The Expressibility of Nondeterministic Auxiliary Stack Automata and its relation to Treesize Bounded Alternating Auxiliary Pushdown Automata

V.Vinay
Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India

V.Chandru*
Industrial Engineering
Purdue University
W.Lafayette IN 47907, U.S.A.

Abstract

Two aspects of nondeterministic auxiliary stack automata (NAuxSA) are studied in this paper. The first is regarding the expressibility of NAuxSA. More specifically, it is shown that the polynomial hierarchy can be characterised in terms of NAuxSA with resource bounds. The second aspect is a duality relation between NAuxSA and alternating auxiliary pushdown automata (Alt-AuxPDA) connecting time bounds on the former with treesize bounds on the latter.

1 Introduction

Ibarra [Ib 71] proved that nondeterministic auxiliary stack automata (NAuxSA) are extremely powerful machines. He showed that NAuxSA(S(n)) equals DTIME(2^{O(n)}). Alternating Turing machines were introduced in [CKS 81]. Ladner, Lipton and Stockmeyer [LLS 84] studied the effects of alternation on a variety of models including auxiliary pushdown and stack automata. They characterised space bounds on these models in terms of deterministic time. However, the effects of simultaneous resource bounds on stack automata has not received much attention. Here, we show that despite their power, NAuxSA are sensitive enough to capture N^P. We do so, by introducing a new resource called scan to denote the number of times a NAuxSA alternates between the pushdown and scan modes.

The class LOGCFL (class of languages NC^1 reducible to CFLs) was first characterized in [Su 78] as the class of languages accepted by NAuxPDA space,time (log n, n^{O(1)}). It is known that LOGCFL and N^P have similar behaviour, i.e. they have similar characterizations on a variety of models. For example, Ruzzo [Ru 80] showed that both have polynomial treesize on ATMs (they differ in the space they use), Venkateswaran [Ve 87,Ve 88] proved that both have semi-unbounded circuit characterizations with identical depth, O(log n) (they differ in the size of the circuit). Not surprisingly they also have similar pebbling characterizations [VVV 90]. Jenner and Kersig [JK 88] showed that the difference between N^P and LOGCFL

is one alternation, i.e. $\mathcal{NP} = A\Sigma_2 - \text{AuxPDA space, time} \ (\log n, n^{O(1)})$. We prove that the difference between LOGCFL and $\mathcal{NP}$ is one scan! In fact, we prove something stronger - the NAuxSA are nonerasing.

- $\mathcal{NP} = NAuxSA \space, \text{time, scan}(\log n, n^{O(1)}, 1)$

So $\mathcal{NP}$ is an NLOG machine with a pushdown store whose contents can be read once. We believe this is one of the first characterizations of $\mathcal{NP}$ in terms of NLOG machines with additional power. We also prove that

- $\mathcal{P} = DAuxSA \space, \text{time}(\log n, n^{O(1)})$

This reveals the connection between $\mathcal{P}$ and DLOG. In general, we show

- $\Sigma_k^r = A\Sigma_k - \text{AuxSA space, time, scan}(\log n, n^{O(1)}, 1)$ for $k \geq 1$

Motivated by Ruzzo's result [Ru 80] we show that

- For $S(n) \geq \log n$ and $Z(n) = \Omega(2^{O(S(n))})$, $NAuxSA \space, \text{time}(S(n), Z^{O(1)}(n)) = Alt - \text{AuxPDA space, tsz}(S(n), Z^{O(1)}(n))$

This leads to yet another characterization of $\mathcal{NP}$

- $\mathcal{NP} = Alt - \text{AuxPDA space, tsz}(\log n, n^{O(1)})$

## 2 Preliminaries

### 2.1 Alternating Auxiliary Pushdown Automata

An alternating auxiliary pushdown automaton (Alt-AuxPDA) is an alternating turing machine (ATM) with an additional pushdown store. The space used by the machine corresponds to space on the worktape only. A more formal definition is given in [LLS 84]. We assume that the Alt-AuxPDA behaves deterministically while pushing or popping. So a configuration could be universal, existential, push, pop or accepting.

By a surface configuration, $v$, of an Alt-AuxPDA machine $M$ on input $x$, we mean $v = (q,i,\alpha,j,z)$ where $q$ is the current state of $M$, $i$ is the input head position, $\alpha$ is the worktape contents, $j$ is the worktape head position and $z$ is the top of stack symbol. A surface configuration has information only about the stack top rather than the whole stack. We use top to extract the top of stack symbol from a surface configuration (top($v$) = $z$).

By a surface computation tree of $M$ on $x \in L(M)$ we mean a tree $S(x)$ whose vertices are labelled with surface configurations such that a vertex labelled by an existential (push,pop) surface configuration has exactly one child, the universal surface configuration has both children and all leaves are accepting.

Note that every push configuration $r$ has a balancing pop configuration along every path in the subtree rooted at the push configuration. We call these pop configurations $v_1, v_2, \cdots, v_k$ as mates of $r$ and vice-versa [Ve 87]. We call their children $z_1, z_2, \cdots, z_k$ as restarts of $r$. 