Journal of Automata, Languages and Combinatorics **6** (2001) 1, 51–74

ON THE POWER OF SUBROUTINES FOR FINITE STATE MACHINES

MARKUS E. NEBEL
Institut für Informatik, Johann Wolfgang Goethe-Universität
D-60054 Frankfurt am Main, Germany
e-mail: nebel@sads.informatik.uni-frankfurt.de

ABSTRACT

In this paper we extend the finite state machines by a subroutine concept. Two implementations are considered. The first implementation yields a new class of languages which is a subclass of the context-free languages. The second one leads to an alternative automata-model for the context-free languages. Besides the generative capacity other properties like determinism, reversal languages, etc. are also studied. We prove that determinism for the second implementation is equivalent to the notion of $LL(1)$-languages. The motivation for those observations comes from a description language for plot data called *DPF* which is used in practice and which possesses simple non-regular constructions only.

Keywords: automata-theory, deterministic parsing, formal languages.

1. Introduction

The generative capacity of finite state machines or alternatively regular expressions, i. e. right-linear grammars, suffices only for a few examples of the application of formal languages. We often need the possibility to express couplings between different parts of a word, e. g. in programming languages where the start of a loop has to be terminated somewhere later in the program. One possibility to obtain such a coupling is to think of a finite state machine with a subroutine concept. In this paper two models are considered on how to extend a normal finite state machine by a subroutine concept.

- **The weak model:** Here, the finite state machine is extended by a stack on which return-addresses (i. e. states) may be stored. There is no real call command for the subroutine-call. After pushing the return address one has to use a normal transition to give the control to the subroutine (see Section 3).
- **The strong model:** Here, the weak model is extended by a real call command (see Section 4).

In both concepts a final state may by interpreted as a signal to return, i. e. to continue the computation with the state on top of the stack.

Immediately the question arises: What generative capacity do both models imply? In the following sections we will answer this question.