Abstract. New results concerning the space complexity of languages accepted by stack automata, alternating stack automata, and alternating pushdown automata are derived. Some of the results generalize previously known results.

1. Introduction

The notion of a path system was first introduced by Cook [2] in order to prove some results concerning certain types of pushdown machines. In a recent paper [5], we considered a new formulation of path systems called parameterized path systems (PPSs). We showed that PPSs can be used to give unified (and simple) proofs of well known theorems concerning resource-bounded computations. Several new results were also obtained in [5]. This paper continues our investigation in [5] to other machine models. Specifically, we derive new results which refine or generalize previously known results [1, 3, 6, 8, 10, 13] concerning stack automata [4], alternating stack automata [10] and alternating pushdown automata [1, 10]. Examples of new results are the following:

1. Any language accepted by a nondeterministic stack automaton (NSA) which makes at most $2^{cn^2}$ alternations between pushing and popping of the stack has deterministic tape complexity $n^4$. This strengthens a result in [6] which shows that any language accepted by a nonerasing NSA has deterministic tape complexity $n^4$.

2. Any language accepted by an alternating NSA which has the property that universal states can only be entered when the machine is in the pushdown mode has deterministic time complexity $2^{cn^2}$. Without this property, the language has deterministic time complexity $2^{2^dn}$ [10].

*Work supported by NSF Grant MCS79-09967.
†Work supported by NSF Grant MCS78-01736.
3. An alternating $S(n)$-tape bounded auxiliary NSA, $S(n) \geq \log n^4$, which has the property that universal states can only be entered when the machine is in the pushdown mode accepts a language $L$ if and only if $L$ is accepted by an $S(n)$-tape bounded auxiliary deterministic stack automaton, and if and only if $L$ is accepted by an $S(n)$-tape bounded alternating auxiliary nondeterministic pushdown automaton.

Section 2 is concerned with ordinary stack automata while Section 3 is concerned with alternating such machines.

2. Stack Automata and Pushdown Automata

We will use the following abbreviations throughout:

1. NTM—nondeterministic Turing machine with a two-way read-only input tape and one read-write storage tape. The deterministic version is denoted by DTM.

2. NSA—two-way nondeterministic stack automaton [4]. A stack is like a pushdown store except that the stack head can go inside the store in a read-only mode. DSA is the deterministic version.

3. NENSA—nonerasing NSA [4]: An NSA which does not pop (i.e., erase) symbols from the stack. NEDSA is the deterministic version.

4. Aux NSA—the NSA with an auxiliary read-write storage (i.e., TM) tape, Aux DSA, etc., are defined similarly.

5. When the stack operates only as a pushdown store, i.e., it operates only in a pushdown mode, NSA reduces to NPDA ( = two-way nondeterministic pushdown automaton). DPDA, Aux NPDA, etc., are defined similarly.

In what follows $S(n)$, $T(n)$, $R(n)$, $C(n)$ and $Z(n)$ are functions on the positive integers.

A machine with a read-write storage tape is $S(n)$-tape bounded if every input of length $n$ that is accepted has an accepting computation in which the read-write storage tape uses no more than $S(n)$ tape squares. The tape bound $S(n)$ does not include the space used in the stack or in the pushdown store. Note that if $S(n) = \text{constant}$, then Aux NSA reduces to NSA. A $T(n)$-time bounded machine is defined similarly. A machine with a stack or a pushdown store is $R(n)$-reversal bounded if every input of length $n$ that is accepted has an accepting computation in which the stack or pushdown store makes at most $R(n)$ alternations between pushing (i.e., writing) and popping (i.e., erasing) of the stack or pushdown store. In the case of a stack, the reversals that the stack head makes inside the stack during a read-only mode are not included in $R(n)$. In particular, if the stack is nonerasing, $R(n)$ can be taken to be 1.

Convention. If a machine satisfies more than one resource constraint, then the constraints are assumed to hold simultaneously for at least one accepting computation (on inputs that are accepted).

---

1The logarithms in this paper are assumed to be in base 2.