Theoretical Computer Science 431 (2012) 4–12

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

An infinite hierarchy of languages defined by dP systems

Gheorghe Păun [a,b,*], Mario J. Pérez-Jiménez [b]

[a] Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 București, Romania

[b] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

ARTICLE INFO

Keywords:
Membrane computing
dP system
Infinite hierarchy
Simple matrix grammar

ABSTRACT

Here, we continue the study of the recently introduced dP automata. They are symport/antiport P systems consisting of a number of components, each one accepting a string, and working together in recognizing the concatenation of these separate strings; the overall string is distributed to the dP automaton components in a balanced way, i.e., in equal parts up to one symbol, like in the communication complexity area. The question whether or not the number of components induces an infinite hierarchy of the recognized languages was formulated as an open problem in the literature. We solve here affirmatively this question (by connecting P automata with right linear simple matrix grammars), then we also briefly discuss the relation between the balanced and the non-balanced way of splitting the input string among components; settling this latter problem remains as a research topic. Some other open problems are also formulated.

1. Introduction

In the membrane computing area (the reader is referred to [9,12], and to the domain website [15] for details), there are many classes of computing devices, generating or accepting multisets, numbers or strings. Here we deal with devices which accept strings, namely, based on symport/antiport rules. (Couples of objects are passed simultaneously across a membrane, in the same direction in the case of symport and in opposite directions in the case of antiport). Basically, the objects which are taken from the environment during a halting computation are arranged in a string in the order of "reading" them, and the obtained string is said to be accepted by the system. This idea was first explored in [3] (the paper was presented during the Workshop on Membrane Computing, Curtea de Argeş, 2002; the respective devices were called P automata) and, almost concomitantly, in [6], in a simplified version. Several papers were devoted to these devices (in particular, characterizations of regular, context-free, and recursively enumerable languages were obtained, and complexity investigations were carried out); we refer to [2] for details, including references.

Although all P systems are distributed computing machinery, the result of a computation – in particular, the string to be accepted by a P automaton – is produced in a single membrane (or as the input of a single membrane), distinguished in advance. In order to solve a problem – in particular, to accept a string – in a distributed way, a class of P systems was introduced in [10], called dP systems. In the general case, such systems consist of a given number of usual P systems, of any type, which can have their separate inputs and communicate from skin to skin membranes by means of antiport rules (like in tissue-like P systems). In this framework, communication complexity issues can be investigated, as in [7]. The case of P automata (based on symport/antiport rules) was considered in some details – and this leads to the notion of *dP automata*. These devices were further investigated in [5,11], by comparing their power with that of usual P automata and with families

* Corresponding author at: Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 București, Romania. Tel.: +40 1 2229826.
E-mail addresses: gpaun@us.es, George.Paun@imar.ro (G. Păun), marper@us.es (M.J. Pérez-Jiménez).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.12.053

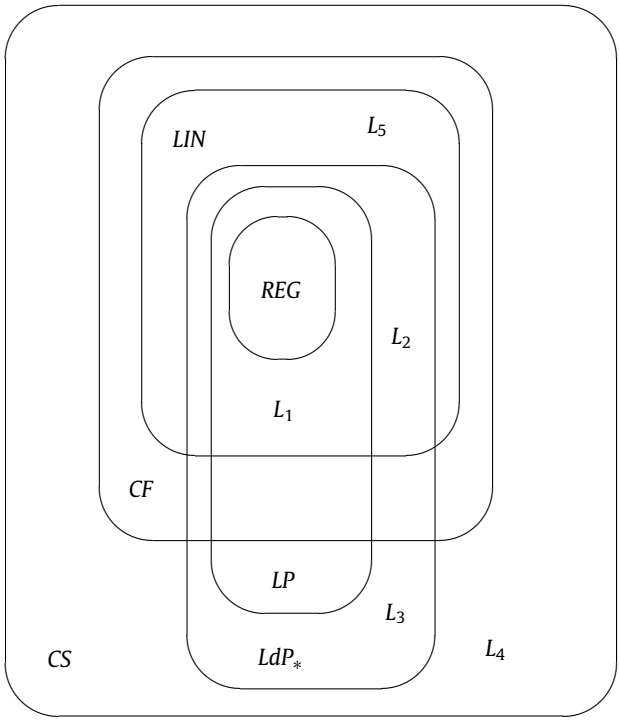

Fig. 1. The place of the families *LP* and *LdP* in Chomsky hierarchy.

of languages in the Chomsky hierarchy. As expected, due to the distribution (and synchronization), dP automata are strictly more powerful than P automata, but the family of languages recognized by them is strictly included in the family of context-sensitive languages. Also expected is the fact that each regular languages can be recognized by a P automaton – see precise details in Fig. 1. (Note that we compare here non-extended P and dP automata, hence without a distinguished terminal alphabet; in the extended case, the P automata are known to be computationally complete.)

A problem left open in [11] asks whether or not the number of components of a dP automaton induces an infinite hierarchy of accepted languages – and this problem is settled here affirmatively. The proof uses a natural connection between dP automata with certain properties and right linear simple matrix grammars, an "old" notion in formal language theory, [8,4]. In this context we provide a simplified proof of the fact that each regular language can be recognized by a P automaton. The possibility to extend such a result/construction to right linear simple matrix grammars and dP automata is formulated as a conjecture.

We also briefly discuss the relationship between the balanced and the non-balanced distribution of a string among the components of a dP automaton, conjecturing that the non-balanced case is more powerful – but the complete solution remains as a task for further research.

2. dP automata

The reader is assumed to have some familiarity with basics of membrane computing, e.g., from [9,12], and of formal language theory, e.g., from [4,13,14], but we recall below all necessary notions.

In what follows, V^* is the free monoid generated by the alphabet V, λ is the empty word, $V^+ = V^* - \{\lambda\}$, $|x|$ denotes the length of the string $x \in V^*$, and $mi(x)$ is the mirror image of $x \in V^*$. *REG*, *LIN*, *CF*, *CS*, *RE* denote the families of regular, linear, context-free, context-sensitive, and recursively enumerable languages, respectively. As usual in membrane computing, the multisets over an alphabet V are represented by strings in V^*; a string and all its permutations correspond to the same multiset, with the number of occurrences of a symbol in a string representing the multiplicity of that object in the multiset. (We work here only with multisets of finite multiplicity.) The terms "symbol" and "object" are used interchangeably, all objects are here represented by symbols.

A *dP automaton* (of degree $n \geq 1$) is a construct

$$\Delta = (O, E, \Pi_1, \ldots, \Pi_n, R),$$

where:

(1) O is an alphabet (of objects);
(2) $E \subseteq O$ (the objects available in arbitrarily many copies in the environment);