Journal of Computer and System Sciences ••• (••••) •••–•••

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Networks of polarized multiset processors

Paolo Bottoni [a], Anna Labella [a], Victor Mitrana [b,*,1]

[a] Department of Computer Science, "Sapienza" University of Rome, Via Salaria 113, 00198 Rome, Italy
[b] Faculty of Mathematics and Computer Science, University of Bucharest, Str. Academiei 14, 010014, Bucharest, Romania

ARTICLE INFO

Article history:
Received 10 June 2016
Received in revised form 11 November 2016
Accepted 14 November 2016
Available online xxxx

Keywords:
Multiset
Macroset
Multiset Turing machine
Polarized multiset processor
Network of polarized multiset processors

ABSTRACT

We propose a highly parallel and distributed multiset computing model having as its underlying structure an undirected graph whose nodes are processors, each endowed with a polarity and with a set of rules all of the same kind, one of increment, decrement or substitution. Processors communicate with each other via a protocol based on the compatibility between their polarization and the polarization of the data, as computed by a valuation mapping. We show that this model can simulate any multiset Turing machine. In its turn, the new model can be simulated by the most general variant of multiset Turing machine.

1. Introduction

The notion of rewriting is in principle applicable to any discrete structure composed of individual elements which can undergo different evolutions, while maintaining the same overall structure. Individual elements can be more or less complex and structures can be more or less complex and constrained, multisets of atoms representing one of the simplest possible combinations.

In [1] a formalism based on multiset rewriting, called *constraint multiset grammar* was introduced with the aim of providing a high-level framework for the definition of visual languages. Constraint multiset grammars were investigated in [2] with a view to the parsing complexity of visual languages. These grammars may be viewed as a bridge between the usual string grammars, dealing with string rewriting, and constraint logic programs, dealing with set rewriting. Other devices based on multiset rewriting have been reported in [3,4] where *abstract rewriting multiset systems* were used for modelling some features of the population development in artificial cell systems. Furthermore, a Chomsky-like hierarchy of grammars rewriting multisets was proposed in [5]. In all these approaches, the multisets are processed in a sequential, non-distributed way. In a membrane system [6], multisets are processed in parallel; moreover, the structures on which rewriting occurs can in turn be the subject of transformations, where membranes that present different levels of nesting can be dissolved. The work presented here proposes another highly parallel and distributed way of processing multisets resembling a bio-inspired computational model based on string rewriting introduced in [7], called network of polarized evolutionary processors (NPEP for short).

Informally speaking, such a network can be defined as a graph whose nodes host processors performing evolutionary operations on the strings contained in the corresponding node. An NPEP alternates evolutionary and communication steps,

* Corresponding author.
E-mail addresses: bottoni@di.uniroma1.it (P. Bottoni), labella@di.uniroma1.it (A. Labella), mitrana@fmi.unibuc.ro (V. Mitrana).

[1] Research supported by the Visiting Professor Program – "Sapienza" University of Rome.

http://dx.doi.org/10.1016/j.jcss.2016.11.006
0022-0000/

until a predefined halting condition is fulfilled. In each evolutionary step, all processors change simultaneously the contents of their nodes according to their associated sets of rules. In the communication step, all the strings that satisfy the communication protocol are exchanged between connected nodes. The protocol regulating the communication step is based on a valuation mapping which assigns to each string an integer value, depending on the values assigned to its symbols. Actually, it is not the exact value of a string which is considered, but just the sign of this value. By means of this valuation, one may metaphorically say that the strings are electrically polarized. Thus, as the nodes are polarized as well, string migration from one node to another through the channel between two cells, according to polarization, seems to be more natural, as well as easier to be implemented. It is worth mentioning that this model is closely related to the flow-based programming paradigm, see, e.g. [8]. A flow-based programming application also defines 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 [9], which is preferable when the data needed for computation is physically dispersed.

Note that the structure of the string does not matter when computing its valuation as the valuation of any anagram of a given string is the same. Therefore, we can consider a representation of the commutative closure of a string instead of the string itself. This is our approach in this note. The paper is organized as follows. In the next section, we define the main concepts that will be used throughout the paper: multiset, macroset, valuation mapping, multiset Turing machine, polarized multiset processor, network of polarized multiset processors (NPMP for short), etc. We then prove that NPMPs can simulate every multiset Turing machine. The converse simulation is shown for the most general variant of multiset Turing machine.

2. Basic definitions

For a finite set A, we denote by $card(A)$ the cardinality of A. A finite *multiset* over a finite set A is a mapping $\sigma : A \longrightarrow \mathbb{N}$; $\sigma(a)$ expresses the number of copies of $a \in A$ in the multiset σ. The *empty multiset* over a set A is denoted by ε_A, that is $\varepsilon_A(a) = 0$ for all $a \in A$. We use the same notation for the *empty set* and *empty macroset*, namely $\emptyset$.

In what follows, a multiset containing the elements $b_1, b_2, \ldots, b_r$, any of them possibly with repetitions, will be denoted by $\langle b_1, b_2, \ldots, b_r \rangle$. Each multiset σ over a set A of cardinality n may also be viewed as an array of size n with non-negative entries.

The set of all multisets over A is denoted by $A^{\#}$. A subset of $A^{\#}$ is called *macroset* over A. The *weight* of a multiset σ as above is $\| \sigma \| = \sum_{a \in A} \sigma(a)$, and $\| \sigma \|_B = \sum_{b \in B} \sigma(b)$ for any subset B of A.

Normally, we use lower case Greek letters for multisets and capital Greek letters for macrosets. For two multisets σ, τ over the set A we define

- the addition multiset $\sigma + \tau$ with $(\sigma + \tau)(a) = \sigma(a) + \tau(a)$ for all $a \in A$;
- the difference multiset $\sigma - \tau$ with $(\sigma - \tau)(a) = \max(\sigma(a) - \tau(a), 0)$ for each $a \in A$;
- scalar multiplication multiset $c\sigma$, with $c \in \mathbb{N}$, with $(c\sigma)(a) = c\sigma(a)$ for all $a \in A$.

A function $\varphi : A \longrightarrow \mathbf{Z}$ is called a *valuation* of A in $\mathbf{Z}$. This function may be naturally extended to the set $A^{\#}$ by $\varphi(\sigma) = \sum_{a \in A} \varphi(a) \cdot \sigma(a)$.

2.1. Multiset Turing machines

We recall from [10] the definition of a multiset Turing machine. Roughly speaking, a multiset Turing machine consists of a bag in which a multiset of symbols is placed and of a read–write head which can pick up a symbol out of the bag (that may be the empty symbol) and add at most one symbol (that may also be the empty one) to the content of the bag. Reading (picking up) and writing (adding) a symbol (element) different than the empty symbol is meant as decreasing and increasing the number of copies of that symbol by 1, respectively. Reading or writing the empty symbol is meant to leave the multiset unchanged. The machine works as follows: it starts in the initial state having a multiset in its bag. Now, while being in some state, the machine reads a symbol from the bag, changes its state and writes a symbol to the bag content. Such a machine accepts a multiset when it enters a final state with an empty bag. In any other case, the initial multiset is rejected. It is worth mentioning that multiset Turing machines are very close to some variants of register machines defined in [11].

Formally, a *multiset Turing machine* (shortly, MTM) is a construct

$$M = (Q, V, U, f, q_0, \flat, F),$$

where Q and $F \subseteq Q$ are the finite sets of states and final states, respectively, V and U are the input and the bag alphabets, respectively, $V \subset U$, q_0 is the initial state, $\flat$ is the empty symbol in $U \setminus V$, and f is the transition mapping from $Q \times U$ into the set of all subsets of $Q \times U$. Furthermore, there are no $q, s \in Q$ such that $(s, \flat) \in f(q, \flat)$. A configuration is a pair (q, τ),

Please cite this article in press as: P. Bottoni et al., Networks of polarized multiset processors, J. Comput. Syst. Sci. (2016), http://dx.doi.org/10.1016/j.jcss.2016.11.006