Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Transducers based on networks of polarized evolutionary processors

Fernando Arroyo [a], Sandra Gómez-Canaval [a], Victor Mitrana [a,b,*], José Ramón Sánchez-Couso [a]

[a] Department of Information Systems, University College of Computer Science, Polytechnic University of Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain
[b] Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, 010014, Bucharest, Romania

ARTICLE INFO

Article history:
Received 22 July 2016
Received in revised form 15 October 2016
Accepted 27 October 2016
Available online xxxx

Keywords:
Evolutionary rule
Network of polarized evolutionary processors
Generalized sequential machine
NPEP-transducer

ABSTRACT

We consider a new type of transducer that does not scan sequentially the input word. This transducer is actually a network of polarized evolutionary processors (*NPEP*) which receives a word as input and collects in the output node, when the computation halts, the translation of the input word. We prove that these transducers can simulate the work of generalized sequential machines on every input. Furthermore, all words obtained by a given finite state transducer by the shortest computations on a given word can also be computed by the new transducers. Unlike the case of generalized sequential machines, every recursively enumerable language can be the transduction, defined by the new transducer, of a very simple regular language.

1. Introduction

In [6], a new type of transducer that does not scan sequentially the input word is considered. This transducer is based on the networks of evolutionary processors (*NEP*) structure, namely it consists of a directed graph whose nodes are processors which work in parallel and are specialized in just one type of a very simple evolutionary operation: inserting, deleting or substituting a symbol by another one. The computation on an input word starts with this word placed in a designated node, the input node, of the network and alternates between evolutionary and communication steps. The computation halts as soon as another designated node, the output node, is nonempty. The translation of the input word is the set of words existing in the output node when the computation halts. In [6], it is shown that these transducers can simulate the work of generalized sequential machines on every input. Furthermore, all words obtained by a given generalized sequential machine by the shortest computations on a given word can also be computed by the new transducers. It is also proved that every recursively enumerable language can be the transduction defined by the new transducer of a very simple regular language over the one-letter alphabet. The transducer actually uses its input as a counter for the number of derivation steps in a random-context grammar with the leftmost derivation. The same idea may be used for proving that these transducers can simulate the shortest computations of an arbitrary Turing machine, used as a transducer, on every input word.

* Corresponding author at: Department of Information Systems, University College of Computer Science, Polytechnic University of Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain.
E-mail addresses: farroyo@etsisi.upm.es (F. Arroyo), sgomez@etsisi.upm.es (S. Gómez-Canaval), victor.mitrana@upm.es (V. Mitrana), jcouso@etsisi.upm.es (J.R. Sánchez-Couso).

http://dx.doi.org/10.1016/j.tcs.2016.10.014
0304-3975/

Work [1] considers a new variant of *NEP* with the aim of proposing a new type of filtering process and discusses the potential of this variant for solving hard computational problems. This investigation is then continued in [2]. Informally, in a network of polarized evolutionary processors (*NPEP*, for short) all nodes are "polarized" in the sense that a sign, including 0, is associated with each node of the network. We say that a node can be negative, neutral or positive. The words that become mobile agents can navigate through the network following a protocol defined by a filtering strategy based on polarization. This strategy is represented by an analogy to the electrical charge (positive, negative or neutral) which is applied to both words and nodes. While the polarization of a node is previously defined and fixed for the whole computation process, the polarization of every word is dynamically computed by a valuation mapping which computes the polarity of that word depending on the values assigned to its symbols. Actually, the valuation mapping does not give the exact value, but just the sign of this value. By means of this valuation, one may metaphorically say that the words are electrically polarized. Words migration from one node to another depends both on their polarization and the node polarization which for simplicity reasons have to be the same. Thus, the words migration from one node to another might simulate the communication channel between the two cells. It is worth mentioning that this model is closely related to the flow-based programming paradigm, see, e.g. [7]. A flow-based programming application defines also a network whose nodes are called "black box" processes which exchange data among them across some connections that are externally specified. In an *NPEP* the connections among the processors are fixed for the whole computation. Note also the difference between this model and the model of distributed computing with mobile agents [5], which is preferable when the data needed for computation is physically dispersed. Particularly, *NPEP* has been proved to be computationally complete in two recent works [3] and [4].

In this work we define a new variant of a transducer based on the *NEP* structure but with the same strategy of communication as that in an *NPEP*. The main differences with respect to the transducer introduced in [6] are: (i) the underlying graph of an *NPEP*-transducer is not a directed graph anymore, but an undirected graph; and (ii) the protocol of cooperation is regulated by the compatibility between the polarity of words and that of nodes. The results presented here are similar to those reported in [6], but obtained in different ways. We prove that these transducers can simulate the shortest computations of finite state transducers as well as the work of generalized sequential machines. Starting from this last result we show that *NPEP*-transducers can receive an input, that encodes a positive integer n, and output the set of all words computed by a generalized sequential machine after n iterative application steps. As a consequence, we show that every recursively enumerable language can be the transduction, defined by the new transducer, of a very simple regular language over the one-letter alphabet, namely the language defined by the regular expression a^+.

2. Basic definitions

We assume the reader to be familiar with fundamental concepts from Formal Language Theory which can be found in many textbooks, e.g., the handbook [8]. We start by summarizing the notions used throughout the paper. An *alphabet* is a finite and nonempty set of symbols. Any finite sequence of symbols from an alphabet V is called a word over V. The set of all words over V is denoted by V^* and the empty word is denoted by ε. The length of a word x is denoted by $|x|$ while $|x|_a$ denotes the number of occurrences of the symbol a in x. For simplicity, we identify a regular language by its regular expression.

A homomorphism from the monoid V^* into the monoid (group) of additive integers $\mathbf{Z}$ is called *valuation* of V^* in $\mathbf{Z}$. The absolute value of an integer k is denoted by $|k|$. Although the length of a word and the absolute value of an integer is denoted in the same way, this cannot cause any confusion as the arguments are understood from the context.

The following rewriting operations might be viewed as linguistic formulations for point mutations of nucleotides in a DNA molecule or local transformations of substances in chemical reactions. A rule $a \to b$ with $a, b \in V \cup \varepsilon$ is defined a *substitution rule* if both a and b are not ε; it is a *deletion rule* if $a \neq \varepsilon$ and $b = \varepsilon$; it is an *insertion rule* if $a = \varepsilon$ and $b \neq \varepsilon$. The sets of all substitution, deletion, and insertion rules over an alphabet V are denoted by Sub_V, Del_V, and Ins_V, respectively.

Given a rule σ as above and a word $w \in V^*$, the following *actions* of σ on w are defined by:

- If $\sigma \equiv a \to b \in Sub_V$, then $\sigma(w) = \begin{cases} \{ubv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise.} \end{cases}$
- If $\sigma \equiv a \to \varepsilon \in Del_V$, then

$$\sigma^r(w) = \begin{cases} \{u : w = ua\}, \\ \{w\}, \text{ otherwise} \end{cases} \qquad \sigma^l(w) = \begin{cases} \{v : w = av\}, \\ \{w\}, \text{ otherwise.} \end{cases}$$

- If $\sigma \equiv \varepsilon \to a \in Ins_V$, then $\sigma^r(w) = \{wa\}$, $\sigma^l(w) = \{aw\}$.

Note that $\alpha \in \{l, r\}$ expresses the way of applying a deletion or insertion rule to a word, namely in the left ($\alpha = l$), or in the right ($\alpha = r$) end of the word, respectively. It is worth mentioning that the action mode of a substitution rule applied to a word w: it returns the set of all words that may be obtained from w depending on the position in w where the rule was actually applied.

Please cite this article in press as: F. Arroyo et al., Transducers based on networks of polarized evolutionary processors, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.10.014