Fast nondeterministic recognition of context-free languages using two queues

Burton Rosenberg
University of Miami

Abstract

We show how to accept a context-free language nondeterministically in $O(n \log n)$ time on a two-queue machine.

Keywords: Algorithms, Formal Languages, Theory of Computation.

1 Introduction

There has been extensive study of the relative power of automata according to the nature of their external stores. Because of the central role of context-free languages in the Chomsky hierarchy, *pushdown automata* have received enormous attention. A pushdown automaton is a nondeterministic, finite-state automaton whose external store object implements stack semantics, that is, supports only *push*, *pop* and *sense empty* operations. Of less centrality are the *queue automata*: deterministic or nondeterministic automata whose external stores implement queue semantics, that is, support only *enqueue*, *dequeue* and *sense empty* operations. Whereas the pushdown automata characterize a distinct and important language class, the context-free languages, queue automata are language-theoretically equivalent to Turing machines. However, when resource costs are taken into account, the landscape becomes more interesting.

A Turing machine running in time T can be straightforwardly simulated by a queue automaton in time T^2. For a deterministic, one-queue automaton, Li and Vitányi [7] show that this is optimal. If we restrict to real or linear time, the class of languages accepted by queue machines is incomparable with that of pushdown automata, even in the nondeterministic case and with multiple queues. This is studied extensively by Book et al [1] and Brandenburg [2]. The intersection of context-free languages and nondeterministic real-time multiple-queue automata is studied by Brandenburg [3]. The language classes given by deterministic queue machines are more complicated; the theory is developed by Cherubini, Citrini, Reghizzi, and Mandrioli [4].

The results of Li, Longpré and Vitányi [6] using Kolomogorov Complexity show that a one-queue machine needs $\Omega(n^{4/3} / \log n)$ nondeterministic time to simulate one stack. Hühne [5] shows a deterministic simulation of a stack in time $O(n^{1+\epsilon})$ using multiple queues, for real $\epsilon > 0$ decreasing as the number of queues increases. This is also shown by Rosenberg [9] by a slightly simpler construction.

In this paper, we show how off-line, nondeterministic two-queue automata can accept the context-free languages in $O(n \log n)$ time. The proof rests upon nondeterministic parsing according to a context-free grammar in Chomsky normal form.

2 Nondeterministic Parsing

In this section we show that a time bound of $O(n \log n)$ is sufficient for a nondeterministic queue machine with two queues to accept any given context-free language L. Assume without loss of generality that L does not contain the empty string. Then it can be generated by a grammar in Chomsky normal form — that is, where all productions are of the form $A \to BC$ or $A \to a$, with A, B, C nonterminals and a a terminal. Our approach will be to use the grammar in this form to construct a queue machine capable of guessing in time $O(n \log n)$ any and only words in L. The guessed word w is then compared against the input.

The queue machine guesses a string of parentheses-enclosed productions in the grammar:

$$(\alpha_1)(\alpha_2)\ldots(\alpha_r),$$

where $\alpha_i = A \to \gamma$ might occur only if $A \to \gamma$ is a production in the grammar. Each pass over the queue will examine each parenthesized string in left-to-right order and copy it to the end of the queue, either verbatim, or while inserting into it its neighbor to the right. This insertion is according to the rules of the grammar. That is, there are two transformation rules taking parenthesized strings from one queue pass to the next,

1. Verbatim copy: $(\alpha) \Rightarrow (\alpha)$.

2. Insert neighbor: $(A \to \alpha B \gamma)(B \to \beta) \Rightarrow (A \to \alpha\beta\gamma)$.

The insert operation will require a second queue in which to store γ while copying β. The invariant held is that any $(A \to \alpha)$ on the queue implies that the string of terminals and non-terminals α can be generated from A by valid productions in the grammar. When a single item remains on the queue, if it is of the form $(S \to w)$ where S is the initial symbol and w is entirely of terminals, then $w \in L$.

We have shown that the proposed machine can guess only words in the language. Next we show that any word in the language can be guessed using fewer than $Kn \log n$ operations, where K is a constant. Given $w \in L$, consider its parse tree. It is a binary tree with internal nodes labeled with nonterminals and leaves labeled with terminals. The word given by the tree is the list of terminals in depth-first order of a tree search. In fact, take any subtree of the parse tree and remove from it all descendents of any set of nodes in the subtree. Then the "word" given by a depth-first listing of the leaves can be generated from the label of the root by productions in the grammar.

It is convenient that this parse tree be "full". To this end, leaves will be "pulled up" into their respective degree-one parents. We use the following well-known lemma on tree decomposition to guide the cutting up of the parse tree into a list of productions. A *full binary tree* is a connected acyclic graph of degree bound 3, of which there is a unique node of degree 2. Each degree-one node is called a *leaf*, and the degree 2 node is called the *root*. Note that neither the empty tree nor the one-node tree is full and that all full trees have an odd number of nodes.

Lemma 1 *Given a full binary tree T on n nodes, there exists an edge which cuts* *from T a subtree of size m between,*

$$n/3 \leq m \leq 2n/3.$$

Beginning with the parse tree T, we apply the lemma to isolate a subtree T' of T. If the root of T' is v, let T'' be T minus all descendents of v. Place T'' to the left of T' and repeat this procedure on each of the subtrees. In terms of forward queue passes, we have reversed the neighbor insertion:

$$(S \to \alpha B \gamma)(B \to \beta) \Rightarrow (S \to \alpha \beta \gamma).$$

Recall that we have considered the leaf nodes to be part of the parent. Hence this recursive cutting will end when the trees are of size three. At most two additional steps are required to reduce such a tree to a sequence of parenthesized productions. These reverse the neighbor insertions of terminals, such as:

$$(A \to BC)(B \to b) \Rightarrow (A \to bC).$$

Hence any word w in L gives a list of productions which for some order of neighbor insertions recovers w. The length of the symbols on the queue is bounded by $8 \,|\, w \,|$, and by following backwards the tree decomposition process, at most $2 + \log_{3/2} \,|\, w \,|$ passes over the queue are needed. This concludes the proof.

Acknowledgements

This work was supported by a Summer Award in Natural Sciences and Engineering from the Department of Research and Sponsored Programs of the University of Miami.

References

[1] R. V. Book, S. A. Greibach, and C. Wrathall, *Reset Machines,* Journal of Computer and System Sciences 19, 1979, 256–276.

[2] Franz J. Brandenburg, *Multiple Equality Sets and Post Machines,* Journal of Computer and System Sciences 21, 1980, 292–316.

[3] Franz J. Brandenburg, *On the Intersection of Stacks and Queues,* Theoretical Computer Science 58, 1988, 69–80.

[4] Alessandra Cherubini, Claudio Citrini, Stefano Crespi Reghizzi, and Dino Mandrioli, *QRT FIFO Automata, Breadth-First Grammars and their Relations,* Theoretical Computer Science 85, 1991, 171–203.

[5] Martin Hühne, *On the power of several queues,* Theoretical Computer Science 113 (1993) 75–91.

[6] Ming Li, Luc Longpré, Paul Vitányi, *The power of the queue,* SIAM J. Comput, Vol. 21, No. 4, (1992) 697–712.

[7] Ming Li, Paul M. Vitányi, *Tape versus queue and stacks: The lower bounds,* Information and Computation, **78** (1988) 56–85.

[8] Burton Rosenberg, *Simulating a stack by queues,* Anales de 22as. Jornadas Argentinas de Informática e Investigación Operativa, (XIX Conferencia Latinoamericana de Informática), August 1993. Tomo 1, Pp. 3–13.

[9] Burton Rosenberg, *Simulating a stack in $O(n^{1+\epsilon})$ using multiple queues,* manuscript.

[10] Bernard Vauquelin and Paul Franchi-Zannettacci, *Automates à File,* Theoretical Computer Science 11, 1980, 221–225.

[11] R. Vollmar, *Über einen Automaten mit Pufferspeicherung,* Computing 5, 1970, 57–70.