Journal of Automata, Languages and Combinatorics **7** (2002) 2, 247–258

THE COMPUTING POWER OF PROGRAMS OVER FINITE MONOIDS [1]

Pascal Tesson and Denis Thérien [2]

School of Computer Science, McGill University
3480 University, Montréal, Québec, H3A 2A7 Canada
e-mail: {ptesso,denis}@cs.mcgill.ca

ABSTRACT

The formalism of programs over monoids has been studied for its close connection to parallel complexity classes defined by small-depth boolean circuits. We investigate two basic questions about this model. When is a monoid rich enough that it can recognize arbitrary languages (provided no restriction on length is imposed)? When is a monoid weak enough that all its computations can be realized in polynomial length? Surprisingly, these two properties appear to be dual to each other.

Keywords: Programs over monoids, nilpotent groups, unambiguous concatenation, wreath product

1. Introduction

Finite monoids can be used as language recognizers in many different ways. Classically, one would use a morphism $\phi : A^* \to M$ and a subset $F \subseteq M$ to recognize the language $L = \phi^{-1}(F) \subseteq A^*$. It is well-known that this framework characterizes the class of regular languages and the algebraic point of view provides a most powerful set of tools to understand and classify the combinatorial properties of such languages (see [5] and [6] for a detailed description of this approach). In this model, the morphism can be seen as a very uniform way to translate a string $a_1 \dots a_n$ in A^* to a string $\phi(a_1) \dots \phi(a_n)$ of monoid elements which is then evaluated in the monoid to yield the value of the "machine" M on its input.

In [1] and [4], a more general device to transform a string in A^* into a string of monoid elements was introduced. An n-input M-program takes as input a word of length n over the alphabet A. It is allowed to query the input positions in arbitrary order and each position can be queried several times. At each query, the letter read in the given position is transformed to a monoid element (precise definition is given in the next section). In this way, the input word $w = a_1 \dots a_n$ gives rise to a string

[1] Full version of a submission presented at the workshop *Logic and Algebra for Concurrency* held at Dresden University of Technology (Germany), September 13 – 16, 2000.

[2] Supported by NSERC and FCAR. This work was completed while the second author was in Germany, supported by the von Humboldt Foundation.