# PATH SYSTEMS AND LANGUAGE RECOGNITION*

Stephen A. Cook
University of California at Berkeley

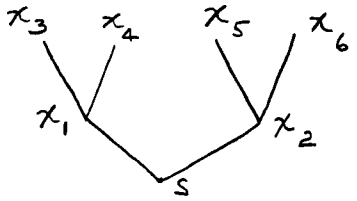
Summary.
Our main result, theorem 2, gives a bound on the storage required for a Turing machine to simulate certain time-bounded pushdown machines. The theorem is a generalization of the result appearing in [3] stating that any context-free language can be recognized by a deterministic Turing machine within storage $(\log n)^2$. We introduce a combinatorial object, called a path system, develop its theory briefly, and use the theory to prove both the result on pushdown machines and the result on context free languages, as well as a third result. The third result is the Theorem of Savitch [5] stating that a non-deterministic $L(n)$ - storage bounded Turing machine can be simulated by a deterministic $(L(n))^2$ - storage bounded Turing machine.

Path Systems.

Definition A path system is a quadruple $\mathcal{A} = \langle S,R,S,T\rangle$, where $X$ is a finite set (of nodes), $R$ is a three place relation on $X$ (the incidence relation), $S \subseteq X$ ($S$ is the set of source nodes) and $T \subseteq X$ ($T$ is the set of terminal nodes).

The admissible nodes of $\mathcal{A}$ are the least set $A$ such that $T \subseteq A$, and such that if $y, z \in A$ and $R(x,y,z)$, then $x \in A$. We say $\mathcal{A}$ is solvable if at least one admissible node is in $S$.

The notion of solvable can be defined alternatively as follows. Starting with a node $s \in S$, we try to form a directed graph by finding nodes $x_1,x_2,\in X$ so $R(s,x_1,x_2)$ holds. Then we let $s,x_1,x_2$ be vertices of the graph, and let $(s,x_1)$ and $(s,x_2)$ be directed edges. Then we find $x_3,x_4$ and $x_5,x_6$ so $R(x_1,x_3,x_4)$ and $R(x_2,x_5,x_6)$ hold, and let $(x_1,x_3)$, $(x_1,x_4)$, $(x_2,x_5)$, $(x_2,x_6)$ be directed edges as shown.




We continue in this way (possibly letting a node have two or more edges coming in) until every directed path leads to a terminal node (member of $T$). This is possible if and only if $\mathcal{A}$ is solvable.

More precisely, $\mathcal{A}$ is solvable if and only if there is a digraph $G = \langle V,E\rangle$ where

a) the set $V$ of vertices is a subset of $X$,
b) for each edge $(x,y) \in E$, there is $z \in V$ such that $(x,z) \in E$ and either $R(x,y,z)$ or $R(x,z,y)$ holds.
c) There are no directed loops in $G$ (i.e. no sequences $(x_1,x_2)$, $(x_2,x_3),\ldots,(x_{k-1},x_k),(x_k,x_1)$ of edges),
d) $V \cap S$ is non-empty,
e) Every vertex $x$ either has edges leading out from itself or $x \in T$.

Such a digraph $G$ is called solution graph for the path systems $\mathcal{A}$.

Def'n The path system $\mathcal{A}$ is tree - like if either $\mathcal{A}$ is unsolvable, or if some solution graph of $\mathcal{A}$ is a binary tree. (For our purposes, a digraph is an binary tree if and only if each vertex has at most one edge coming in and either two or zero edges coming out, and no directed loops).

We shall be concerned only with tree-like path systems here. However, more general path systems are of interest; for example the proofs of the results on auxiliary pushdown machines in [2] could be stated in terms of general path systems.

Example Let $G = \langle V, \Sigma, P, \sigma\rangle$ be a context-free grammar with terminals $V$, alphabet $\Sigma$, productions $P$, and initial symbol $\sigma$. We shall assume $G$ is in the normal form in which all production are of one of the forms

$$\xi_1 \to \xi_2\ \xi_3, \quad \xi \to A,$$

where $\xi_1$, $\xi_2$, $\xi_3$ and $\xi$ are nonterminals, and $A$ is a terminal.

For a non-empty string $w \in V^*$ we associate a tree like path system $\mathcal{A} = \langle X,R,S,T\rangle$ with $w$ and $G$ as follows (here $n$ is the length of $w$):

* Most of the material here appeared in a different form in the unpublished report [1]

X is the set of triples $\langle \xi, i, j \rangle$, where $\xi$ is a non terminal, and $i$ and $j$ are integers satisfying $1 \leq i \leq j \leq n$.

$R(x,y,z)$ holds if and only if $x,y,z$ have the form $\langle \xi_1, i, k \rangle$, $\langle \xi_2, i, j \rangle$, $\langle \xi_3, j+1, k \rangle$ respectively, where $\xi_1 \rightarrow \xi_2\ \xi_3$ is a production of $G$,

$S = \{\langle \sigma, 1, n \rangle\}$,

T is the set of all nodes of the form $\langle \xi, i, i \rangle$, where $\xi \rightarrow W_i$ is production of $G$, and $w = W_1 W_2 \ldots W_n$.

It is not hard to see that the admissible nodes of $\mathcal{A}$ are precisely those nodes $\langle \xi, i, j \rangle$ such that the string $W_i W_{i+1} \ldots W_j$ is derivable from $\xi$ in G. Thus $w = W_1 W_2 \ldots W_n$ is in the language generated by G if and only if $\langle \sigma, 1, n \rangle$ is admissible, i.e. if and only if $\mathcal{A}$ is solvable.

To see that $\mathcal{A}$ is tree - like, we note if $w$ is in the grammar generated by G, then a generation tree for $w$ tells us how to construct a solution digraph for $\mathcal{A}$ which is a binary tree. In fact, the generation tree, with final branches pruned, will be isomorphic to the solution digraph.

We shall use this example to show that context-free languages can be recognized within storage $(\log n)^2$.

Query Machines

Let $\Sigma$ be a finite alphabet, and suppose R is a finite r-place relation on $\Sigma^*$ (i.e. R is a finite set of r-tuples of strings on $\Sigma$). The norm $|R|$ of R is the length of the longest string in some r-tuple of R. That is,

$$|R| = \max \{\ell \mid \ell = \max_{i=1}^{r} |w_i|, \text{ for some } \langle w_1, \ldots, w_r \rangle \text{ such that } R(w_1, \ldots, w_r) \text{ holds}\}$$

A query machine (simular to the Turing machines M with oracle defined in [4]) is a (deterministic) Turing machine designed to recognize a class $\mathcal{R}$ of finite r-place relations on $\Sigma^*$. The query machine M is equipped with a special query tape, in addition to a work tape, and M has three distingushed states, called the yes state, no state, and query state.

A query machine is deterministic, except when in the query state it can continue in either the yes state or no state. If R is an r-place relation on $\Sigma^*$, then the R-computation of M is the computation M undergoes starting in the initial state with both tapes blank, such that whenever M enters the query state with a string of the form $w_1 * w_2 * \ldots * w_r$ on the query tape, $w_i \in \Sigma^*$, the next state assumed is the yes state if $R(w_1, \ldots, w_r)$ holds and the no state if $R(w_1, \ldots, w_r)$ fails to hold. If the string on the query tape is not of the form $w_1 * \ldots * w_r$ when M is in the query state, then M halts.

We say M accepts R within storage $\ell$ if the R-computation ends in an accepting state and no more than $\ell$ squares of the work tape are scanned. Suppose $\mathcal{R}$ is a set of finite r-place relations on $\Sigma^*$, and L(n) is a function from natural numbers to reals. We say M accepts R within storage L(n) if for all $R \in \mathcal{R}$, M accepts R within storage $L(|R|)$, and for all $R \notin \mathcal{R}$, M fails to accept R (in any storage). If some such machine M exists for the set $\mathcal{R}$, then we say $\mathcal{R}$ has tape complexity L(n).

A path system $\mathcal{A} = \langle X, R, S, T \rangle$ is a $\Sigma$-path system if the set X of nodes is a set of strings on $\Sigma$ excluding the empty string. The relation $R^*$ codes such a path system $\mathcal{A}$ if $R^* = R \cup \{\langle x, \Lambda, \Lambda \rangle \mid x \in S\} \cup \{\langle \Lambda, y, \Lambda \rangle \mid y \in T\}$, where $\Lambda$ is the empty string.

Theorem 1 Let $\Sigma$ be a finite alphabet. Then there is a set $\mathcal{R}$ of finite 3-place relations on $\Sigma^*$ such that

1) $\mathcal{R}$ includes all $R^*$ which code solvable tree-like $\Sigma$-path systems
2) $\mathcal{R}$ includes no $R^*$ which code unsolvable $\Sigma$ path systems
3) $\mathcal{R}$ has tape complexity $L(n) = n^2$

The proof is an abstraction of that outlined in [3] showing that every context free language has tape complexity $(\log n)^2$. In fact, that result follows from the present result.

Corollary 1 Every context-free language has tape complexity $L(n) = (\log n)^2$.

Proof. Let $G = \langle V, \Sigma, P, \sigma \rangle$ be a context-free language, and for each $w \in V^*$ let $\mathcal{A}_w$ be the path system associated with $w$ and G which we described earlier. Let $\Sigma' = \Sigma \cup \{0,1,/\}$. Then $\mathcal{A}_w$ can be translated into a tree-like $\Sigma'$-path system $\mathcal{A}'_w$ by representing

the node $\langle \xi,i,j \rangle$ by the string $\xi/T/J$, where $T$, $J$ are the binary notations for $i$ and $j$. Note that if $R_w^*$ codes $\mathcal{Q}'_w$, then $|R^*_w| \leq 2 \log_2 |w| - 3$. Thus a Turing machine $M$ can be constructed which incorporates the query machine of theorem 1. Given an input $w$, $M$ has the query machine perform an $R^*_w$-computation by answering the queries appropriately, and $M$ accepts $w$ if and only if the query machine accepts $R^*_w$. The storage required by $M$ is dominated by that required by the query machine, which, by theorem 1, is $(2 \log_2|w| + 3)^2$. This can be reduced to $(\log |w|)^2$, by standard results on Turing machines.

Corollary 2 (Savitch [5]) If a set $A$ of strings on $\Sigma$ is accepted by a non-deterministic Turing machine within storage $L(n) \geq \log n$, then $A$ is accepted by a deterministic Turing machine within storage $(L(n))^2$.

Proof. With each input $w$ to the non-deterministic Turing machine one can associate a tree-like path system $\mathcal{Q}_w$ coded by a relation $R^*_w$ such that $|R^*_w| \leq L(n)$, and $\mathcal{Q}_w$ is solvable if and only if the Turing machine accepts $w$. The proof proceeds as in the above proof.

Theorem 2 If a set $A$ of strings is accepted by a (non-deterministic) multihead two-way pushdown automaton within time $T(n) = c\, n^k$ for some constants $c$ and $k$, then $A$ has tape complexity $(\log_2 n)^2$.

The proof proceeds along the lines of the preceeding corollaries, although the path system is considerably more complicated. This result can be generalized to other types of pushdown machines, including the auxiliary pushdown machines of [2] using results in [2]. Corollaries 1 and 2 above are immediate corollaries of the generalization.

References


1. Cook, S. A. and W. J. Savitch. "Mazes and Turing Machines". Technical Report No. 29, Computer Center, University of California, December, 1968.

2. Cook, S. A. "Variations on Pushdown Machines" Proceedings of the ACM Symposium pm Theory of Computing, May 1969, Marina del Rey, California, pp. 229-232

3. Lewis, P.M. II, R. E. Stearns, and J. Hartmanis. "Memory Bounds for the Recognition of Context-Free and Context Sensitive Languages." IEEE Conference Record on 1965 Symposium on Switching Circuit Theory and Logical Design.

4. Kreider, D. L. and R. W. Ritchie. "Predicatably Computable Functionals and Definitions by Recursion". Zeitschrift fur math. Logiks and Grundlagen der Math., Vol. 10, 65-80 (1964).

5. Savitch, W. J. Nondeterministic Tape Bounded Turing Machines. Doctoral Thesis, University of California, Berkeley, September, 1969.