Information Processing Letters 56 (1995) 201-208

Information Processing Letters

A McCulloch-Pitts neural net to characterize even linear languages [1]

Jose M. Sempere *, Damián López

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46071, Valencia, Spain

Received 16 January 1995; revised 6 September 1995
Communicated by L. Boasson

Abstract

We present a McCulloch-Pitts neural net to recognize even linear languages. The language class is studied in order to define the net topology. Finally, the equivalence between the language class and the languages recognized by the net is proved.

Keywords: Formal languages; Abstract machines; Neural nets

1. Introduction

The Even Linear Language class introduced by Amar and Putzolu [1] is a subclass of the linear language class, a proper subclass of the context-free language class [3]. Typically, the definition of these classes has been done by providing grammars that can generate the language class. Another way of defining these classes has been done by providing abstract machines that can accept the classes (i.e. finite automata, pushdown automata, and so on).

This paper presents a neural net that accepts an Even Linear Language, that is, an abstract device which, given a string of an even linear language, accepts the string. Specifically, we take McCulloch-Pitts neural nets [4] to define this class. McCulloch-Pitts nets was introduced in 1943 in order to modelize neural processes. The characteristics of these processes, as McCulloch and Pitts modelized, are significantly related to finite state machine processes. In this sense, a formal proof of the equivalence between the languages accepted by McCulloch-Pitts nets and the regular language class [3] can be viewed in [4], and a recent study of its complexity can be viewed in [2]. What we present in this paper is a topology for these nets in order to be able to accept even linear languages. This paper is organized as follows: in the first place we formally define the Even Linear Language class and the McCulloch-Pitts nets. Other definitions are given to establish some results. Afterwards, we present an equivalence between these concepts based on the topology we propose. In this way, a constructive algorithm to obtain McCulloch-Pitts nets from even linear grammars is proposed. A theorem that formally proves the equivalence is provided too and we present some conclusions for future works.

* Corresponding author. Email: jsempere@dsic.upv.es.

[1] Work partially supported by the Spanish CICYT under grant TIC93-0633-CO2.

2. Basic concepts and notation

The main definitions we use about formal languages can be found in [3] and [4]. Given an alphabet Σ, we denote the set of all strings over this alphabet as Σ^*, the empty string as λ, and the length of a string w as $|w|$. A grammar is a tuple $G = (\Sigma, N, P, S)$, where Σ is an alphabet of terminal symbols, N is a set of nonterminal symbols, P is a set of productions of the grammar and S is the nonterminal start symbol (the axiom). Given a grammar G, a relation between strings of symbols can be defined. So, we will denote that a string β can be obtained from a string α, by making a production substitution in the grammar G, with $\alpha \Rightarrow_G \beta$. If the number of production substitutions is greater or less than one, then we denote it by $\alpha \Rightarrow_G^* \beta$.

Finally, given a set C, we will denote the power set of C by $\mathcal{P}(C)$.

Definition 1. An even linear grammar (ELG), $G = (\Sigma, N, P, S)$, is defined with the following productions:

- $A \to xBy$, where $A, B \in N$ and $x, y \in \Sigma^*$ with $|x| = |y|$,
- $A \to x$, where $A \in N$ and $x \in \Sigma^*$.

An even linear language L is the language generated by an even linear grammar G and it is denoted as $L = L(G)$. Every even linear grammar can be put in the following standard form as established in [1]:

- $A \to aBb$, where $A, B \in N$ and $a, b \in \Sigma$,
- $A \to a$, where $A \in N$ and $a \in \Sigma \cup \{\lambda\}$.

Definition 2. Given an even linear grammar in the standard form $G = (\Sigma, N, P, S)$, we will say that this grammar is *deterministic* if $A \to aBb \in P$ and $A \to aCb \in P$ imply that $B = C$.

From now on, we will deal with *deterministic* even linear grammars, given that their equivalence with the complete class can be found in [5]. Amar and Putzolu [1] proved that every even linear language is characterized by a *quasi-congruence* finite index relation. A *quasi-congruence* relation is similar to a *congruence* relation in the sense that, given two strings, its equivalence implies the equivalence of the strings obtained by including the previous strings in right and left equal length contexts. Another characterization of even linear languages can be found in [5], where Sempere and García defined a finite index relation by taking pairs of strings as the relation space. Finally, Takada [6] presents a reduction of the learning problem for even linear languages to the learning problem for regular languages.

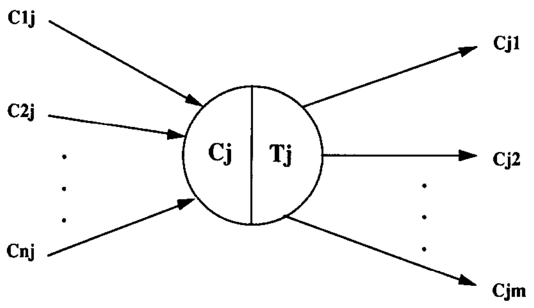

Fig. 1. A cell of a McCulloch-Pitts net.

Definition 3. A McCulloch-Pitts net is a tuple $M = (C, \Sigma, \delta, \mu, I, F)$, where C is a set of cells of the net, Σ is an input alphabet, $\delta : C \to \mathcal{P}(C)$ is a transition function, $\mu : \Sigma \to \mathcal{P}(C)$ is an external activation function, $I \in C$ is the initial cell and $F \in C$ is the final or acceptance cell.

An extension of the function δ to act over a set of cells $D \subseteq C$ is defined as

$$\delta(D) = \bigcup_{c \in D} \delta(c).$$

Every cell of the net has input lines, output lines and an activation integer value T_j, called its *threshold*. The input lines can be activated or not, and if the number of active input lines is greater or equal to the threshold value, then the cell and the output lines are activated. We take the value 1 for active states or lines and the value 0 for nonactive states. In Fig. 1, we can see a scheme of a McCulloch-Pitts cell. The activation of the cell C_j maintains the following function

$$F(C_j) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} C_{ij} \geqslant T_j, \\ 0 & \text{otherwise.} \end{cases}$$

The initial cell is active before processing a string and initially it is the only active cell. A string is accepted by the net iff the final cell is active after processing all its symbols. If we take a threshold value 2, the activation of a cell is reduced to receiving only two active input lines, and we can define a function φ, which can establish which cells of the net can be activated after the analysis of any input string. The definition of φ is based on the functions δ and μ as follows: $\varphi : \Sigma \times C \to \mathcal{P}(C)$, where $\varphi(a,c) = \mu(a) \cap \delta(c)$. An extension of this function over a set $D \subseteq C$ is defined as

$$\varphi(a,D) = \bigcup_{c\in D} \varphi(a,c)$$

and, finally, an extension of this function over strings is easy to do as follows,

$$\varphi(x_1x_2\dots x_n, D) = \varphi(x_2\dots x_n, Q),$$

where $x_1x_2\dots x_n \in \Sigma^*$, $D \subseteq C$ and $Q = \delta(D) \cap \mu(x_1)$.

We can define the language accepted by a McCulloch-Pitts net M, and we denote it by $L(M)$, as the following set:

$$L(M) = \{x \in \Sigma^* \mid F \in \varphi(x,I)\}.$$

In Fig. 2, we can see an example of a McCulloch-Pitts net. Note that the cells have threshold values of 2. The net has been obtained by taking the original finite automaton that accepts the language and applying the algorithm that transforms a finite automaton to a McCulloch-Pitts neural net [4].

3. Equivalence between even linear languages and languages accepted by McCulloch-Pitts nets

In this section we are going to establish the formal equivalence between the Even Linear Language class and the languages accepted by McCulloch-Pitts nets. In the first place, we will have to redefine the McCulloch-Pitts nets. Once we have the new nets, we will define an algorithm to construct a McCulloch-Pitts net from any deterministic even linear grammar. We will prove the formal equivalence between the language classes.

3.1. Extension of the McCulloch-Pitts nets

The generation of strings in any even linear grammar in standard form adds two terminal symbols in each generation step, that is, a string is generated from its extremes to its center. Thus, the parsing or analysis of any string with any abstract machine or grammar can be carried out following the generation process. In this sense, we will define a McCulloch-Pitts net by adapting the original net of Definition 3. With the new net, the analysis process of any string is performed as we have described above.

The new net has two subnets for analyzing the right and left extremes of the string. In addition, a subnet is needed to control the analysis of the center of the string and performs the acceptance or rejection of it. This net is formally defined as follows.

Definition 4. An Extended McCulloch-Pitts neural net (EMP) is a tuple $M = (C, \Sigma, \delta, \mu, I, F)$, where C is a set of cells and is organized in subsets R_c, L_c and F corresponding to the right, left and final subnets, Σ is an input alphabet, $\delta : C \to \mathcal{P}(C)$ is a transition function, $\mu : (\Sigma \cup \{\lambda\}) \to \mathcal{P}(C)$ is an external activation function, $I \subseteq C$ is a set of initial cells and $F \subseteq C$ is a set of final or acceptance cells.

If we take $\Sigma = \{a_1, \dots, a_n\}$ and $C = R_c \cup L_c \cup F$, with $R_c \cap L_c \cap F = \emptyset$, then the subnets are defined as $R_c = \{c_{1r}, \dots, c_{nr}\}$, $L_c = \{c_{1l}, \dots, c_{nl}\}$ and $F = \{c_{111f}, \dots, c_{(n+1)(n+1)(n+1)f}\}$. The set of initial cells is defined as $I = \{c_{1l}, c_{1r}\}$. The extension of the function δ to act over sets of cells is defined as in the previous section. Every cell of the net has a threshold value of 2, except the cells of subnet F that have a threshold value of 3. In Fig. 3, we can see the scheme of an EMP.

The definition of function μ is as follows:

$$\mu : (\Sigma \cup \{\lambda\}) \to \mathcal{P}(C),$$

$$\mu(x) = \begin{cases} \{c_{il}, c_{ir}, c_{jikf} \mid 1 \leqslant j,k \leqslant n+1\} \\ \quad \text{if } x = a_i, \\ \{c_{j(n+1)kf} \mid 1 \leqslant j,k \leqslant n+1\} \\ \quad \text{if } x = \lambda. \end{cases}$$

The definition of function δ has some restrictions by which cells of a subnet can affect cells of other subnets. So, we will impose the following two restrictions:

- $\delta(c_{il}) \supseteq \{c_{ijkf} \mid \forall 1 \leqslant j,k \leqslant n+1\}$,
- $\delta(c_{kr}) \supseteq \{c_{ijkf} \mid \forall 1 \leqslant i,j \leqslant n+1\}$.

The extension of δ to act over a set of cells D is like in the standard net, that is,

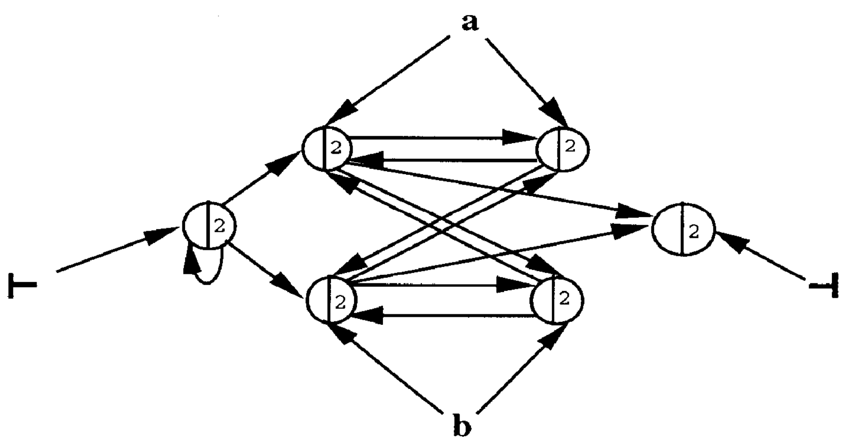

Fig. 2. A McCulloch-Pitts net to accept the language $\vdash(a+b)((a+b)(a+b))^*\dashv$.

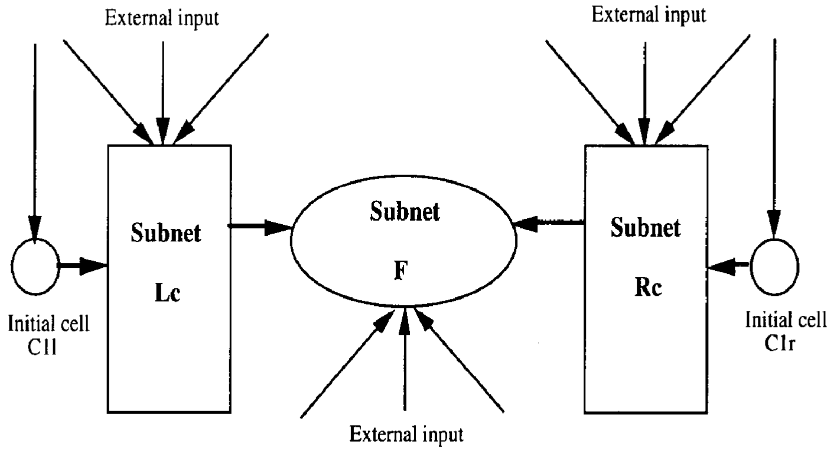

Fig. 3. An Extended McCulloch-Pitts net.

$$\delta(D) = \bigcup_{c\in D} \delta(c).$$

We will denote the set $\delta(D)\cap L_c$ as $\delta_l(D)$, $\delta(D)\cap R_c$ as $\delta_r(D)$, and $\delta(D)\cap F$ as $\delta_f(D)$. In the same sense, we will define the following sets:

$$\delta^l(D) = \bigcup_{c\in D\cap L_c} \delta(c), \qquad \delta^r(D) = \bigcup_{c\in D\cap R_c} \delta(c),$$

$$\delta^f(D) = \bigcup_{c\in D\cap F} \delta(c).$$

We can define the function φ, as in the standard net, in order to establish which cells can be activated after the analysis of a string. The function is defined as $\varphi : (\Sigma\cup\{\lambda\})\times C \to \mathcal{P}(C)$, where $\varphi(a,c) = \mu(a)\cap\delta^l(c)\cap\delta^r(c)$ with $a\in\Sigma\cup\{\lambda\}$. An extension to act over a set of cells D is

$$\varphi(a,D) = \bigcup_{c\in D} \varphi(a,c)$$

and finally, to act over strings is

$$\varphi(x_1x_2\dots x_n, D) = \varphi(x_2\dots x_{n-1}, Q),$$

where $x_1x_2\dots x_n\in\Sigma^*$ and $n\geqslant 2$, $D\subseteq C$ and $Q = (\delta_l(D)\cap\mu(x_1))\cup(\delta_r(D)\cap\mu(x_n))$.

We can define the language accepted by an EMP M, and we denote it by $L(M)$, as the following set:

$$L(M) = \{x\in\Sigma^* \mid F\cap\varphi(x,I) \neq \emptyset\}.$$

3.2. Constructive algorithm

In this section, we propose an algorithm which obtains an EMP from a deterministic ELG. Given that the string analysis in this net is the string generation process in the grammar, we need a method to start the analysis from the extremes of the string to its center. In this sense, we will use the initial cells of subnets L_c and R_c to start the analysis from the left and right extremes. This process is performed to arrive to the center of the string. At that moment, the final cells control which pair of symbols are the last to be analyzed. So, the subnet F controls the arrival to the center of the string. When this has happened, then two different situations can be performed. If the length of the analyzed string is even, then an external activation with λ is needed for the F net. Otherwise, the last symbol to be analyzed will be the external input to a final cell.

The proposed algorithm in Fig. 4 performs this task by analyzing every production of the grammar in two parts. That is, if $A \longrightarrow aBb \in P$ then the subnet L_c analyzes the part $A \longrightarrow aB$ and the subnet R_c analyzes the part $A \longrightarrow Bb$. Finally, if a nonterminal symbol has a terminal production, that is $A \longrightarrow a$, and it appears in the rightside of other production, that is $B \longrightarrow bAc$, then an external input will activate a final cell.

In order to make the analysis more clear, we will bind every string with two special symbols $\vdash$ and $\dashv$, which will be the only symbols that will excite the initial cells. After the analysis of these symbols, the rest of the string will be analyzed. The language that we associate to any even linear language with these special symbols is defined as follows.

Definition 5. Given a language $L \subseteq \Sigma^*$, we define the extreme bounded language of L as the set $\vdash L \dashv = \{\vdash u \dashv \mid u \in L\}$ with $\vdash, \dashv \notin \Sigma$.

Let us look at an example of application of the algorithm proposed in Fig. 4. Given the following grammar:

$$A_1 \longrightarrow a_1A_1a_2 \mid a_1A_2a_1 \mid \lambda$$
$$A_2 \longrightarrow a_1A_1a_1 \mid a_2A_2a_2 \mid a_1 \mid a_2$$

the associated EMP is the following one:

```
Input:    A deterministic ELG in standard form
          $G = (\Sigma, N, P, S)$ with $\Sigma = \{a_1, \ldots, a_n\}$,
          $N = \{A_1, \ldots, A_m\}$ and $S = A_1$.

Output:   An EMP $M = (C, \Sigma', \delta, \mu, I, F)$
          with $C = R_c \cup L_c \cup I \cup F$,
          such that $L(M) = \vdash L(G) \dashv$.

Method:   $\Sigma' = \{a_0, a_1, \ldots, a_n, a_{n+1}\}$
            with $a_0 = \vdash$ and $a_{n+1} = \dashv$
          $R_c = \{c_{11r}, \ldots, c_{nmr}, c_{(n+1)r}\}$
          $L_c = \{c_{0l}, c_{11l}, \ldots, c_{nml}\}$
          $F = \{c_{000f}, \ldots, c_{(n+1)(n+1)(n+1)f}\}$

          $I = \{c_{0l}, c_{(n+1)r}\}$

          $\mu(a_0) = \{c_{0l}\}$
          $\mu(a_{n+1}) = \{c_{(n+1)r}\}$
          $\mu(a_i) \supseteq \{c_{ikl}, c_{ikr} \mid 1 \leqslant k \leqslant m\}$ $1 \leqslant i \leqslant n$

          $\delta(c_{0l}) \supseteq \{c_{i1l} \mid 1 \leqslant i \leqslant n\} \cup \{c_{0l}\}$
          $\delta(c_{(n+1)r}) \supseteq \{c_{i1r} \mid 1 \leqslant i \leqslant n\} \cup \{c_{(n+1)r}\}$

          if $A_1 \longrightarrow \lambda \in P$ then
              $\delta(c_{0l}) \supseteq \{c_{0(n+1)(n+1)f}\}$
              $\delta(c_{(n+1)r}) \supseteq \{c_{0(n+1)(n+1)f}\}$
              $\mu(\lambda) \supseteq \{c_{0(n+1)(n+1)f}\}$

          $\forall A_1 \longrightarrow a_i \in P$ do
              $\delta(c_{0l}) \supseteq \{c_{0i(n+1)f}\}$
              $\delta(c_{(n+1)r}) \supseteq \{c_{0i(n+1)f}\}$
              $\mu(a_i) \supseteq \{c_{0i(n+1)f}\}$

          $\forall A_i \longrightarrow a_kA_ja_p \in P$ do
              $\delta(c_{kil}) \supseteq \{c_{hjl} \mid 1 \leqslant h \leqslant n\}$
              $\delta(c_{pir}) \supseteq \{c_{hjr} \mid 1 \leqslant h \leqslant n\}$

          $\forall A_i \longrightarrow a_kA_ja_p \in P$ do

              $\forall A_j \longrightarrow a_q \in P$ do
                  $\delta(c_{kil}) \supseteq \{c_{kqpf}\}$
                  $\delta(c_{pir}) \supseteq \{c_{kqpf}\}$
                  $\mu(a_q) \supseteq \{c_{kqpf}\}$

              if $A_j \longrightarrow \lambda \in P$ then
                  $\delta(c_{kil}) \supseteq \{c_{k(n+1)pf}\}$
                  $\delta(c_{pir}) \supseteq \{c_{k(n+1)pf}\}$
                  $\mu(\lambda) \supseteq \{c_{k(n+1)pf}\}$

endMethod.
```

Fig. 4. The proposed algorithm to obtain EMPs from ELGs.

$\Sigma' = \{\vdash, a_1, a_2, \dashv\}$
$R_c = \{c_{11r}, c_{12r}, c_{21r}, c_{22r}, c_{3r}\}$
$L_c = \{c_{0l}, c_{11l}, c_{12l}, c_{21l}, c_{22l}\}$
$F = \{c_{000f}, c_{001f}, \ldots, c_{332f}, c_{333f}\} I = \{c_{0l}, c_{3r}\}$

$\mu(\vdash) = \{c_{0l}\}$
$\mu(\dashv) = \{c_{3r}\}$
$\mu(a_1) \supseteq \{c_{11l}, c_{12l}, c_{11r}, c_{12r}\}$
$\mu(a_2) \supseteq \{c_{21l}, c_{22l}, c_{21r}, c_{22r}\}$
$\delta(c_{0l}) \supseteq \{c_{11l}, c_{21l}, c_{0l}\}$
$\delta(c_{3r}) \supseteq \{c_{11r}, c_{21r}, c_{3r}\}$

$A_1 \to \lambda$
$\delta(c_{0l}) \supseteq \{c_{033f}\}$
$\delta(c_{3r}) \supseteq \{c_{033f}\}$
$\mu(\lambda) \supseteq \{c_{033f}\}$

$A_1 \to a_1A_1a_2$
$\delta(c_{11l}) \supseteq \{c_{11l}, c_{21l}\}$
$\delta(c_{21r}) \supseteq \{c_{11r}, c_{21r}\}$

$A_1 \to a_1A_2a_1$
$\delta(c_{11l}) \supseteq \{c_{12l}, c_{22l}\}$
$\delta(c_{11r}) \supseteq \{c_{12r}, c_{22r}\}$

$A_2 \to a_1A_1a_1$
$\delta(c_{12l}) \supseteq \{c_{11l}, c_{21l}\}$
$\delta(c_{12r}) \supseteq \{c_{11r}, c_{21r}\}$

$A_2 \to a_2A_2a_2$
$\delta(c_{22l}) \supseteq \{c_{12l}, c_{22l}\}$
$\delta(c_{22r}) \supseteq \{c_{12r}, c_{22r}\}$

$A_1 \to a_1A_1a_2 \wedge A_1 \to \lambda$
$\delta(c_{11l}) \supseteq \{c_{132f}\}$
$\delta(c_{21r}) \supseteq \{c_{132f}\}$
$\mu(\lambda) \supseteq \{c_{132f}\}$

$A_1 \to a_1A_2a_1 \wedge A_2 \to a_1$
$\delta(c_{11l}) \supseteq \{c_{111f}\}$
$\delta(c_{11r}) \supseteq \{c_{111f}\}$
$\mu(a_1) \supseteq \{c_{111f}\}$

$A_1 \to a_1A_2a_1 \wedge A_2 \to a_2$
$\delta(c_{11l}) \supseteq \{c_{121f}\}$
$\delta(c_{11r}) \supseteq \{c_{121f}\}$
$\mu(a_2) \supseteq \{c_{121f}\}$

$A_2 \to a_1A_1a_1 \wedge A_1 \to \lambda$
$\delta(c_{12l}) \supseteq \{c_{131f}\}$
$\delta(c_{12r}) \supseteq \{c_{131f}\}$
$\mu(\lambda) \supseteq \{c_{131f}\}$

$A_2 \to a_2A_2a_2 \wedge A_2 \to a_1$
$\delta(c_{22l}) \supseteq \{c_{212f}\}$
$\delta(c_{22r}) \supseteq \{c_{212f}\}$
$\mu(a_1) \supseteq \{c_{212f}\}$

$A_2 \to a_2A_2a_2 \wedge A_2 \to a_2$
$\delta(c_{22l}) \supseteq \{c_{222f}\}$
$\delta(c_{22r}) \supseteq \{c_{222f}\}$
$\mu(a_2 \supseteq \{c_{222f}\}$

In Fig. 5, we can see the scheme of the EMP of the example.

After this, we will enunciate several results that prove the equivalence between the language accepted by the EMP that the algorithm outputs and the language generated by the input ELG.

Lemma 6. *Given an ELG G and its associated EMP M, if* $A_1 \Rightarrow_G^* xA_iy \Rightarrow_G uA_hv$, *then there exist cells* c_{kil}, c_{pir} *which will be active after processing the string* $\vdash uv \dashv$.

Proof. The result will be proved as an induction process over the number of derivation steps.

Induction base. Let us take $A_1 \Rightarrow_G a_kA_ia_p \Rightarrow_G a_ka_{k'}A_ha_{p'}a_p$ with $a_k, a_{k'}, a_p, a_{p'} \in \Sigma$. Then $A_1 \to a_kA_ia_p$ and $A_i \to a_{k'}A_ha_{p'} \in P$. So $\delta(c_{k1l}) \supseteq \{c_{jil} \mid 1 \leqslant j \leqslant n\}$ and $\delta(c_{p1r}) \supseteq \{c_{jir} \mid 1 \leqslant j \leqslant n\}$. By construction $\delta(c_{0l}) \supseteq \{c_{k1l} \mid 1 \leqslant k \leqslant n\} \cup \{c_{0l}\}$ and $\delta(c_{(n+1)r}) \supseteq \{c_{p1r} \mid 1 \leqslant p \leqslant n\} \cup \{c_{(n+1)r}\}$.

So, $\varphi(\vdash a_ka_{k'}a_{p'}a_p\dashv, I) = \varphi(a_ka_{k'}a_{p'}a_p, T) = \varphi(a_{k'}a_{p'}, Q) = W$, where
- $T = (\delta_l(I) \cap \mu(\vdash)) \cup (\delta_r(I) \cap \mu(\dashv)) = \{c_{0l}, c_{(n+1)r}\} = I$,
- $Q = (\delta_l(I) \cap \mu(a_k)) \cup (\delta_r(I) \cap \mu(a_p)) = \{c_{k1l}, c_{p1r}\}$,
- $W = (\delta_l(Q) \cap \mu(a_{k'})) \cup (\delta_r(Q) \cap \mu(a_{p'})) = \{c_{k'il}, c_{p'ir}\}$.

It is clear that in the case of $A_1 \Rightarrow_G a_{k'}A_ha_{p'}$, then $x = y = \lambda$ and $A_1 = A_i$. So, $\varphi(\vdash a_{k'}a_{p'}\dashv, I) = \varphi(a_{k'}a_{p'}, T) = W$, where
- $T = (\delta_l(I) \cap \mu(\vdash)) \cup (\delta_r(I) \cap \mu(\dashv)) = \{c_{0l}, c_{(n+1)r}\} = I$,
- $W = (\delta_l(I) \cap \mu(a_{k'})) \cup (\delta_r(I) \cap \mu(a_{p'})) = \{c_{k'1l}, c_{p'1r}\}$,

Induction hypothesis. $A_1 \Rightarrow_G^* xA_iy \Rightarrow_G uA_hv$ in $n-1$ derivation steps and we take, as hypothesis, that $\varphi(\vdash uv\dashv, I) \supseteq \{c_{kil}, c_{pir}\}$.

Induction Step. Now we take $A_1 \Rightarrow_G^* xA_iy \Rightarrow_G uA_hv$ in n derivation steps, so $A_1 \Rightarrow_G^* wA_qz \Rightarrow_G xA_iy \Rightarrow_G uA_hv$ with $u = wa_ka_{k'}$ and $v = a_{p'}a_pz$ and $A_q \to a_kA_ia_p \in P$ and $A_i \to a_{k'}A_ha_{p'} \in P$.

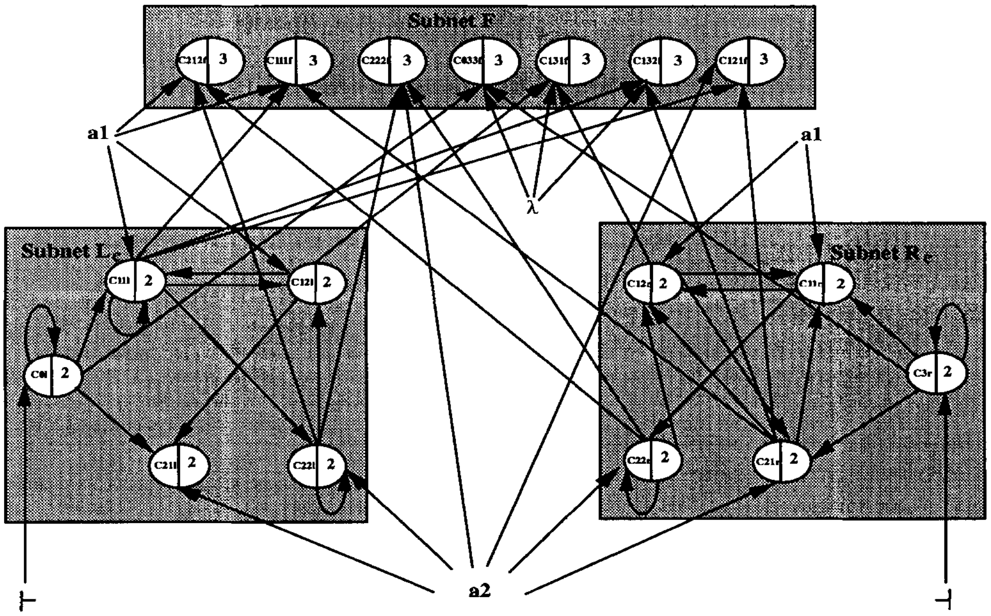

Fig. 5. An Extended McCulloch–Pitts net for $\vdash L(G)\dashv$.

Then $\varphi(\vdash w a_k a_{k'} a_{p'} a_p z \dashv, I) = \varphi(a_{k'} a_{p'}, T)$, where $\{c_{kql}, c_{pqr}\} \subseteq T$ by induction hypothesis. By construction, $\delta(c_{kql}) \supseteq \{c_{hil} \mid 1 \leqslant h \leqslant n\}$ and $\delta(c_{pqr}) \supseteq \{c_{hir} \mid 1 \leqslant h \leqslant n\}$.

So $\varphi(a_{k'} a_{p'}, T) = (\delta_l(T) \cap \mu(a_{k'})) \cup (\delta_r(T) \cap \mu(a_{p'}))$ and, finally, it contains $\{c_{k'il}, c_{p'ir}\}$. □

Theorem 7. *For every deterministic ELG G, there exists an EMP M that accepts the extreme bounded language of L(G).*

Proof. Once we have proved Lemma 6, we can easily prove that if any string belongs to $L(G)$ then its extreme bounded string belongs to $L(M)$. We will prove that if $A_1 \Rightarrow_G^* w$, then $\varphi(\vdash w \dashv, I) \cap F \neq \emptyset$. Let us study the following cases:

Case 1: $A_1 \Rightarrow_G \lambda$. Then $\delta(c_{0l}) \supseteq \{c_{0(n+1)(n+1)f}\}$, $\delta(c_{(n+1)r}) \supseteq \{c_{0(n+1)(n+1)f}\}$, and $\mu(\lambda) \supseteq \{c_{0(n+1)(n+1)f}\}$. So, $\varphi(\vdash\dashv, I) = \varphi(\lambda, I) = \mu(\lambda) \cap \delta^l(I) \cap \delta^r(I) = \{c_{0(n+1)(n+1)f}\}$ and $\vdash\dashv \in L(M)$.

Case 2: $A_1 \Rightarrow_G a_i$. Then $\delta(c_{0l}) \supseteq \{c_{0i(n+1)f}\}$, $\delta(c_{(n+1)r}) \supseteq \{c_{0i(n+1)f}\}$ and $\mu(a_i) \supseteq \{c_{0i(n+1)f}\}$. So $\varphi(\vdash a_i \dashv, I) = \varphi(a_i, I) = \mu(a_i) \cap \delta^l(I) \cap \delta^r(I) = \{c_{0i(n+1)f}\}$ and $\vdash a_i \dashv \in L(M)$.

Case 3: $A_1 \Rightarrow_G^* w$. We can study two different cases depending on the size of w.

- w has an even length.
  Then $A_1 \Rightarrow_G^* x A_i y \Rightarrow_G x a_k A_j a_p y \Rightarrow_G x a_k a_p y = w$. So $A_i \to a_k A_j a_p \in P$ and $A_j \to \lambda \in P$.
  As we have proved before, $\varphi(\vdash x a_k a_p y \dashv, I) = \mu(\lambda) \cap \delta^l(T) \cap \delta^r(T)$ and $T \supseteq \{c_{kil}, c_{pir}\}$. In this case, $\delta(c_{kil}) \supseteq \{c_{k(n+1)pf}\}$, $\delta(c_{pir}) \supseteq \{c_{k(n+1)pf}\}$ and $\mu(\lambda) \supseteq \{c_{k(n+1)pf}\}$. So, $\mu(\lambda) \cap \delta^l(T) \cap \delta^r(T) \supseteq \{c_{k(n+1)pf}\}$ and $\vdash w \dashv \in L(M)$.
- w has an odd length.
  Then $A_1 \Rightarrow_G^* x A_i y \Rightarrow_G x a_k A_j a_p y \Rightarrow_G x a_k a_q a_p y = w$. So $A_i \to a_k A_j a_p$ and $A_j \to a_q \in P$.
  As we have proved before, $\varphi(\vdash x a_k a_q a_p y \dashv, I) = \varphi(a_q, T) = \mu(a_q) \cap \delta^l(T) \cap \delta^r(T)$ and $T \supseteq \{c_{kil}, c_{pir}\}$. In this case, $\delta(c_{kil}) \supseteq \{c_{kqpf}\}$, $\delta(c_{pir}) \supseteq \{c_{kqpf}\}$ and $\mu(a_q) \supseteq \{c_{kqpf}\}$. So, $\mu(a_q) \cap \delta^l(T) \cap \delta^r(T) \supseteq \{c_{kqpf}\}$ and $\vdash w \dashv \in L(M)$. □

Let us take notice that the converse results of Lemma 6 and Theorem 7 are easy to prove by following the induction process that we have carried out before. So, we can conclude that the language class accepted by EMPs, with the cell connection restrictions of the proposed algorithm, is the Even Linear Language class.

4. Conclusions

We have presented a neural net to recognize a formal language class. We think that McCulloch–Pitts neural net can be extended to recognize other classes by making new topologies. In this sense, future works can be directed in the following two ways:

- To study language classes in order to define finite index relations that imply new topologies. This work is of great interest in other areas like *computational learning* and *pattern recognition*, given that dealing with finite index families is easier than nonfinite families in these two areas.
- To propose new topologies for McCulloch–Pitts nets and to characterize the families that these new nets accept. The computability of this net has special advantages like its great parallelism as opposed to other devices like finite automata or pushdown automata. So, we think that this device is better for tasks like *pattern matching* and *string accepting*.

Our interest in McCulloch–Pitts nets has other goals such as establishing a formal equivalence between finite state machines and other neural nets like perceptron, Bolzmann Machines or recurrent neural nets. We think that the problem of computational learning with these nets has disadvantages like the initial structure of the net. If the formal equivalence could be proved, then learning with these devices would avoid the great amount of a priori information currently needed.

Acknowledgements

We would like to thank the anonymous referee for helpful suggestions and sharp remarks to our original work.

References

[1] V. Amar and G. Putzolu, On a family of linear grammars, *Inform. and Control* **7** (1964) 283–291.

[2] N. Alon, A. Dewdney and T. Ott, Efficient simulation of finite automata by neural nets, *J. ACM* **38** (1991) 495–514.

[3] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).

[4] M. Minsky, *Computation. Finite and Infinite Machines* (Prentice-Hall, Englewood, NJ, 1967).

[5] J. Sempere and P. Garcia, A characterization of even linear languages and its application to the learning problem, in: Lecture Notes in Artificial Intelligence in: *Proc. 2nd Internat. Coll. on Grammatical Inference and Applications*, ICGI-94 (Springer, Berlin, 1994) 38–44.

[6] Y. Takada, Grammatical inference of even linear languages based on control sets, *Inform. Process. Lett.* **28** (1988) 193–199.