Nat Comput (2011) 10:429–445
DOI 10.1007/s11047-010-9238-5

Complexity-preserving simulations among three variants of accepting networks of evolutionary processors

Paolo Bottoni · Anna Labella · Florin Manea · Victor Mitrana · Ion Petre · Jose M. Sempere

Published online: 21 January 2011
© Springer Science+Business Media B.V. 2011

Abstract In this paper we consider three variants of accepting networks of evolutionary processors. It is known that two of them are equivalent to Turing machines. We propose here a direct simulation of one device by the other. Each computational step in one model is simulated in a constant number of computational steps in the other one while a

Florin Manea is on leave of absence from the University of Bucharest.

P. Bottoni · A. Labella
Department of Computer Science, "Sapienza" University of Rome, Via Salaria 113, 00198 Rome, Italy
e-mail: bottoni@di.uniroma1.it

A. Labella
e-mail: labella@di.uniroma1.it

F. Manea
Faculty of Computer Science, Otto-von-Guericke University, P.O. Box 41 20, 39016 Magdeburg, Germany
e-mail: flmanea@gmail.com

V. Mitrana (✉)
Faculty of Mathematics, University of Bucharest, Str. Academiei 14, 010014 Bucharest, Romania
e-mail: mitrana@fmi.unibuc.ro

V. Mitrana
Depto. Organización y Estructura de la Información, Universidad Politécnica de Madrid, Crta. de Valencia km. 7, 28031 Madrid, Spain

I. Petre
Department of Information Technologies, Åbo Akademi University, ICT-building, Joukahaisenkatu 3-5 A, 20520 Turku, Finland
e-mail: ipetre@abo.fi

J. M. Sempere
Department of Information Systems and Computation, Technical University of Valencia, Camino de Vera s/n., 46022 Valencia, Spain
e-mail: jsempere@dsic.upv.es

translation via Turing machines squares the time complexity. We also discuss the possibility of constructing simulations that preserve not only complexity, but also the shape of the simulated network.

Keywords Evolutionary processor · Uniform evolutionary processor · Network of evolutionary processors · Filtered connection

1 Introduction

A basic architecture for parallel and distributed computing consists of several processors, each of them placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. Each node processor acts on the local data in accordance with some predefined rules. Local data is then sent through the network according to well-defined protocols. Only data which is able to pass a filtering process can be communicated. This filtering process may be required to satisfy some conditions imposed by the sending processor, by the receiving processor, or by both of them. All the nodes simultaneously send their data and the receiving nodes also simultaneously handle all the arriving messages, according to specific strategies. This general architecture is met in several areas of Computer Science like Artificial Intelligence (Hillis 1985; Fahlman et al. 1983), Symbolic Computation (Errico and Jesshope 1994), Grammar Systems (Păun and Sântean 1989), and Membrane Computing (Păun 2000).

A further example of such an architecture is the *accepting hybrid network of evolutionary processors* (AHNEP for short) originated in connection with the work (Csuhaj-Varjú and Salomaa 1997), where a distributed computing device called a network of language processors is proposed. A network of language processors consists of several language generating devices associated with nodes of a virtual graph that rewrite words (representing the current state of the nodes) according to some prescribed rewriting mode and communicate the obtained words along the network using input and output filters defined by the membership condition with respect to some regular languages.

In (Castellanos et al. 2001) the concept (considered from a formal language theory point of view in (Csuhaj-Varjú and Salomaa 1997)) was modified in the following way inspired by cell biology (see also (Csuhaj-Varjú and Mitrana 2000) that considers a computing model which might model some properties of evolving cell communities at the syntactical level). Each processor placed in a node is called an evolutionary processor, i.e. an abstract processor which is able to perform very simple operations, namely point mutations in a DNA sequence (insertion, deletion or substitution of a pair of nucleotides). More generally, each node may be viewed as a cell having genetic information encoded in DNA sequences which may evolve by local evolutionary events, i.e. point mutations. Each node is specialized just for one of these evolutionary operations. Furthermore, the data in each node is organized in the form of multisets of words (each word may appear in an arbitrarily large number of copies), and all copies are processed in parallel so that all the possible events that can take place do actually take place. Furthermore, all the nodes simultaneously send their data and the receiving nodes also simultaneously handle all the arriving messages, according to some strategies modeled as permitting and forbidding filters and filtering criteria; see (Margenstern et al. 2005). A series of papers was devoted to different variants of this model viewed as language generating devices; a few rather recent works investigating this model are (Alhazov et al. 2009a, b; Dassow and Truthe 2007). The work (Martín-Vide and Mitrana 2005) is an early survey in this area.

The reader interested in a more detailed discussion about the accepting model is referred to (Margenstern et al. 2005; Manea et al. 2007). In (Margenstern et al. 2005) it is shown that this model is computationally complete and a characterization of the complexity class **NP** based on AHNEPs is presented.

It is clear that filters associated with each node of an AHNEP allow a strong control of the computation. Indeed, every node has one input and one output filter; two nodes can exchange data if it passes the output filter of the sender *and* the input filter of the receiver. Moreover, if some data is sent out by some node and not able to enter any node, then it is lost. The AHNEP model considered in (Margenstern et al. 2005) is simplified in (Drăgoi et al. 2007) by moving the filters from the nodes to the edges. Each edge is viewed as a two-way channel such that the input and output filters, respectively, of the two nodes connected by the edge coincide. Clearly, the possibility of controlling the computation in such networks seems to be diminished. For instance, there is no possibility to discard data during the communication steps. In spite of this fact, in the aforementioned work one proves that these new devices, called accepting networks of evolutionary processors with filtered connections (AHNEPFC) are still computationally complete. This means that moving the filters from the nodes to the edges does not decrease the computational power of the model. Although the two variants are equivalent from the computational power point of view, no direct proof for this equivalence has been proposed until the work (Bottoni et al. 2009a), where direct simulations between the two variants are presented. Moreover, both simulations are time efficient, namely each computational step in one model is simulated in a constant number of computational steps in the other. This is particularly useful when one wants to translate a solution from one model into the other, whereas a translation via a Turing machine squares the time complexity of the new solution. The aim of this paper is to consider another variant which simplifies the general AHNEP model such that filters remain associated with nodes but the input and output filters of every node coincide. This variant called accepting networks of uniform evolutionary processors (UAHNEP) is situated somehow in between the aforementioned ones. This paper is along the same lines of (Bottoni et al. 2009a) and extends it with a new simulation between AHNEP and UAHNEP that is still complexity-preserving. Moreover and rather unexpectedly, this simulation also preserves the completeness of the simulated network, a property that does not always hold for the other two simulations.

2 Basic definitions

We start by summarizing the notions used throughout the paper (for more details see Rozenberg and Salomaa 1997). An *alphabet* is a finite and nonempty set of symbols. The cardinality of a finite set A is written $card(A)$. Any finite sequence of symbols from an alphabet V is called *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 $alph(x)$ denotes the minimal alphabet W such that $x \in W^*$.

We say that a rule $a \to b$, with $a, b \in V \cup \{\varepsilon\}$ and $ab \neq \varepsilon$ is 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 set 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 $w \in V^*$, we define the following *actions* of σ on w:

– If $\sigma \equiv a \to b \in Sub_V$, then

$$\sigma^*(w) = \begin{cases} \{ubv : \exists u, v \in V^* \ (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

Note that a rule as above is applied to all occurrences of the letter a in different copies of the word w. An implicit assumption is that arbitrarily many copies of w are available.

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

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

– If $\sigma \equiv \varepsilon \to a \in Ins_V$, then $\sigma^*(w) = \{uav : \exists u, v \in V^* \ (w = uv)\}$, $\sigma^r(w) = \{wa\}$, $\sigma^l(w) = \{aw\}$.

$\alpha \in \{*, l, r\}$ expresses the way of applying a deletion or insertion rule to a word, namely at any position ($\alpha = *$), in the left ($\alpha = l$), or in the right ($\alpha = r$) end of the word, respectively. For every rule σ, action $\alpha \in \{*, l, r\}$, and $L \subseteq V^*$, we define the *α-action of σ on L* by $\sigma^\alpha(L) = \bigcup_{w \in L} \sigma^\alpha(w)$. Given a finite set of rules M, we define the *α -action of M* on the word w and the language L by:

$$M^\alpha(w) = \bigcup_{\sigma \in M} \sigma^\alpha(w) \text{ and } M^\alpha(L) = \bigcup_{w \in L} M^\alpha(w),$$

respectively. By convention we set $\emptyset^\alpha(w) = \{w\}$. In what follows, we shall refer to the rewriting operations defined above as *evolutionary operations* since they may be viewed as linguistic formulations of local DNA mutations.

For two disjoint subsets P and F of an alphabet V and a word z over V, we define the predicates:

$$\begin{array}{lll} \varphi^{(s)}(z; P, F) \equiv P \subseteq alph(z) & \wedge & F \cap alph(z) = \emptyset \\ \varphi^{(w)}(z; P, F) \equiv (P \neq \emptyset) \to (alph(z) \cap P \neq \emptyset) & \wedge & F \cap alph(z) = \emptyset. \end{array}$$

The construction of these predicates is based on *random-context conditions* defined by the two sets P (*permitting contexts/symbols*) and F (*forbidding contexts/symbols*). Informally, the first condition ((s) stands for strong) requires that all permitting symbols are present in z and no forbidding symbol is present in z, while the second one ((w) stands for weak) is a weaker variant of the first, requiring that at least one permitting symbol appears in z and no forbidding symbol is present in z. For every language $L \subseteq V^*$ and $\beta \in \{(s), (w)\}$, we define:

$$\varphi^\beta(L, P, F) = \{z \in L \mid \varphi^\beta(z; P, F)\}.$$

An *evolutionary processor over V* is a tuple (M, PI, FI, PO, FO), where:

– M is a set of substitution, deletion or insertion rules over the alphabet V. Formally: $(M \subseteq Sub_V)$ or $(M \subseteq Del_V)$ or $(M \subseteq Ins_V)$. The set M represents the set of evolutionary rules of the processor. As one can see, a processor is "specialized" in one evolutionary operation only.
– $PI, FI \subseteq V$ are the *input* permitting/forbidding contexts of the processor, while $PO, FO \subseteq V$ are the *output* permitting/forbidding contexts of the processor. Informally, the permitting contexts sets of symbols that should be present in a word, for it enters/

leaves the processor, while the forbidding contexts sets of symbols that should not be present in a word for it enters/leaves the processor.

An evolutionary processor as above with $PI = PO = P$ and $FI = FO = F$ is called a *uniform evolutionary processor* and is defined as the triple (M, P, F). We denote the set of (uniform) evolutionary processors over V by $(U)EP_V$. Obviously, the (uniform) evolutionary processor described here is a mathematical concept similar to that of an evolutionary algorithm, both being inspired by the Darwinian evolution. The rewriting operations we have considered might be interpreted as mutations and the filtering process described above might be viewed as a selection process. Recombination is missing but it was asserted that evolutionary and functional relationships between genes can be captured by taking only local mutations into consideration (Sankoff et al. 1992). Furthermore, we are not concerned here with a possible biological implementation of these processors, though it is a matter of great importance.

An *accepting hybrid network of evolutionary processors* (AHNEP for short) is a 7-tuple $\Gamma = (V, U, G, \mathcal{N}, \alpha, \beta, x_I, x_O)$, where:

- V and U are the input and network alphabets, respectively, $V \subseteq U$.
- $G = (X_G, E_G)$ is an undirected graph without loops, with the set of nodes X_G and the set of edges E_G. Each edge is given in the form of a binary set. G is called the *underlying graph* of the network.
- $\mathcal{N} : X_G \longrightarrow EP_U$ is a mapping which associates with each node $x \in X_G$ the evolutionary processor $\mathcal{N}(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$.
- $\alpha : X_G \longrightarrow \{*, l, r\}$; $\alpha(x)$ gives the action mode of the rules of node x on the words existing in that node.
- $\beta : X_G \longrightarrow \{(s), (w)\}$ defines the type of the *input/output filters* of a node. More precisely, for every node, $x \in X_G$, the following filters are defined:

$$\text{input filter} : \rho_x(\cdot) = \varphi^{\beta(x)}(\cdot; PI_x, FI_x),$$
$$\text{output filter} : \tau_x(\cdot) = \varphi^{\beta(x)}(\cdot; PO_x, FO_x).$$

That is, $\rho_x(z)$ (resp. τ_x) indicates whether or not the word z can pass the input (resp. output) filter of x. More generally, $\rho_x(L)$ (resp. $\tau_x(L)$) is the set of words of L that can pass the input (resp. output) filter of x.

- x_I and $x_O \in X_G$ are the *input node*, and the *output node*, respectively, of the AHNEP.

An *accepting hybrid network of uniform evolutionary processors* (abbreviated as UAHNEP) is an AHNEP with uniform evolutionary processors only.

An *accepting hybrid network of evolutionary processors with filtered connections* (shortly AHNEPFC) is an 8-tuple $\Gamma = (V, U, G, \mathcal{R}, \mathcal{N}, \alpha, \beta, x_I, x_O)$, where:

- V, U, G, α, x_I, and x_O have the same meaning as for AHNEPs.
- $\mathcal{R} : X_G \longrightarrow 2^{Sub_U} \cup 2^{Del_U} \cup 2^{Ins_U}$ is a mapping which associates with each node the set of evolutionary rules that can be applied in that node. As above, each node is associated only with one type of evolutionary rules.
- $\mathcal{N} : E_G \longrightarrow 2^U \times 2^U$ is a mapping which associates with each edge $e \in E_G$ the disjoint sets $\mathcal{N}(e) = (P_e, F_e)$, $P_e, F_e \subset U$.
- $\beta : E_G \longrightarrow \{(s), (w)\}$ defines the *filter* type of an edge.

Figure 1 makes the differences between the three variants as well as the gradual way of passing from the (intuitively) more complex variant to the simplest one clearer. In this

figure, we sketch two connected nodes (represented by circles) for every variant and the filters associated with them or with their connection.

For all three variants we say that $card(X_G)$ is the size of Γ. When we want to refer to any of the three variants we use the notation [U]AHNEP[FC].

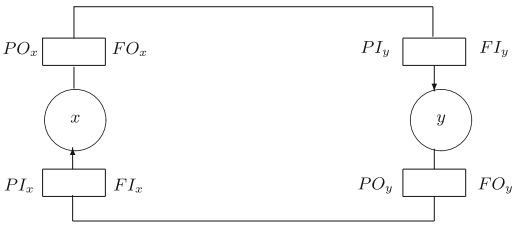
A *configuration* of an [U]AHNEP[FC] Γ as above is a mapping $C : X_G \longrightarrow 2^{V^*}$ which associates a set of words with every node of the graph. A configuration may be understood as the sets of words which are present in any node at a given moment. A configuration can change either by an *evolutionary step* or by a *communication step*.

An evolutionary step is common to all models. When changing by an evolutionary step each component $C(x)$ of the configuration C is changed in accordance with the set of evolutionary rules M_x associated with the node x and the way of applying these rules $\alpha(x)$. Formally, we say that the configuration C' is obtained in *one evolutionary step* from the configuration C, written as $C \Longrightarrow C'$, if and only if

$$C'(x) = M_x^{\alpha(x)}(C(x)) \text{ for all } x \in X_G.$$

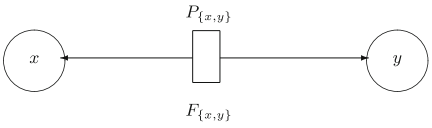
A communication step is common to AHNEP and UAHNEP. When changing by a communication step, each node processor $x \in X_G$ of an [U]AHNEP sends one copy of each word it has (without keeping any copy of it), which is able to pass the output filter of x, to all the node processors connected to x and receives all the words sent by any node processor connected with x provided that they can pass its input filter. Formally, we say that the configuration C' is obtained in *one communication step* from configuration C, written as $C \vdash C'$, if and only if

$$C'(x) = (C(x) \setminus \tau_x(C(x))) \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \rho_x(C(y)))$$



Processors connected in an AHNEP

Processors connected in an UAHNEP.
The two filters of each node in an AHNEP collapsed into only one.



Processors connected in an AHNEPFC
The filters in the ends of each edge of an UAHNEP collapsed into only one.

Fig. 1 Processors and filters in different types of AHNEPs

for all $x \in X_G$. Note that words which leave a node are eliminated from that node. If they cannot pass the input filter of any node, they are lost.

When changing by a communication step, each node processor $x \in X_G$ of an AHNEPFC sends one copy of each word it has to every node processor y connected to x, provided they can pass the filter of the edge between x and y. It keeps no copy of these words but receives all the words sent by any node processor z connected with x providing that they can pass the filter of the edge between x and z.

Formally, we say that the configuration C' is obtained in *one communication step* from configuration C, written as $C \vdash C'$, iff

$$C'(x) = \left(C(x) \setminus \left(\bigcup_{\{x,y\} \in E_G} \varphi^{\beta(\{x,y\})}(C(x), \mathcal{N}(\{x,y\})) \right) \right) \cup \left(\bigcup_{\{x,y\} \in E_G} \varphi^{\beta(\{x,y\})}(C(y), \mathcal{N}(\{x,y\})) \right)$$

for all $x \in X_G$. Note that a copy of a word remains in the sending node x only if it is not able to pass the filter of any edge connected to x.

Let Γ be an [U]AHNEP[FC], the computation of Γ on the input word $z \in V^*$ is a sequence of configurations $C_0^{(z)}, C_1^{(z)}, C_2^{(z)}, \ldots$, where $C_0^{(z)}$ is the initial configuration of Γ defined by $C_0^{(z)}(x_I) = \{z\}$ and $C_0^{(z)}(x) = \emptyset$ for all $x \in X_G,\ x \neq x_I,\ C_{2i}^{(z)} \Longrightarrow C_{2i+1}^{(z)}$ and $C_{2i+1}^{(z)} \vdash C_{2i+2}^{(z)}$, for all $i \geq 0$. By the previous definitions, each configuration $C_i^{(z)}$ is uniquely determined by the configuration $C_{i-1}^{(z)}$. A computation as above is said to be an *accepting computation* if there exists a configuration in which the set of words existing in the output node x_O is non-empty. The *language accepted* by Γ is

$$L(\Gamma) = \{z \in V^* \mid \text{ the computation of } \Gamma \text{ on z is an accepting one}\}.$$

We define a time complexity measure on [U]AHNEP[FC]s. To this aim we consider an [U]AHNEP[FC] Γ with the input alphabet V. The *time complexity* of the halting computation $C_0^{(z)}, C_1^{(z)}, C_2^{(z)}, \ldots C_m^{(z)}$ of Γ on $z \in V^*$ is denoted by $time_\Gamma(z)$ and equals m. The time complexity of Γ is the function from $f : \mathbb{N}$ to $f : \mathbb{N}$, $Time_\Gamma(n) = max\{time_\Gamma(z) \mid z \in L(\Gamma), |z| = n\}$. In other words, $Time_\Gamma(n)$ delivers the maximal number of computational steps done by Γ for accepting an input word of length n.

For a function $f : \mathbb{N} \longrightarrow \mathbb{N}$ we define:

$$\mathbf{Time}_{[U]AHNEP[FC]}(f(n)) = \{L \mid \text{ there exists an [U]AHNEP[FC]}\Gamma \text{ which accepts } L, \text{ and } n_0 \text{ such that } \forall n \geq n_0 (Time_\Gamma(n) \leq f(n))\}.$$

In the following sections we show that

$$\mathbf{Time}_{AHNEP}(f(n)) = \mathbf{Time}_{UAHNEP}(f(n)) = \mathbf{Time}_{AHNEPFC}(f(n))$$

for any function $f : \mathbb{N} \longrightarrow \mathbb{N}$. The proofs are based on direct simulations of each variant by the others and these simulations preserve the computational complexity.

3 Direct simulations between AHNEPs and UAHNEPs

As each UAHNEP can be immediately transformed into an AHNEP, we have:

Proposition 1 $\mathbf{Time}_{UAHNEP}(f(n)) \subseteq \mathbf{Time}_{AHNEP}(f(n))$ *for any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$.

The converse is also true, namely:

Proposition 2 $\mathbf{Time}_{AHNEP}(f(n)) \subseteq \mathbf{Time}_{UAHNEP}(f(n))$ *for any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$.

Proof Let $\Gamma = (V, U, G, \mathcal{N}, \alpha, \beta, x_1, x_n)$ be an AHNEP with the underlying graph $G = (X_G, E_G)$ and $X_G = \{x_1, x_2, \ldots, x_n\}$ for some $n \geq 1$. Let further $dom(M_{x_i}) = \{X \in U \mid X \to Y \in M_{x_i}\}$. We construct the UAHNEP $\Gamma' = \left(V, U', G', \mathcal{N}', \alpha', \beta', x_1^0, x_n^0\right)$, where

$$U' = U \cup U^{\blacktriangle} \cup U_{\blacktriangledown} \cup T, \quad U^{\blacktriangle} = \{X^{\blacktriangle} \mid X \in U\},$$
$$U_{\blacktriangledown} = \{X_{\blacktriangledown} \mid X \in U\}, \qquad T = \{\pounds\} \cup \{\$_i, \#_i, \xi_i, ¥_i \mid 1 \leq i \leq n\},$$

and the nodes and edges of G' are defined as in Tables 1, 2, 3, 4 and $\left\{x_1^0, x_1^{start}\right\} \in E_{G'}$.

Case 1. Let x_i, $1 \leq i \leq n-1$, be a substitution node. If $\beta(x_i) = (w)$, then the nodes defined in Table 2 belong to $X_{G'}$.
All the edges

- $\left\{x_1^{start}, x_1^1\right\}$,
- $\left\{x_i^{check-in}, x_i^1\right\}$ for $1 \leq i \leq n-1$,
- $\left\{x_i^1, x_i^2\right\}$, $\left\{x_i^1, x_i^2(Y)\right\}$ for $Y \in dom(M_{x_i})$ and $1 \leq i \leq n-1$,
- $\left\{x_i^1, x_i^{return_1}\right\}$, $\left\{x_i^1, x_i^{return_2}\right\}$ for $1 \leq i \leq n-1$,
- $\left\{x_i^2, x_i^3\right\}$ for $1 \leq i \leq n-1$,
- $\left\{x_i^3, x_i^{check-out}\right\}$, $\left\{x_i^3, x_i^{return_1}\right\}$, $\left\{x_i^3, x_i^{return_2}\right\}$, for $1 \leq i \leq n-1$,
- $\left\{x_i^{check-out}, x_i^{continue}\right\}$, $\left\{x_i^{check-out}, x_i^2(Y)\right\}$ for $Y \in dom(M_{x_i})$ and $1 \leq i \leq n-1$,
- $\left\{x_i^{continue}, x_j^{check-in}\right\}$ for all $\{x_i, x_j\} \in E_G$, $1 \leq i \neq j \leq n-1$,
- $\left\{x_i^2(Y), x_i^{return_1}\right\}$, $\left\{x_i^2(Y), x_i^{return_2}\right\}$ for $Y \in dom(M_{x_i})$, and $1 \leq i \leq n-1$

belong to $E_{G'}$. For a better visualization we refer to Fig. 2.

If $\beta(x_i) = (s)$, then $x_i^{return_2}$ is replaced by $p \geq 1$ nodes of the form $x_i^{return_2^k}$, $1 \leq k \leq p$, where $PO_{x_i} = \{Z_1, Z_2, \ldots, Z_p\}$, $p \geq 1$. They are presented in Table 3. Furthermore, if $PO_{x_i} = \emptyset$, then $x_i^{return_2}$ is removed. Now an edge between x_i^1, x_i^3 and each node $x_i^2(Y)$, $Y \in dom(M_{x_i})$, on the one hand, and each node $x_i^{return_2^k}$, on the other hand, is added to $E_{G'}$.

Case 2. If x_i, $1 \leq i \leq n-1$, is an insertion node, then all the nodes, except for $x_i^2(Y)$, $Y \in dom(M_{x_i})$, defined in Tables 2 and 3 belong to $X_{G'}$. Also all the edges, except for those incident to $x_i^2(Y)$, $Y \in dom(M_{x_i})$, belong to $E_{G'}$.

Case 3. Let x_i, $1 \leq i \leq n-1$, be a deletion node. If $\beta(x_i) = (w)$, then the following modifications regarding the nodes defined in Table 2 have to be done:

Table 1 Two initial nodes of the UAHNEP simulating an AHNEP

Node	M	P	F	α'	β'
x_1^0	$\{\varepsilon \to \pounds\}$	$\{\pounds\}$	$(U_{\blacktriangledown} \cup U^{\blacktriangle} \cup T) \setminus \{\pounds\}$	$*$	(s)
x_1^{start}	$\{\pounds \to \#_1\}$	$\emptyset$	$(U_{\blacktriangledown} \cup U^{\blacktriangle} \cup T) \setminus \{\pounds, \#_1\}$	$*$	(s)

Table 2 Derived UAHNEP nodes for simulating an AHNEP node

Node	M	P	F	α'	β'
$x_i^{check\text{-}in}$	$\{\$_i \to \#_i\}$	PI_{x_i}	$(FI_{x_i} \cup U_\blacktriangledown \cup T) \setminus \{\$_i, \#_i\}$	$*$	$\beta(x_i)$
x_i^1	$\{Y \to X_\blacktriangledown \mid Y \to X \in M_{x_i}\}$	$\{\#_i\}$	$\{\zeta_i\}$	$\alpha(x_i)$	(s)
x_i^2	$\{\#_i \to \zeta_i\}$	$U_\blacktriangledown$	$T \setminus \{\zeta_i, \#_i\}$	$*$	(w)
$x_i^2(Y)$, $Y \in dom(M_{x_i})$	$\{\#_i \to \zeta_i\}$	$\{\#_i\}$	$(\{Y\} \cup T \cup U_\blacktriangledown) \setminus \{\zeta_i, \#_i\}$	$*$	(w)
x_i^3	$\{X_\blacktriangledown \to X \mid X \in U\}$	$\{\zeta_i\}$	$\emptyset$	$*$	(s)
$x_i^{check\text{-}out}$	$\{\zeta_i \to ¥_i\}$	PO_{x_i}	$FO_{x_i} \cup U_\blacktriangledown \cup (T \setminus \{\zeta_i, ¥_i\})$	$*$	$\beta(x_i)$
$x_i^{continue}$	$¥_i \to \$_j \mid \{x_i, x_j\} \in E_G\}$	$\emptyset$	$T \setminus (\{¥_i\} \cup \{\$_j \mid \{x_i, x_j\} \in E_G\})]]$	$*$	(s)
$x_i^{return_1}$	$\{\zeta_i \to \#_i\}$	FO_{x_i}	$U_\blacktriangledown \cup \{\$_j, ¥_j \mid 1 \le j \le n\}$	$*$	(w)
$x_i^{return_2}$	$\{\zeta_i \to \#_i\}$	$\emptyset$	$PO_{x_i} \cup \{\$_j, ¥_j \mid 1 \le j \le n\} \cup U_\blacktriangledown$	$*$	(s)

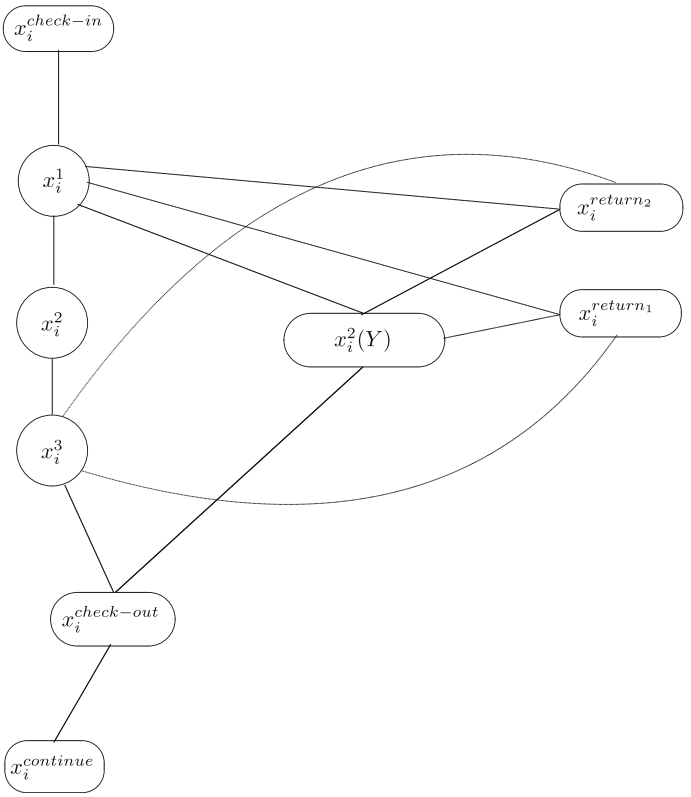


Fig. 2 Overall structure of a subnetwork of the UAHNEP simulating a substitution node of the AHNEP

Table 3 Return nodes for the case of strong filtering

Node	M	P	F	α'	β'
$x_i^{return_2^k}$	$\{\zeta_i \to \#_i\}$	$PO_{x_i} \setminus \{Z_k\}$	$\{Z_k\} \cup U_\blacktriangledown \cup \{\$_j, ¥_j \mid 1 \le j \le n\}$	$*$	(w)

Table 4 Modified nodes for weak filtering

Node	M	P	F	α'	β'
x_i^1	$\{Y \to Y^{\blacktriangle} \mid Y \to \varepsilon \in M_{x_i}\}$	$\{\#_i\}$	$\{\zeta_i\}$	$*$	(s)
x_i^3	$\{Y^{\blacktriangle} \to \varepsilon \mid Y \in U\}$	$\{\zeta_i\}$	$\emptyset$	$\alpha(x_i)$	(s)

- $U_{\blacktriangledown}$ is replaced by $U^{\blacktriangle}$ in the filters of the nodes $x_i^{check-in}$, x_i^2, $x_i^{return_1}$, $x_i^{return_2}$ and $x_i^{check\text{-}out}$.
- Nodes x_i^1 and x_i^3 are replaced by the nodes in Table 4.

If $\beta(x_i) = (s)$, then $U_{\blacktriangledown}$ is replaced by $U^{\blacktriangle}$ in all filters of the nodes $x_i^{return_2^k}$ from Table 3. Furthermore, if $PO_{x_i} = \emptyset$, then $x_i^{return_2}$ is removed. All the corresponding edges in $E_{G'}$ are modified accordingly.

The output node x_n^0 is defined as follows: $M_{x_n^0} = M_{x_n}$, $P_{x_n^0} = PI_{x_n}$ and $F_{x_n^0} = FI_{x_n}$, with $\alpha'(x_n^0) = \alpha(x_n)$, $\beta'(x_n^0) = \beta(x_n)$. Finally, we add all the edges $\{x_i^{continue}, x_n^0\}$, $1 \le i \le n-1$, to $E_{G'}$.

We now analyze a computation of Γ' on an input word, say z. In the input node x_1^0, the symbol £ is inserted at all positions of z in different copies of z. All these words enter x_1^{start} where the symbol £ is replaced by $\#_1$. We start now a simulation of the first evolutionary step executed by Γ on the input word z. More generally, we may assume that the current word is $z = z_1\#_i z_2$, for some $1 \le i \le n-1$, placed in x_i^1, and $z_1 z_2 \in U^*$ is placed in x_i of X_G.

We suppose that x_i is a substitution node in Γ and $\beta(x_i) = (w)$. The analysis for the case when x_i is a substitution node in Γ and $\beta(x_i) = (s)$ is identical. In x_i^1, an occurrence of some symbol Y in $z_1\#_i z_2$ is replaced by $X_{\blacktriangledown}$ iff the same occurrence of Y in $z_1 z_2$ can be replaced by X in the node $x_i \in X_G$. Let $y_1\#_i y_2$ be one word obtained after a substitution rule has been applied to $z_1\#_i z_2$ in x_i^1. Note that if there exists $Y \in dom(M_{x_i})$ such that $Y \notin alph(z_1 z_2)$, then $z_1\#_i z_2$ may go out from x_i^1 and enter the following nodes:

- $x_i^{check-in}$, provided that $z_1 z_2$ can pass the input filter of x_i from Γ,
- $x_i^{return_1}$ and $x_i^{return_2}$, provided that $z_1 z_2$ cannot pass the output filter of x_i from Γ,
- $x_i^2(Y)$.

It is worth mentioning that in this case also $z_1 z_2$ can stay unchanged for one evolutionary step in $x_i \in G$. We analyze all cases. If $z_1\#_i z_2$ goes out from x_i^1 and enters $x_i^{check-in}$, then the "ping-pong" process between x_i^1 and $x_i^{check-in}$ may continue either forever or until the word contains a symbol from $U_{\blacktriangledown}$. If $z_1\#_i z_2$ goes out from x_i^1 and enters $x_i^{return_1}$ or $x_i^{return_2}$, then a similar "ping-pong" process takes place between $x_i^{return_1}$ or $x_i^{return_2}$ on the one hand, and x_i^1 and $x_i^2(Y)$, on the other hand. If $z_1\#_i z_2$ goes out from x_i^1 and enters $x_i^2(Y)$, then $\#_i$ is replaced by ζ_i; the new word $z_1\zeta_i z_2$ is simultaneously sent to all nodes $x_i^{check-out}$, $x_i^{return_1}$ and $x_i^{return_2}$. If it enters $x_i^{check-out}$, then ζ_i is replaced successively by ¥$_i$ (in $x_i^{check-out}$) and some \$$_j$ (in $x_i^{continue}$) such that $\{x_i, x_j\} \in E_G$. The obtained word $z_1\$_j z_2$ is sent to $x_j^{check-in}$. This situation resembles exactly the situation when $z_1 z_2$ is sent to x_j after staying unchanged for one evolutionary step in x_i. The case when $z_1|\zeta_i z_2$ enters any of the nodes $x_i^{return_1}$ and $x_i^{return_2}$ is considered above.

The only case remaining to be analyzed is when $z_1\#_i z_2$ is transformed into $y_1\#_i y_2$ (either $y_1 = z_1$ or $y_2 = z_2$) after applying a substitution rule in x_i^1. Then $y_1\#_i y_2$ is sent out. Its itinerary through the network is as follows: x_i^2, where $\#_i$ is replaced by ζ_i, then x_i^3, where $X_{\blacktriangledown}$ is replaced by X. After leaving x_i^3, the new word, say z', can enter either $x_i^{check-in}$ or at least one of $x_i^{return_1}$ and $x_i^{return_2}$. If it enters $x_i^{check-in}$ and consequently $x_i^{continue}$, then the following computational

step in Γ was simulated in Γ': z_1z_2 was transformed into z', by applying a substitution rule $Y \to X$ in $x_i \in X_G$ and z' was sent to all the nodes connected to x_i. The situation when z' enters one of the nodes $x_i^{return_1}$ and $x_i^{return_2}$ corresponds to the situation when z' cannot pass the output filter of x_i and a new evolutionary step in x_i is to be considered. Note that every such evolutionary step in Γ can be simulated by Γ' in 6 evolutionary steps and 5 communication steps.

The last case treated above works entirely well for an insertion node x_i in Γ no matter its filters type, while the whole discussion above is still valid for a deletion node x_i.

By all the above considerations, we conclude that $L(\Gamma) = L(\Gamma')$ and $Time_{\Gamma'}(n) \in \mathcal{O}(Time_{\Gamma}(n))$. □

4 Direct simulations between AHNEPs and AHNEPFCs

Proposition 3 $\mathbf{Time}_{AHNEP}(f(n)) \subseteq \mathbf{Time}_{AHNEPFC}(f(n))$ *for any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$.

Proof Let $\Gamma = (V, U, G, \mathcal{N}, \alpha, \beta, x_1, x_n)$ be an AHNEP with the underlying graph $G = (X_G, E_G)$ and $X_G = \{x_1, x_2, \ldots, x_n\}$ for some $n \geq 1$. We construct the AHNEPFC $\Gamma' = \left(V, U', G', \mathcal{R}, \mathcal{N}', \alpha', \beta', x_1^s, x_n^s\right)$, where

$$U' = U \cup \{X_i, X^d \mid X \in U, i \in \{1, \ldots, n\}\} \cup \{\$_i \mid i \in \{1, \ldots, n\}\} \cup \{\#, \$\}.$$

The nodes of the graph $G' = \left(X'_G, E'_G\right)$, the sets of rules associated with them and the way in which they are applied, as well as the edges of E'_G together with the filters associated with them are defined in the following.

First, for every pair of nodes x_i, x_j from X_G such that $\{x_i, x_j\} \in E_G$ we have the following nodes in Γ'

$$x_{i,j}^1 : \quad R\left(x_{i,j}^1\right) = \{\varepsilon \to \$\}, \quad \alpha'\left(x_{i,j}^1\right) = l,$$
$$x_{i,j}^2 : \quad R\left(x_{i,j}^2\right) = \{\$ \to \varepsilon\}, \quad \alpha'\left(x_{i,j}^2\right) = l,$$

and the following edges (the nodes x_i^f and x_j^s are defined as in Fig. 3):

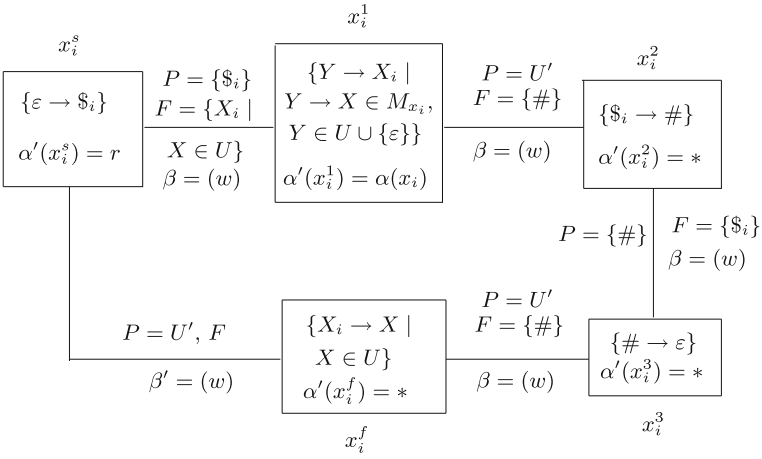


Fig. 3 The basic subnetwork used in the simulation of AHNEPs by AHENPFCs

$$\begin{aligned}
\left\{x_i^f, x_{i,j}^1\right\}: &\quad P = PO(x_i), \quad F = FO(x_i) \cup \{\$\}, \quad \beta' = \beta(x_i),\\
\left\{x_{i,j}^1, x_{i,j}^2\right\}: &\quad P = PI(x_j), \quad F = FI(x_j), \quad \beta' = (w),\\
\left\{x_{i,j}^2, x_j^s\right\}: &\quad P = PI(x_j), \quad F = FI(x_i) \cup \{\$, \$_j\}, \quad \beta' = \beta(x_j).
\end{aligned}$$

For each node x_i in Γ we add a subnetwork to Γ' according to the cases considered in the sequel.

Case 1. For an insertion or a substitution node $x_i \in X_G$ with weak filters Γ' contains the subnetwork depicted in Fig. 3.

The set of forbidden symbols F on the edge $\left\{x_i^f, x_i^s\right\}$ is defined by:

$$F = \begin{cases} FO(x_i) \cup PO(x_i) \cup \{\$_i\}, & \text{if } x_i \text{ is an insertion node} \\ PO(x_i) \cup \{\$_i\}, & \text{if } x_i \text{ is a substitution node} \end{cases}$$

Case 2. If x_i is an insertion or a substitution node with strong filters we just add the following nodes to the construction above:

$$x_i^{s,Z} : R\left(x_i^{s,Z}\right) = \{\varepsilon \to \$_i\}, \alpha'\left(x_i^{s,Z}\right) = *,$$

and the edges

$$\begin{aligned}
&\left\{x_i^{s,Z}, x_i^1\right\} : P = \{\$_i\}, F = \{X_i \mid X \in U\}, \beta' = (w),\\
&\left\{x_i^f, x_i^{s,Z}\right\} : P = U', F = \begin{cases} FO(x_i) \cup \{Z, \$_i\}, & \text{if } x_i \text{ is an insertion node} \\ \{Z, \$_i\}, & \text{if } x_i \text{ is a substitution node} \end{cases}
\end{aligned}$$

and $\beta' = (w)$, for all $Z \in PO(x_i)$.

Case 3. If $x_i \in X_G$ is a deletion node, then the construction in Case 1 is modified as follows:

- The way of applying the rules in node x_i^s is changed to l if $\alpha(x_i) = r$.
- Parameters of the node x_i^1 are now $R\left(x_i^1\right) = \left\{X \to X^d \mid X \to \varepsilon \in M_{x_i}\right\}$, $\alpha'\left(x_i^1\right) = *$.
- A new node is added: x_i^4 with $R\left(x_i^4\right) = \left\{X^d \to \varepsilon\right\}$, $\alpha'\left(x_i^4\right) = \alpha(x_i)$.
- Parameters of the node x_i^f are now $R\left(x_i^f\right) = \left\{X^d \to X \mid X \in U\right\}$, $\alpha'\left(x_i^f\right) = *$.

In this case, the edges are:

$$\begin{aligned}
\left\{x_i^s, x_i^1\right\}: &\quad P = \{\$_i\}, \quad F = \left\{X^d \mid X \in U\right\}, \quad \beta' = (w),\\
\left\{x_i^1, x_i^2\right\}: &\quad P = U', \quad F = \{\#\}, \quad \beta' = (w),\\
\left\{x_i^2, x_i^3\right\}: &\quad P = \{\#\}, \quad F = \{\$_i\}, \quad \beta' = (w),\\
\left\{x_i^3, x_i^4\right\}: &\quad P = U', \quad F = \{\#\}, \quad \beta' = (w),\\
\left\{x_i^4, x_i^s\right\}: &\quad P = U', \quad F = \emptyset, \quad \beta' = (w),\\
\left\{x_i^f, x_i^s\right\}: &\quad P = FO(x_i), \quad F = \{\$_i\}, \quad \beta' = (w).
\end{aligned}$$

Let us follow a computation of Γ' on the input word $w \in V^*$. Let us assume that w lies in the input node x_1^s of Γ'. In the same time, we assume that w is found in x_1, the input node of Γ. Inductively, we may assume that a word w is found in some x_i, a node of Γ, as well as in x_i^s from Γ'.

In the sequel we consider two cases: x_i is a substitution or a deletion node. Since the reasoning for an insertion node is pretty similar to that for a substitution node, it is left to the reader. Let x_i be a substitution node, where a rule $Y \to X$ is applied to w producing either w_1Xw_2, if $w = w_1Yw_2$ or w, if w doesn't contain Y. Here is the first difference with respect to an insertion node where every rule that could be applied is actually applied. In Γ', the word w is processed as follows. First w becomes $w\$_i$ in x_i^s, then it can enter x_i^1 only. Here it may become $w_1X_iw_2\$_i$, if $w = w_1Yw_2$, or it is left unchanged. Further, $w\$_i$ can go back to x_i^s, where another $\$_i$ symbol is added to its righthand end. Then it returns to x_i^1 and the situation above is repeated. When x_i is an insertion node, then this "ping-pong" process cannot happen. On the other hand, $w_1X_iw_2\$_i$ enters x_i^2. It is worth mentioning that any word arriving in x_i^2 contains at most one occurrence of X_i for some $X \in U$. In x_i^2, all the symbols $\$_i$ are replaced by # which is to be deleted in x_i^3. Finally, the current word enters x_i^f where the symbol X_i, if present, is rewritten into X. Thus, in node x_i^f we have obtained the word w_1Xw_2, if $w = w_1Yw_2$, or w if w doesn't contain Y; all the other words that may be obtained during these five steps either lead to the same word in x_i^f or have no effect on the rest of the computation.

The second case to be considered is when x_i is a deletion node containing a rule $Y \to \varepsilon$; we will assume that this node is a left deletion node, all the other cases being treated similarly. In this node, the word w is transformed into w', if $w = Yw'$, or is left unchanged, otherwise. In Γ' the word is processed as follows. First, in x_1^s a symbol $\$_i$ is inserted in the rightmost end of the word. Then the word enters x_i^1, where it is transformed into $w_1Y^dw_2\$_i$ (if $w = w_1Yw_2$, for all the possible $w_1, w_2 \in U^*$) or $w\$_i$ (if Y doesn't occur in w). After this step, $w\$_i$ goes back to x_i^s, where another $\$_i$ symbol is added. It then returns to x_i^1 and the situation above is repeated. On the other hand, all words $w_1Y^dw_2\$_i$ enter x_i^2. Again, we mention that every word arriving in x_i^2 contains at most one occurrence of X^d for some $X \in U$. Here all the symbols $\$_i$ are replaced by #. The words can now enter x_i^3 only, where all the symbols # are deleted. Further they go to node x_i^4, where the symbol X^d is deleted, provided that it is the leftmost one. Otherwise, they are left unchanged. Then each obtained word goes to x_i^f, where it is transformed back into w, if the symbol X^d was not deleted in the previous node, or is left unchanged. If the word still contains X^d, then it goes back to node x_i^4 and the above considerations can be applied again. If the word obtained doesn't contain any X^d, then it is either w', where $w = Yw'$, or w; all the other words that we may obtain during these six steps either lead to the same word in x_i^f or have no effect on the rest of the computation.

In conclusion, if $w \in U^*$ is a word in the nodes x_i of Γ and x_i^s of Γ', then we can obtain $w' \in U^*$ in one processing step of Γ if and only if we can obtain w' in the node x_i^f of Γ' in 5 processing steps (if x_i is an insertion or substitution node) or in 6 processing steps (if x_i is a deletion node). At this point we note that w' can leave x_i and enter x_j in Γ if and only if w' can leave x_i^f and enters x_j^s via the nodes $x_{i,j}^1$ and $x_{i,j}^2$. If w' can leave x_i but cannot enter x_j in Γ, then it is trapped in $x_{i,j}^1$ in Γ'. Finally, if w' cannot leave node x_i, then it is resent by x_i^f to x_i^s (in the case of deletion nodes, and insertion and substitution nodes with weak filters) or to the nodes $x_i^{s,Z}$, for all $Z \in PO(x_i)$ (in the case of insertion and substitution nodes with strong filters); from this point the process described above is repeated, with the only difference that in the case of insertion and substitution nodes with strong filters, the role of node x_i^s is played by the nodes $x_i^{s,Z}$.

From the above considerations, it follows that Γ' simulates in at most 6 processing steps and 5 communication steps a processing step of Γ, and in another 2 processing steps and 3 communication steps a communication step of Γ. We conclude that $L(\Gamma) = L(\Gamma')$ and $Time_{\Gamma'}(n) \in \mathcal{O}(Time_{\Gamma}(n))$. □

The converse of the previous proposition holds.

Proposition 4 $\mathbf{Time}_{AHNEPFC}(f(n)) \subseteq \mathbf{Time}_{AHNEP}(f(n))$ *for any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$.

Proof Let $\Gamma = (V, U, G, \mathcal{R}, \mathcal{N}, \alpha, \beta, x_1, x_n)$ be an AHNEPFC with $G = (X_G, E_G)$, X_G having n nodes $x_1, x_2, \ldots, x_n$. We construct the AHNEP $\Gamma' = \left(V, U', G', \mathcal{N}', \alpha', \beta', x_I, x_O\right)$, where

$$
\begin{aligned}
U' &= V \cup X \cup \{Y\}, X = \{X_{i,j} \mid 1 \leq i \neq j \leq n, i \neq n, \text{ and } \{x_i, x_j\} \in E_G\} \\
G' &= \left(X'_G, E'_G\right), \\
X'_G &= \{x_I, x_O\} \cup \left\{x_{i,j}, x'_{i,j} \mid 1 \leq i \neq j \leq n, i \neq n, \text{ and } \{x_i, x_j\} \in E_G\right\}, \\
E'_G &= \{\{x_I, x_{1,i}\} \mid 2 \leq i \leq n\} \cup \left\{\left\{x_{i,j}, x'_{i,j}\right\} \mid 1 \leq i \neq j \leq n, i \neq n\right\} \\
&\quad \cup \left\{\left\{x'_{i,j}, x_{j,k}\right\} \mid 1 \leq i \neq j \leq n, 1 \leq j \neq k \leq n\right\} \\
&\quad \cup \left\{\left\{x'_{i,n}, x_O\right\} \mid 1 \leq i \leq n-1\right\},
\end{aligned}
$$

and the other parameters defined as follows:

- node x_I : $M = \{\varepsilon \rightarrow X_{1,i} \mid 2 \leq i \leq n\}$,
  - $PI = V,\ FI = X,\ PO = X,\ FO = \emptyset$,
  - $\alpha' = *,\ \beta' = (w)$.
- nodes $x_{i,j}, 1 \leq i \neq j \leq n, i \neq n$: $M = \mathcal{R}(x_i)$,
  - $PI = \{X_{i,j}\},\ FI = X \setminus \{X_{i,j}\},\ PO = P_{\{x_i,x_j\}},\ FO = F_{\{x_i,x_j\}}$,
  - $\alpha' = \alpha(x_i),\ \beta' = \beta(\{x_i, x_j\})$.
- nodes $x'_{i,j}$, $1 \leq i \neq j \leq n, i \neq n$:
  - $M = \begin{cases} \{X_{i,j} \rightarrow X_{j,k} \mid 1 \leq k \leq n, k \neq j\}, & \text{if } j < n \\ \{X_{i,j} \rightarrow Y\}, & \text{if } j = n \end{cases}$
  - $PI = \{X_{i,j}\},\ FI = X \setminus \{X_{i,j}\},\ PO = (X \cup \{Y\}) \setminus \{X_{i,j}\},\ FO = \emptyset$,
  - $\alpha' = *, \beta' = (w)$.
- node x_O : $M = \emptyset,\ PI = \{Y\},\ FI = \emptyset,\ PO = \emptyset,\ FO = \emptyset$,
  - $\alpha' = *, \beta^* = (s)$.

Any computation in Γ' on an input word $w \in V^+$ produces in x_I all words $w_1 X_{1,i} w_2$ with $w_1, w_2 \in V^*$ such that $w = w_1 w_2$ and $2 \leq i \leq n$ provided that $\{x_1, x_i\} \in E_G$. Each word containing $X_{1,i}$ enters $x_{1,i}$. In a more general setting, we assume that a word $y_1 X_{i,j} y_2,\ y_1, y_2 \in V^*$, enters $x_{i,j}$ at a given step of the computation of Γ' on w. This means that $y = y_1 y_2$ enters x_i at a given step of the computation of Γ on w. Let y be transformed into $z = z_1 z_2$ in node x_i and z can pass the filter on the edge between x_i and x_j. Let us further assume that y_p is transformed into $z_p,\ p = 1, 2$. This easily implies that $y_1 X_{i,j} y_2$ is transformed into $z_1 X_{i,j} z_2$ in node $x_{i,j}$ and $z_1 X_{i,j} z_2$ can pass the output filter of $x_{i,j}$. Note that the converse is also true. Now, $z_1 X_{i,j} z_2,\ j \neq n$, enters $x'_{i,j}$ where all words $z_1 X_{j,k} z_2$, with $1 \leq k \neq j \leq n$ and $\{x_j, x_k\} \in E_G$, are produced. Each word $z_1 X_{j,k} z_2$ enters $x_{j,k}$ and the process of simulating the computation in Γ resumes. On the other hand, $z_1 X_{i,n} z_2$ enters $x'_{i,n}$ where $X_{i,n}$ is replaced by Y. All words produced in $x'_{i,n}$, for some $1 \leq i \leq n-1$, enter x_O and the computation ends. Note that by the considerations

above, a word enters $x'_{i,n}$ if and only if a word from x_i was able to pass the filter on the edge between x_i and x_n in Γ.

Note that two consecutive steps (evolutionary and communication) in Γ are simulated by four steps (two evolutionary and two communication) in Γ'. Therefore, $L(\Gamma) = L(\Gamma')$ and $Time_{\Gamma'}(n) \in \mathcal{O}(Time_{\Gamma}(n))$. hold □

As a direct consequence of the results presented in the previous two sections we can state the main result of this paper:

Theorem 1

$$\mathbf{Time}_{AHNEP}(f(n)) = \mathbf{Time}_{UAHNEP}(f(n)) = \mathbf{Time}_{AHNEPFC}(f(n))$$

for any function $f : \mathbb{N} \longrightarrow \mathbb{N}$.

5 Simulations preserving complexity and the shape

The simulations presented above may lead to underlying graphs of the simulating networks that differ very much from the underlying graphs of the simulated networks. However, it looks like there is some form of duality between edges and nodes in the simulations. In network theory, some types of underlying graphs are common like *rings, stars, grids*, etc. Networks of evolutionary words processors, seen as language generating or accepting devices, having underlying graphs of these special forms have been considered in several papers, see, e.g., (Martín-Vide and Mitrana 2005) for an early survey. On the other hand, in almost all works reported so far AHNEPs and AHNEPFCs have complete underlying graphs.

Simulations preserving the type of the underlying graph of the simulated network (together with its computational complexity) represent, in our view, a matter of interest. We briefly discuss here the case of networks with a complete underlying graph. Starting from the observation that every AHNEPFC can be immediately transformed into an equivalent AHNEPFC with a complete underlying graph (the edges that are to be added are associated with filters which make them useless), we may immediately state that Proposition 3 holds for complete AHNEPs and AHNEPFCs as well.

Theorem 2 *If a language is accepted by a complete AHNEP in* $\mathcal{O}(f(n))$ *time, then it is accepted by a complete AHNEPFC in* $\mathcal{O}(f(n))$, *for any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$.

A stronger result that can be obtained directly from the proof of Proposition 2 is:

Theorem 3 *For any function* $f : \mathbb{N} \longrightarrow \mathbb{N}$, *a language is accepted by a complete AHNEP in* $\mathcal{O}(f(n))$ *time, if and only if it is accepted by a complete UAHNEP in* $\mathcal{O}(f(n))$.

Proof It suffices to complete the underlying graph G' of Γ' in the proof of Proposition 2. Apart from the considerations in that proof, it should be mentioned that the following situation may happen. Although a word, say z, enters at some moment x_i in Γ and cannot leave x_i unchanged, the corresponding word in Γ' can go out from $x_i^{check\text{-}in}$, enters some $x_i^2(Y)$, then $x_i^{check-out}$, $x_i^{continue}$, and finally $x_j^{check\text{-}in}$ for some $1 \leq j \neq i \leq n-1$. However, if this may happen, then z should be in x_i and x_j at the same time because Γ is a complete AHNEP. Therefore, this fact does not influence the accepted language of Γ'. □

Although it is true that every AHNEP is equivalent to a complete AHNEP (every Turing machine can be simulated by a complete AHNEP), we do not know a simple and direct

transformation as that for AHNEPFCs. Therefore, a direct simulation of complete AHNEPFC by complete AHNEP remains open. Furthermore, simulations preserving complexity as well as the type of the underlying graph remain to be further investigated.

6 Final remarks

The language decided by an AHNEP and AHNEPFC is defined in (Margenstern et al. 2005) and (Drăgoi et al. 2007), respectively. In a similar way the language decided by an UAHNEP can be defined. It is easy to note that the construction in the proof of Proposition 4 works for the decided languages as well. However, in the proofs of Propositions 2 and 3, Γ' does not detect the non-accepting halting computations of Γ, since configurations obtained in consecutive processing steps of Γ are not obtained here in consecutive processing steps. Thus Γ' doesn't necessarily decide the language decided by Γ. It is known from the simulation of Turing machines by AHNEPs and AHNEPFCs (Manea et al. 2007; Drăgoi and Manea 2008) that the languages decided by AHNEPs can be also decided by AHNEPFCs. By a similar construction to those from (Manea et al. 2007; Drăgoi and Manea 2008), one may prove that any recursive language can be decided by UAHNEPs. Under these circumstances, the following problem is legitimate: Can the constructions from the proofs of Propositions 3 and 2 be modified for a direct simulation of AHNEPs halting on every input? Finally, as the networks of evolutionary picture processors introduced in (Bottoni et al. 2009b) do not have insertion nodes, it would be of interest to find direct simulations for these devices. In other words, can we simulate networks having all nodes of just one or two types out of the three without introducing nodes of the missing type(s)?

Acknowledgements This work was supported by the Academy of Finland, projects 132727, 122426, and 108421. F. Manea acknowledges the support from the Alexander von Humboldt Foundation. J Sempere acknowledges the support from the Spanish Ministerio de Educación y Ciencia project TIN2007-60769.

References

Alhazov A, Bel Enguix G, Rogozhin Y (2009a) Obligatory hybridnetworks of evolutionary processors. In: International conference on agents and artificial intelligence (ICAART 2009), pp 613–618

Alhazov A, Csuhaj-Varj E, Martn-Vide C, Rogozhin Y (2009b) On the size of computationally complete hybrid networks ofevolutionaryprocessors. Theor Comput Sci 410:3188–3197

Bottoni P, Labella A, Manea F, Mitrana V, Sempere J (2009a) Filter position in networks of evolutionary processors does not matter: a direct proof. In: Proc. 15th international meeting on DNA computing and molecular programming. 8–11 June 2009, Fayetteville, Arkansas

Bottoni P, Labella A, Mitrana V, Sempere JM (2009b) Networks of evolutionary picture processors with filtered connections. In: Unconventional computation, 8th international conference (UC 2009), LNCS, vol 5715. Springer, Heidelberg, pp 70–84

Castellanos J, Martín-Vide C, Mitrana V, Sempere J (2001) Solving NP-complete problems with networks of evolutionary processors. In: International work-conference on artificial and natural neural networks (IWANN 2001), Lecture notes in computer science, vol 2084, pp 621–628

Csuhaj-Varjú E, Mitrana V (2000) Evolutionary systems: a language generating device inspired by evolving communities of cells. Acta Inform 36:913–926

Csuhaj-Varjú E, Salomaa A (1997) Networks of parallel language processors. In: New trends in formal languages, Lecture notes in computer science, vol 1218, pp 299–318

Dassow J, Truthe B (2007) On the power of networks of evolutionary processors. In: Machines, computations, and universality (MCU 2007), Lecture notes in computer science, vol 4667, pp 158–169

Drăgoi C, Manea F (2008) On the descriptional complexity of accepting networks of evolutionary processors with filtered connections. Int J Found Comput Sci 19:1113–1132
Drăgoi C, Manea F, Mitrana V (2007) Accepting networks of evolutionary processors with filtered connections. J Univers Comput Sci 13:1598–1614
Errico L, Jesshope C (1994) Towards a new architecture for symbolic processing. In: Artificial intelligence and information-control systems of robots '94, World Scientific, Singapore, pp 31–40
Fahlman SE, Hinton GE, Seijnowski TJ (1983) Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In: Proc. of the national conference on artificial intelligence, pp 109–113
Hillis W (1985) The connection machine. MIT Press, Cambridge
Manea F, Martin-Vide C, Mitrana V (2007) On the size complexity of universal accepting hybrid networks of evolutionary processors. Math Struct Comput Sci 17:753–771
Margenstern M, Mitrana V, Perez-Jimenez M (2005) Accepting hybrid networks of evolutionary systems. In: DNA based computers 10, Lecture notes in computer science, vol, pp 235–246
Martín-Vide C, Mitrana V (2005) Networks of evolutionary processors: results and perspectives. In: Molecular computational models: unconventional approaches. dea Group Publishing, Hershey, pp 78–114
Păun G (2000) Computing with membranes. J Comput Syst Sci 61:108–143
Păun G, Sântean L (1989) Parallel communicating grammar systems: the regular case. Ann Univ Bucharest Ser Matematica Inform 38:55–63
Rozenberg G, Salomaa A (eds) (1997) Handbook of formal languages. Springer–Verlag, Berlin
Sankoff D et al. (1992) Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome. Proc Natl Acad Sci USA 89:6575–6579