the storage and input tapes will move one square to the left and then $s' \in S - \{\#\}$ will be printed on the storage tape one square to the right of s.

(2) $s' = \lambda$ implies the storage tape is left unchanged and unmoved, so $(q', s) \in M(q, s, a_i)$. The next situation is (q', s, a_{i+1}) assuming $1 \leq i < n$ where the input string is $x = a_1, a_2 \ldots a_n$.

(3) $s' = \sigma$ implies the symbol s is erased from the scanned square of the storage tape and the tape is moved one square to the right. If the string written on the storage tape at time t before the transition was $s_0, s_1 \ldots s_k, s$, with all $s_i \in S$, then the string at time t+1 after the transition is $s_0, s_1, \ldots, s_k$. $(q', s_k) \in M(q, s, a_i)$ and the new situation is (q', s_k, a_{i+1}). This is defined only if $k \geq 0$.

If $a = \lambda$ in any of these instructions, then the input tape is left unmoved and the transition is independent of the input symbol scanned, $(q, s, \lambda) \overset{p}{\rightarrow} (q', s')$ implies $(q', s') \in M(q, s, a)$ for all $a \in T$ and the next situation is (q', s', a).

A pushdown P automaton $\hat{A}$ terminates in acceptance if it makes a lambda transition, $M(q, s, a_n) \rightarrow \lambda$ such that $s = \#$ or such that the storage tape is empty after deleting s during this final transition (where a_n denotes the last symbol of the input string x). It is convenient to write $(q_f, \#, \#)$ to denote the situation after the automaton has halted in acceptance although # and q_f are fictitious (i.e., # T and $q_f \notin Q$). Also, $\hat{A}$ terminates in nonacceptance if it is in a

situation (q, s, a) such that $M(q, s, a) = \phi$ or if the read head goes past the last input symbol a_n or the read-write head attempts to move to the square before the first square of the storage tape. A transition sequence for $x \in T^+$ is a sequence of situations $(q_0, s_0, a_0), (q_1, s_1, a_1), \ldots, (q_n, s_n, a_n)$ where $\hat{A}$ is started in an initial configuration with xy written on the input tape for some string $y \in T^*$, and a sequence of elementary transitions consistent with M occur, $(q_i, s_i, a_i) \overset{p_{i+1}}{\rightarrow} (q_{i+1}, s_{i+1})$, where p_{i+1} is the probability of the transition. The situation after the sequence of transitions must be (q_n, s_n, a_n) where $a_n = y_1$, the first terminal symbol of y. If further, $y = \lambda$ and (q_n, s_n, a_n) is the final situation, $(q_t, \#, \#)$, then the sequence is called an <u>accepting transition sequence</u>. The probability of a transition sequence is the product of the probabilities of the elementary transitions, $\bar{p}_k = \prod_{i=1}^{n} p_i$. The <u>probability of occurrence of x</u> is $\mu^*(x) = \sum_{k=1}^{m} \xi(q_0)\bar{p}_k$ where m is the number of transition sequences for x. The <u>probability of acceptance of x</u> is $\mu(x) = \sum_{k=1}^{\ell} \xi(q_0)\bar{p}_k$ where ℓ is the number of accepting transition sequences for x. The probabilistic transition function M can be specified by a set of transition items $(q, s, a) \rightarrow (q', s')$ and a probability p associated with each transition item. The criterion which we impose upon automata is that the total probability of leaving any state must sum to 1, noting that other normalizations are possible and in fact may be desirable in other applications [1].

<u>Definition</u>: A probabilistic pushdown storage automaton (Q, M, S, Ξ) over T is <u>normalized</u> iff Ξ is a stochastic vector and

$$\sum_{s \in S} \sum_{s' \in Su\{\lambda,\sigma\}} \sum_{a \in Tu\{\lambda\}} \sum_{q' \in Q} p[(q,s,a) \rightarrow (q',s')] = 1 \quad \forall\, q \in Q.$$

Example: A normalized pushdown P automaton for the language $\{a^n b^n | n \geq 0\}$.

$\hat{A} = (Q, M, S, \Xi)$ over $T = \{a,b\}$

(1) $Q = \{q_0, q_1, q_2\}$

(2) $S = \{s,\#\}$

(3) M has instructions:

$(q_0, \#, a) \overset{1}{\rightarrow} (q_1, s)$

$(q_1, s, a) \overset{1/2}{\rightarrow} (q_1, s)$

$(q_1, s, b) \overset{1/2}{\rightarrow} (q_2, \sigma)$

$(q_2, s, b) \overset{1/2}{\rightarrow} (q_2, \sigma)$

$(q_2, s, b) \overset{1/2}{\rightarrow} (q_t, \sigma)$, i.e., $(q_2, s, b) \overset{1/2}{\rightarrow} \lambda$.

(4) $\Xi = (1, 0, 0)$

The model of Turing Automaton defined herein is derived from the 1-tape online turing machine[5]. A Probabilistic Turing Automaton (called a Turing P automaton) is basically a pushdown P automaton in which the storage tape is allowed to move left and right without erasing. More formally,

Definition: A Probabilistic Turing Automaton $\hat{A}$ over T is a system (Q, M, S, Ξ) where Q is a finite set of states, S is a finite set of storage tape symbols, M is a probabilistic transition function, $M: Q \times S \times T \overset{P}{\rightarrow} P(Q \times S \times J) \cup \{\lambda\}$ where $J = \{-1, 0, 1\}$ and Ξ is the initial distribution vector.

An instruction of $\hat{A}$ may be written $(q, s, a) \overset{P}{\rightarrow} (q', s', j)$ where $j = 1$ indicates move the tape one square to the left and then print s', $j = -1$ implies print s' and then move to the right, and $j = 0$ implies print s' but do not move the tape.

All other definitions and restrictions are the same as for pushdown P automata. This version of Turing automaton differs from the usual in that it is probabilistic, it contains a move-before-write type of instruction, its storage

tape is only infinite to the right, and it must terminate by scanning the # in the initial square for acceptance. It is easy to see that the latter three alterations do not change the set of languages accepted by a Turing automaton.

ALGORITHM: Given any Turing P automaton, $\hat{A}$, the following procedure yields a pushdown P automaton $\hat{A}'$, which accepts an approximation language $L(\hat{A}') = (T^+, \mu')$, to the language accepted by $\hat{A}$, $L(\hat{A}) = (T^+, \mu)$. Let $\hat{A} = (Q, M, S, \Xi)$ where we restrict Ξ to be 1 for one state (initial state q_0) and 0 for all others, then $\hat{A}' = (Q', M', S', \Xi')$. S' consists of all of S together with stack symbols ?s for all symbols s in S. M' can be described by a set of instructions:

(a) For each $(q, s, a) \overset{p}{\rightarrow} (q', s', +1)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q', s')$ and $(s, ?s, a) \overset{p}{\rightarrow} (q', s')$ into M'.

(b) For each $(q, s, a) \overset{p}{\rightarrow} (q', s', 0)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q_0, \sigma)$, $(q_0, \lambda, \lambda) \overset{1}{\rightarrow} (q', s')$, and $(q, ?s, a) \overset{p}{\rightarrow} (q_0, \sigma)$ into M', where $q_0 \in Q'$ is a new state which only appears in these three instructions.

(c) For each $(q, s, a) \overset{p}{\rightarrow} (q', s', -1)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q_1, \sigma)$, $(q_1, \lambda, \lambda) \overset{1}{\rightarrow} (q_2, \sigma)$, $(q_2, \lambda, \lambda) \overset{1}{\rightarrow} (q', s')$, and $(q, ?s, a) \overset{p}{\rightarrow} (q_1, \sigma)$ into P. Put new symbols q_1 and q_2 into Q'.

(d) For each $(q, s, a) \overset{p}{\rightarrow} (q', \lambda, -1)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q', \sigma)$ and $(q, ?s, a) \overset{p}{\rightarrow} (q', \sigma)$ into M.

(e) For each $(q, s, a) \overset{p}{\rightarrow} (q', \lambda, 0)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q', \lambda)$, $(q, ?s, a) \overset{p}{\rightarrow} (q', \lambda)$ into M.

(f) For each $(q, s, a) \overset{p}{\rightarrow} (q', \lambda, +1)$ in M, put $(q, s, a) \overset{p}{\rightarrow} (q', ?s)$ into M for all symbols ?s defined in S'.

The six instruction types cover all possible directions, with and without printing, that the Turing P automaton can move. Notice that the pushdown P automaton can simulate all of these transitions except moving to the right without printing, in which case it prints ?s. Thus, if type (f) instructions are not used in the Turing P automaton program, then the P language recognized will also be recognized by the pushdown P automaton constructed from the Turing P automaton by the previous algorithm. Furthermore, if the square in which ?s would get printed by the pushdown P automaton is never revisited by the Turing P automaton, then the approximation is again exact.

Definition: The (un-normalized) Initial Definite P Language, $\hat{L}_{ID}$ of a pushdown P automaton $\hat{A}'$ which approximates a Turing P automaton $\hat{A}$ is the set of all strings x which are maximal initial definite segments of strings $z \in T^* \ni \mu(z) > 0$. The probability assigned to x is the probability of partial acceptance with respect to $\hat{A}$, $\mu_{ID}(x) = \mu^*(x)$. All other elements y of T* have $\mu_{ID}(y) = 0$. x is maximal initial definite if it fulfills:

(1) initial: $\exists\, y \in T^* \ni xy = x,\ \mu(z) > 0$

(2) definite: $\exists$ a transition sequence for x with respect to $\hat{A}'$ which contains no $(a, q, ?s) \overset{P}{\rightarrow} (q', \beta')$ type instructions (called indefinite instructions).

(3) maximal: $\exists$ a transition sequence for x fulfilling (2) above which can be extended to an accepting transition sequence for z with respect to $\hat{A}'$ such that the first instruction after the initial definite segment x is indefinite, $(q, ?s, a) \rightarrow (q', s')$.

Define lid = $\sum_{x \in T^+} \mu_{ID}(x)$. Define $\hat{L}_{ID}$ state set as the set of states of Q' in which the pushdown P automaton can reside after strings $x \ni \mu_{ID}(x) > 0$ have been input.

THEOREM: Let $\hat{A}$ be a Turing P automaton. Let $\hat{A}'$ be the approximating push-down P automaton as described above. If the set of states of A' reachable from the L_{ID} state set all have type 4 normalization (where q reachable means there is a transition sequence such that $q_n = q$, and $q_0 \in L_{ID}$ state set) then the following error bound holds:

$$\sum_{z \in T^+} |\mu(z) - \mu'(z)| \leq \text{lid} - \sum_{z \in T^+} \mu(z)$$

Proof: Consider a Markov chain whose states are the states of the automaton $\hat{A}'$, and whose transition probabilities p_{ij} are just the probabilities p_{ij} of a transition from state q_i to q_j. Take as initial state any $q_i \in L_{ID}$ state set. By normalization,

$$\sum_{q_j \in Q' \cup \{q_t\}} p_{ij} = 1.$$

It is then possible to compare $\mu_{ID}(x)$ to $\mu'(z)$ for all strings $z \in L(\hat{A}') \ni z = xy$ for some string y. $\mu_{ID}(x) \geq \sum_{z=xy} \mu'(z)$. The inequality is not a strict equality because we have not excluded anomalies such as instructions which lead to deadend, nonaccepting states. Since all accepting transition sequences with respect to $\hat{A}'$ begin with transition sequences for some x with $\mu_{ID}(x) > 0$,

$$\sum_{x \in L_{ID}} \mu_{ID}(x) \geq \sum_{z \in T^+} \mu'(z).$$

$\text{lid} \geq \sum_{z \in T^+} \mu'(z)$, $\sum_{z \in T^+} |\mu(z) - \mu'(z)| = \sum_{z \in T^+} \mu'(z) - \mu(z)$ because each string of this is accepted under this approximation with probability $\geq \mu(z)$.

$$\sum_{z \in T^+} |\mu(z) - \mu'(z)| \leq \sum_{z \in T^+} \mu'(z) - \sum_{z \in T^+} \mu(z)$$

$$\leq \text{lid} - \sum_{z \in T^+} \mu(z)$$

$$\leq \text{lid} - 1 \text{ if } L(\hat{A}') \text{ is a normalized probabilistic language.}$$

Thus, if error of approximation is measured by $\sum_{z \in T^+} |\mu(z) - \mu'(z)|$, then an error bound is lid - 1.

CAE:cah

REFERENCES

[1] Ellis, C. A., Probabilistic Languages and Automata, University of Illinois Ph.D. Thesis, 1969.

[2] Haines, L. H., Generation and Recognition of Formal Languages, M.I.T. Ph.D. Thesis, 1965.

[3] Karlin, S., Stochastic Processes, Academic Press, 1969.

[4] Paz, A., Introduction to Probabilistic Automata, Academic Press, 1971.

[5] Rosenberg, A. L., "Real-Time Definable Languages," JACM 14, 4, pp. 645-662.