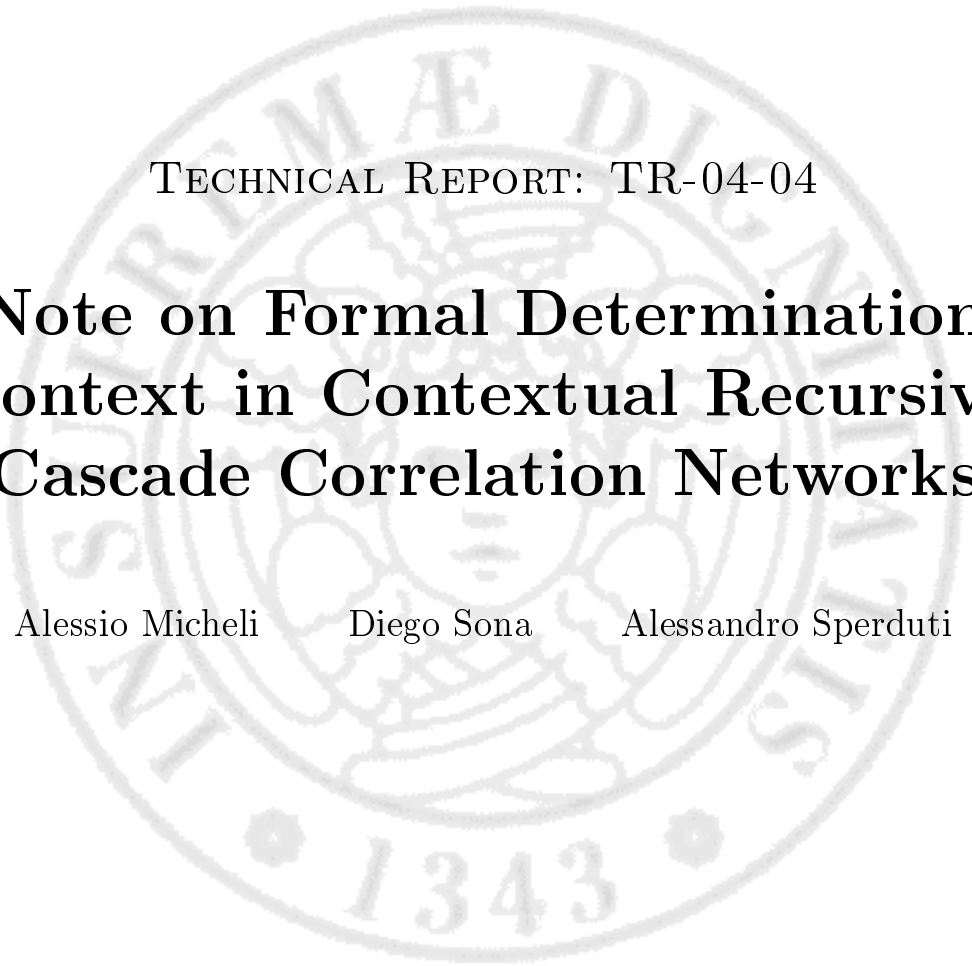
Università di Pisa

Dipartimento di Informatica



Technical Report: TR-04-04

A Note on Formal Determination of Context in Contextual Recursive Cascade Correlation Networks

Alessio Micheli Diego Sona Alessandro Sperduti

January 26, 2004

ADDRESS: Via Buonarroti 2, 56127 Pisa, Italy. TEL: +39 050 2212700 FAX: +39 050 2212726

A Note on Formal Determination of Context in Contextual Recursive Cascade Correlation Networks

Alessio Micheli* Diego Sona† Alessandro Sperduti‡

January 26, 2004

Abstract

In this note we consider the Contextual Recursive Cascade Correlation model (CRCC), which is able to learn contextual mappings in structured domains. Specifically, we propose a formal characterization of the "context window", i.e., given a state variable, the "context window" is the set of state variables directly or indirectly contributing to its determination. On the basis of this definition, a formal expression describing the "context windows" for the CRCC is derived. The resulting expression is very compact and describes the context window as a function of the state variables in the model, and some topological properties of the structures in input to the model. Moreover, the expression elucidates how the shape of the context window is modified by the introduction of new state variables, i.e., hidden units. For comparison, a similar expression is also derived for the non-contextual Recursive Cascade Correlation model.

1 Introduction

In recent years, neural network models for processing of structured data have been developed and studied (see for example [7, 3, 8, 2]). Almost all these models assume causality and stationarity. Unfortunately, in several real-world applications, such as language understanding and DNA and proteins analysis, these assumptions do not hold. For this reason some authors are starting to propose models that try to exploit contextual information within recurrent and recursive models. Examples of these models for the processing of sequences are described in [1, 4, 9], while in [5, 6] the *Contextual Recursive Cascade Correlation* model (CRCC), able to process structured data, i.e., directed positional acyclic graphs, in an incremental and contextual fashion has been proposed.

*Dip. di Informatica, Università di Pisa, Italy. micheli@di.unipi.it

†SRA division, ITC-IRST, Trento, Italy. sona@itc.it

‡Dip. di Matematica Pura ed Applicata, Università di Padova, Italy. sperduti@math.unipd.it

In CRCC, which combines both structured data and contextual information, the adoption of an incremental learning strategy, such as Cascade Correlation, makes it hard to relate the insertion of new state variables in the model (i.e., hidden units) with the "shape" of the contextual information used during learning and computation.

The aim of this paper is to formally elucidate how the "shape" of the contextual information is determined by the insertion of new state variables. Specifically, here we abstract from any specific neural realization of the CRCC model, and propose to define the "context window" of a given state variable in the model as the set of (already defined) state variables that contribute, directly or indirectly, to the determination of the considered state variable. Starting from this definition, we are able to define a compact expression which describes the context window as a function of the state variables in the model, and the sets of parents and children belonging to the considered vertex in the current input structure. This is obtained also by resorting to the definition of a couple of operators, the *"descendants"* and *"dot"* operators, which contribute to capture the relevant information needed to understand and to concisely describe the contextual window. Informally, given a set of vertexes of the input structure, the descendants operator recovers all the descendants of these vertexes, while the "dot" operator returns the set of state variables associated to a set of vertexes given as argument. The composition of these operators, in combination with the sets of parents and children of a given vertex, allows the formal derivation of the desired expression. The same machinery can be trivially used to define the "context window" for the non-contextual Recursive Cascade Correlation (RCC) [7].

Besides to be of interest by themselves, the formal results obtained in this paper, can be exploited to characterize the computational power of CRCC versus other models, such as RCC. This is done in [6].

2 Structured Domains

Given a DAG (directed acyclic graph) $\mathcal{D}$ we denote the vertexes set with $\text{vert}(\mathcal{D})$ and the edges set with $\text{edg}(\mathcal{D})$, where the edges are ordered couples of vertexes. Given a vertex $v \in \text{vert}(\mathcal{D})$ we denote the set of edges entering and leaving from v with $\text{edg}(v)$. Moreover, we define:

- $\text{out_set}(v) = \{u | (v, u) \in \text{edg}(v)\}$;
- $\text{in_set}(v) = \{u | (u, v) \in \text{edg}(v)\}$;
- $\text{out_deg}(v) = |\text{out_set}(v)|$ (outdegree);
- $\text{in_deg}(v) = |\text{in_set}(v)|$ (indegree).

We assume that instances in the learning domain are DPAGs (directed positional acyclic graphs) with bounded outdegree *out* and indegree *in*. An instance of DPAG is a DAG where we assume that for each vertex $v \in \text{vert}(\mathcal{D})$, two injective functions $P_v : \text{edg}(v) \rightarrow [1, 2, ..., in]$ and $S_v : \text{edg}(v) \rightarrow [1, 2, ..., out]$ are defined on the edges entering and leaving from v. In this way, a positional index is assigned to each entering and leaving edge from a node v. Moreover, we define $\forall u \in \text{vert}(\mathcal{D})$

$$\forall j \in [1, \ldots, in] \quad \text{in_set}_j(u) = \begin{cases} v & \text{if } \exists v \in \text{vert}(\mathcal{D}) \mid P_u((v,u)) = j \\ nil & \text{otherwise} \end{cases}$$

$$\forall j \in [1, \ldots, out] \quad \text{out_set}_j(u) = \begin{cases} v & \text{if } \exists v \in \text{vert}(\mathcal{D}) \mid S_u((u,v)) = j \\ nil & \text{otherwise} \end{cases}$$

We shall require the DPAGs to possess a supersource[1], i.e. a vertex $s \in \text{vert}(\mathcal{D})$ such that every vertex in $\text{vert}(\mathcal{D})$ can be reached by a directed path starting from s. Moreover, vertexes are labeled by vectors of real numbers which either represent numerical or categorical variables. With $\boldsymbol{l}(v)$ we denote the label associated to v, and with $l_i(v)$ the i-th element of the label.

3 Processing of Structured Data by Recursive Models

Recursive Neural Networks [3, 8] possess, in principle, the ability to memorize "past" information to perform structural mappings. The state transition function $\tau()$ and the output function $g()$, in this case, prescribe how the state variable, or better the state vector $\boldsymbol{x}(v)$ associated to a vertex v is used to obtain the state and output vectors corresponding to other vertexes, respectively. Specifically, given a state vector $\boldsymbol{x}(v) \equiv [x_1(v), \ldots, x_m(v)]^t$, we define the extended shift operators $q_j^{-1} x_k(v) \equiv x_k(\text{out_set}_j(v))$ and $q_j^{+1} x_k(v) \equiv x_k(\text{in_set}_j(v))$. If $\text{out_set}_j(v) = nil$ then $q_j^{-1} x_k(v) \equiv x_k(nil) \equiv x_0$, the null state[2]. Similarly, if $\text{in_set}_j(v) = nil$ then $q_j^{+1} x_k(v) \equiv x_0$. Moreover, we define

$$\boldsymbol{q}^{-1} x_k(v) = \begin{bmatrix} q_1^{-1} x_k(v) \\ \vdots \\ q_{out}^{-1} x_k(v) \end{bmatrix}, \tag{1}$$

$$\boldsymbol{q}^{+1} x_k(v) = \begin{bmatrix} q_1^{+1} x_k(v) \\ \vdots \\ q_{in}^{+1} x_k(v) \end{bmatrix}, \tag{2}$$

[1] If no supersource is present, a new vertex connected to all the vertexes of the graph with null *indegree* can be added.

[2] In this paper, we assume $x_0 = 0$. Other assumptions can be considered, e.g. having a different null state for each j and missing entering/leaving edge.

and, given $e \in \{-1, +1\}$,

$$\boldsymbol{q}^e \boldsymbol{x}(v) \quad = \quad [\boldsymbol{q}^e x_1(v), \ldots, \boldsymbol{q}^e x_m(v)]. \tag{3}$$

On the basis of these definitions, the mapping implemented by a Recursive Neural Network can be described by the following equations:

$$\begin{cases} \boldsymbol{x}(v) &= \tau(\boldsymbol{l}(v), \boldsymbol{q}^{-1}\boldsymbol{x}(v)) \\ \boldsymbol{y}(v) &= g(\boldsymbol{l}(v), \boldsymbol{x}(v)) \end{cases} \tag{4}$$

where $\boldsymbol{x}(v)$ and $\boldsymbol{y}(v)$ are respectively the network state and the network output associated to vertex v. This formulation is based on a structural version of the *causality* assumption, i.e., the output $\boldsymbol{y}(v)$ of the network at vertex v only depends on v and on its descendants. Specifically, RCC equations, where we disregard direct connections between hidden units, can be written, for $j = 1, \ldots, m$, as

$$x_j(v) = \tau_j(\boldsymbol{l}(v), \boldsymbol{q}^{-1}[x_1(v), \ldots, x_j(v)]), \tag{5}$$

where $x_i(v)$ is the output of the i-th hidden unit in the network when processing vertex v. Since RCC is a constructive algorithm, training of a new hidden unit is based on already frozen units. Thus, when training hidden unit k, the state variables $x_1, \ldots, x_{k-1}$ for *all* the vertexes of all the DPAGs in the training set are already available, and can be used in the definition of x_k. This observation is very important since it yields to the realization that *contextual* information is already available in RCC, but it is not exploited.

The *Contextual Recursive Cacscade Correlation* network [5] exploits this information. Specifically, eqs. (5) can be expanded in a contextual fashion by using, where possible, the shift operator $\boldsymbol{q}^{+1}$:

$$x_1(v) = \tau_1(\boldsymbol{l}(v), \boldsymbol{q}^{-1} x_1(v)), \tag{6}$$

$$\begin{aligned} x_j(v) = \tau_j(\boldsymbol{l}(v), & \boldsymbol{q}^{-1}[x_1(v), \ldots, x_j(v)], \\ & \boldsymbol{q}^{+1}[x_1(v), \ldots, x_{j-1}(v)]), \\ & j = 2, \ldots, m. \end{aligned} \tag{7}$$

4 Formal Determination of Context

In this section, we characterize the "shape" of the context exploited by a state variable in terms of the set of state variables which contribute directly or indirectly to its determination. Specifically, we propose a definition of "contextual window" and elucidate how the "shape" of the "contextual window" evolves with the addition of hidden units, both in the CRCC model and in the RCC model.

In order to formalize the above concepts, let us define the "descendants" operator $\downarrow$:

Definition 1 (Descendants operator) *The "descendants" operator* $\downarrow$ *applied to a subset of vertexes* $V \subseteq \mathrm{vert}(\mathcal{D})$ *returns the union of* V *with the set of all its descendants*

$$\downarrow V = V \cup \{u | v \in V \ \wedge \exists \ path(v, u)\},$$

or alternatively, using a recursive definition

$$\downarrow V = V \cup \downarrow \mathrm{out_set}(V),$$

where the descendants set of an empty set is still an empty set, i.e., $\downarrow \emptyset = \emptyset$.

Moreover, let the set functions $\mathrm{in_set}(\cdot)$ and $\mathrm{out_set}(\cdot)$ be defined also for sets of vertexes as argument (e.g., $\mathrm{in_set}(V) = \bigcup_{v \in V} \mathrm{in_set}(v)$) and let denote the repeated application with the powered function (e.g., $\mathrm{in_set}^p(V)$ corresponds to the repeated application of the function in_set for p times):

$$\mathrm{in_set}^p(V) \equiv \underbrace{\mathrm{in_set}(\cdots(\mathrm{in_set}(V))\cdots)}_{p \text{ times}}$$

Actually, we will use the repeated application of the in_set function composed with the "descendants" operator $\downarrow$, i.e,

$$(\downarrow \mathrm{in_set})^p(V) \equiv \downarrow \mathrm{in_set}(\cdots(\downarrow \mathrm{in_set}(V))\cdots)$$

For representational purposes we also introduce the following notation

Definition 2 (Dot operator) *Given a subset of vertexes* V*, with the notation* $x_j.V$ *we refer to the set of state variables* $\{x_j(v) | v \in V\}$*, where for an empty subset of vertexes we define* $x_j.\emptyset = \{x_0\}$.

We are interested in the associative properties of the above operators:

Lemma 1 (Associative rules) *For any couple of vertexes sets* $V, Z \subseteq \mathrm{vert}(\mathcal{D})$*, the following equations hold*

$$\downarrow V \cup \downarrow Z = \downarrow (V \cup Z)$$

and

$$x_j.V \cup x_j.Z = x_j.(V \cup Z)$$

Proof. It directly follows from the respective definitions of "descendants" (Definition 1) and "dot" (Definition 2) operators. ■

Finally, let formally define the concept of *context window*, i.e. the set of internal states which may contribute to the computation of a new internal state

Definition 3 *The context window of a state variable $x_k(v)$, denoted by $\mathcal{C}(x_k(v))$, is defined as the set of* all *state variables (directly or indirectly) contributing to its determination.*

In the following we will use the above definition to characterize the context window of the CRCC and RCC models.

First of all, let us state some basic results that will be used to derive the main theorems.

Lemma 2 $\mathcal{C}(x_0) = \emptyset$

Proof. Clearly, since the (constant value) state x_0 is not the result of a computation performed on other states, the set of states contributing to its determination is an empty set. ■

All the results reported in the following will concern the CRCC model.

Referring to eq. (6) it is now possible to derive the following

Lemma 3 *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$ the following equation holds*

$$\mathcal{C}(x_1(v)) = x_1.\downarrow \mathrm{out_set}(v).$$

Proof. It can be proved by induction on the partial order of vertexes.
By definition of context window (direct and indirect contribution), and by eq. (6) for any v in $\mathrm{vert}(\mathcal{D})$, the context window for the state variable $x_1(v)$ can be expressed as

$$\mathcal{C}(x_1(v)) = \underbrace{\bigcup_{i=1}^{out} q_i^{-1} x_1(v)}_{\text{direct}} \cup \underbrace{\bigcup_{i=1}^{out} \mathcal{C}(q_i^{-1} x_1(v))}_{\text{indirect}} \tag{8}$$

Base Case: v is a leaf of $\mathcal{D}$, i.e. a vertex without outgoing edges.

By definition of out_set, $\forall i \in [1, \ldots, \text{out}]$ $\text{out_set}_i(v) = \emptyset$, thus by definition of "descendants" operator $x_1.\downarrow\text{out_set}(v) = x_1.\emptyset = \{x_0\}$.
Moreover, by definition of extended shift operator, $\forall i \in [1, \ldots, out]$ $q_i^{-1}x_1(v) = x_0$, and by Lemma 2, $\mathcal{C}(q_i^{-1}x_1(v)) = \mathcal{C}(x_0) = \emptyset$. This proves that

$$\mathcal{C}(x_1(v)) = x_1.\downarrow\text{out_set}(v) = \{x_0\}$$

for any leaf v of the data structure $\mathcal{D}$.

Inductive Step: v is an internal vertex of $\mathcal{D}$. i.e. a vertex with at least one outgoing edge.

Assume that

$$\forall u \in \text{out_set}(v) \qquad \mathcal{C}(x_1(u)) = x_1.\downarrow\text{out_set}(u) \qquad \text{(Inductive Hypothesis)}$$

By definition of generalized shift operator,

$$\bigcup_{i=1}^{out} q_i^{-1}x_1(v) = \bigcup_{i=1}^{out} x_1(\text{out_set}_i(v)) = \bigcup_{u\in\text{out_set}(v)} x_1(u)$$

and

$$\bigcup_{i=1}^{out} \mathcal{C}(q_i^{-1}x_1(v)) = \bigcup_{u\in\text{out_set}(v)} \mathcal{C}(x_1(u))$$

thus, eq. (8) becomes

$$\mathcal{C}(x_1(v)) = \bigcup_{u\in\text{out_set}(v)} [x_1(u) \cup \mathcal{C}(x_1(u))].$$

By the induction hypothesis,

$$\mathcal{C}(x_1(v)) = \bigcup_{u\in\text{out_set}(v)} [x_1(u) \cup x_1.\downarrow\text{out_set}(u)].$$

Since $x_1(u) = x_1.\{u\}$, and by the associative rule for "dot" (Lemma 1), we have

$$\mathcal{C}(x_1(v)) = \bigcup_{u\in\text{out_set}(v)} x_1.[\{u\}\cup\downarrow\text{out_set}(u)],$$

and by the definition of "descendants" (Definition 1), the thesis follows:

$$\mathcal{C}(x_1(v)) = \bigcup_{u\in\text{out_set}(v)} x_1.\downarrow u = x_1.\downarrow\text{out_set}(v)$$

where in the last step we have again used the associative rule for "dot". ■

Corollary 1 *Given a DPAG $\mathcal{D}$, for any couple of vertexes $u, v \in \mathrm{vert}(\mathcal{D})$ connected by a path from v to u then*

$$\mathcal{C}(x_1(u)) \subseteq \mathcal{C}(x_1(v)).$$

Proof. It follows directly from Lemma 3, from the definitions of "descendants" operator and context window, and from the fact that the graph is acyclic. ■

Referring to eq. (7) it is now possible to derive the following lemmata, which are then used to prove Theorem 1

Lemma 4 *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$ the following equation holds*

$$\mathcal{C}(x_2(v)) = x_1.\downarrow \mathrm{in_set}(\downarrow v) \ \cup \ x_2.\downarrow \mathrm{out_set}(v). \tag{9}$$

Proof. It can be proved by induction on the partial order of vertexes.
By definition of context window, and by eq. (7) for any v in $\mathrm{vert}(\mathcal{D})$ the context window for the state variable $x_2(v)$ can be expressed by

$$\begin{aligned} \mathcal{C}(x_2(v)) \ = \ & \bigcup_{i=1}^{out} \left[q_i^{-1} x_2(v) \ \cup \ \mathcal{C}(q_i^{-1} x_2(v)) \right] \ \cup \\ & \bigcup_{i=1}^{out} \left[q_i^{-1} x_1(v) \ \cup \ \mathcal{C}(q_i^{-1} x_1(v)) \right] \ \cup \\ & \bigcup_{i=1}^{in} \left[q_i^{+1} x_1(v) \ \cup \ \mathcal{C}(q_i^{+1} x_1(v)) \right] \end{aligned} \tag{10}$$

Base Case: v is a leaf of $\mathcal{D}$, i.e. a vertex without outgoing edges.

By the definitions of out_set, in_set, and "descendants" operator it follows that $x_2.\downarrow \mathrm{out_set}(v) = x_2.\emptyset = \{x_0\}$, and $x_1.\downarrow \mathrm{in_set}(\downarrow v) = x_1.\downarrow \mathrm{in_set}(v)$. Thus eq. (9) becomes

$$\mathcal{C}(x_2(v)) = x_1.\downarrow \mathrm{in_set}(v) \cup \{x_0\}.$$

By definition of generalized shift operators, $q_i^{-1} x_2(v) = x_2.\mathrm{out_set}_i(v) = \{x_0\}$, and $q_i^{+1} x_1(v) = x_1.\mathrm{in_set}_i(v)$. Moreover, by Lemma 2, $\mathcal{C}(q_i^{-1} x_2(v)) = \mathcal{C}(x_2.\mathrm{out_set}_i(v)) = \mathcal{C}(\{x_0\}) = \emptyset$, and by Lemma 3, $\mathcal{C}(q_i^{+1} x_1(v)) = \mathcal{C}(x_1(\mathrm{in_set}_i(v))) = x_1.\downarrow \mathrm{out_set}(\mathrm{in_set}_i(v))$. Thus, by the associative rule on "dot" and by definition of "descendants" operator, eq. (10) becomes

$$\begin{aligned} \mathcal{C}(x_2(v)) \ &= \ \{x_0\} \cup \ x_1.\mathrm{in_set}(v) \cup \ x_1.\downarrow \mathrm{out_set}(\mathrm{in_set}(u)) \\ &= \ x_1.\downarrow \mathrm{in_set}(v) \end{aligned}$$

The thesis follows for all leaves of the structure.

Inductive Step: v is an internal vertex of $\mathcal{D}$. i.e. a vertex with at least one outgoing edge.

Assume that $\forall u \in \text{out_set}(v)$

$$\mathcal{C}(x_2(u)) = x_1.\downarrow\text{in_set}(\downarrow u) \cup\ x_2.\downarrow\text{out_set}(u) \qquad \text{(Inductive Hypothesis).}$$

By definition of generalized shift operator and by Lemma 3, eq. (10) becomes

$$\begin{aligned}\mathcal{C}(x_2(v)) \;=\; & \bigcup_{u\in\text{out_set}(v)} [x_2(u) \cup\ \mathcal{C}(x_2(u))] \cup \\ & \bigcup_{u\in\text{out_set}(v)} [x_1(u) \cup\ x_1.\downarrow\text{out_set}(u)] \cup \\ & \bigcup_{u\in\text{in_set}(v)} [x_1(u) \cup\ x_1.\downarrow\text{out_set}(u)]\end{aligned}$$

By the associative rule for "dot" and the definition of "descendants" operator

$$\begin{aligned}\mathcal{C}(x_2(v)) \;&=\; \bigcup_{u\in\text{out_set}(v)} [x_2(u) \cup\ \mathcal{C}(x_2(u))] \cup \bigcup_{u\in\text{out_set}(v)} x_1.\downarrow u \cup \bigcup_{u\in\text{in_set}(v)} x_1.\downarrow u \\ &=\; \bigcup_{u\in\text{out_set}(v)} [x_2(u) \cup\ \mathcal{C}(x_2(u))] \cup\ x_1.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(v)\end{aligned}$$

By the inductive hypothesis and by the definition of "descendants" operator

$$\begin{aligned}\mathcal{C}(x_2(v)) \;&=\; \bigcup_{u\in\text{out_set}(v)} [x_2(u) \cup\ x_2.\downarrow\text{out_set}(u) \cup\ x_1.\downarrow\text{in_set}(\downarrow u)] \cup \\ &\quad\; x_1.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(v) \\ &=\; \bigcup_{u\in\text{out_set}(v)} [x_2.\downarrow u \cup\ x_1.\downarrow\text{in_set}(\downarrow u)] \cup \\ &\quad\; x_1.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(v) \\ &=\; x_2.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(\downarrow\text{out_set}(v)) \cup \\ &\quad\; x_1.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(v)\end{aligned}$$

By the associative rules and by definition of in_set

$$\begin{aligned}\mathcal{C}(x_2(v)) \;&=\; x_2.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow[\text{in_set}(v) \cup\ \text{in_set}(\downarrow\text{out_set}(v))] \cup \\ &\quad\; x_1.\downarrow\text{out_set}(v) \\ &=\; x_2.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(v \cup\ \downarrow\text{out_set}(v)) \cup \\ &\quad\; x_1.\downarrow\text{out_set}(v) \\ &=\; x_2.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(\downarrow v) \cup\ x_1.\downarrow\text{out_set}(v).\end{aligned}$$

Since for any $v \in \text{vert}(\mathcal{D})$ it holds $\downarrow\text{out_set}(v) \subset \downarrow v \subseteq \downarrow\text{in_set}(\downarrow v)$ the thesis follows:

$$\mathcal{C}(x_2(v)) = x_2.\downarrow\text{out_set}(v) \cup\ x_1.\downarrow\text{in_set}(\downarrow v).$$

■

Lemma 5 *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$ and for any $k \geq 2$, it holds*

$$\mathcal{C}(x_k(v)) \supseteq \mathcal{C}(x_{k-1}(v)).$$

Proof. By definition of context window and by eq. (7), for any v in $\mathrm{vert}(\mathcal{D})$ the context window for the state variable $x_k(v)$ can be expressed by

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & \bigcup_{i=1}^{out} \left[q_i^{-1}x_1(v) \;\cup\; \mathcal{C}(q_i^{-1}x_1(v)) \cup \;\cdots\cup\; q_i^{-1}x_k(v) \;\cup\; \mathcal{C}(q_i^{-1}x_k(v))\right] \;\cup \\ & \bigcup_{i=1}^{in} \left[q_i^{+1}x_1(v) \;\cup\; \mathcal{C}(q_i^{+1}x_1(v)) \cup \;\cdots\cup\; q_i^{+1}x_{k-1}(v) \;\cup\; \mathcal{C}(q_i^{+1}x_{k-1}(v))\right]\end{aligned}$$

by the definition of generalized shift operator

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & \bigcup_{u\in \mathrm{out_set}(v)} [x_1(u) \;\cup\; \mathcal{C}(x_1(u)) \cup \;\cdots\cup\; x_k(u) \;\cup\; \mathcal{C}(x_k(u))] \;\cup \\ & \bigcup_{u\in \mathrm{in_set}(v)} [x_1(u) \;\cup\; \mathcal{C}(x_1(u)) \cup \;\cdots\cup\; x_{k-1}(u) \;\cup\; \mathcal{C}(x_{k-1}(u))]\end{aligned}$$

and reorganizing the unions of sets in a different way

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & \bigcup_{u\in \mathrm{out_set}(v)} [x_1(u) \;\cup\; \mathcal{C}(x_1(u)) \cup \;\cdots\cup\; x_{k-1}(u) \;\cup\; \mathcal{C}(x_{k-1}(u))] \;\cup \\ & \bigcup_{u\in \mathrm{in_set}(v)} [x_1(u) \;\cup\; \mathcal{C}(x_1(u)) \cup \;\cdots\cup\; x_{k-2}(u) \;\cup\; \mathcal{C}(x_{k-2}(u))] \;\cup \\ & \bigcup_{u\in \mathrm{out_set}(v)} [x_k(u) \;\cup\; \mathcal{C}(x_k(u))] \;\cup\; \bigcup_{u\in \mathrm{in_set}(v)} [x_{k-1}(u) \;\cup\; \mathcal{C}(x_{k-1}(u))]\end{aligned}$$

which can be rewritten as

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & \mathcal{C}(x_{k-1}(v)) \;\cup\; \mathcal{C}(x_k(\mathrm{out_set}(v))) \;\cup\; \mathcal{C}(x_{k-1}(\mathrm{in_set}(v))) \;\cup \\ & x_k(\mathrm{out_set}(v)) \;\cup\; x_{k-1}(\mathrm{in_set}(v))\end{aligned}$$

hence

$$\mathcal{C}(x_k(v)) \supseteq \mathcal{C}(x_{k-1}(v)).$$

■

We are now ready to state the main theorem, i.e., the expression which explicitly defines the CRCC context for any state variable:

Theorem 1 *Given a DPAG $\mathcal{D}$, for any vertex $v \in \mathrm{vert}(\mathcal{D})$, and for any index $k \geq 2$, the following equation holds*

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i.(\downarrow \mathrm{in_set})^{k-i}(\downarrow v) \ \cup \ x_k.\downarrow \mathrm{out_set}(v). \tag{11}$$

Proof. The theorem can be proved by induction on the partial order of vertexes, and on the order of indexes of variables.
By definition of context window, by eq. (7), and by definition of generalized shift operator, for any v in $\mathrm{vert}(\mathcal{D})$ the context window for the state variable $x_k(v)$ can be expressed by

$$\begin{aligned}\mathcal{C}(x_k(v)) \ = \ & \bigcup_{u \in \mathrm{out_set}(v)} [x_1(u) \ \cup \ \mathcal{C}(x_1(u)) \cup \ \cdots \cup \ x_k(u) \ \cup \ \mathcal{C}(x_k(u))] \ \cup \\ & \bigcup_{u \in \mathrm{in_set}(v)} [x_1(u) \ \cup \ \mathcal{C}(x_1(u)) \cup \ \cdots \cup \ x_{k-1}(u) \ \cup \ \mathcal{C}(x_{k-1}(u))]\end{aligned}$$

Applying lemma 5 (i.e., $\forall i < k$, $\mathcal{C}(x_k(u)) \supseteq \mathcal{C}(x_i(u))$)

$$\begin{aligned}\mathcal{C}(x_k(v)) \ = \ & \bigcup_{u \in \mathrm{out_set}(v)} [x_1(u) \ \cup \ \cdots \ \cup \ x_k(u)] \ \cup \ \bigcup_{u \in \mathrm{out_set}(v)} \mathcal{C}(x_k(u)) \cup \\ & \bigcup_{u \in \mathrm{in_set}(v)} [x_1(u) \ \cup \ \cdots \ \cup \ x_{k-1}(u)] \ \cup \ \bigcup_{u \in \mathrm{in_set}(v)} \mathcal{C}(x_{k-1}(u)) \\ = \ & \bigcup_{i=1}^{k} x_i.\mathrm{out_set}(v) \ \cup \ \mathcal{C}(x_k.\mathrm{out_set}(v)) \ \cup \\ & \bigcup_{i=1}^{k-1} x_i.\mathrm{in_set}(v) \ \cup \ \mathcal{C}(x_{k-1}.\mathrm{in_set}(v))\end{aligned} \tag{12}$$

Base Case: v is a leaf of $\mathcal{D}$, i.e. a vertex without outgoing edges.

For $k = 2$, eq. (11) reduces to eq. (9) (the base case for which the theorem has been proved). Now, let assume that $\forall j, \ 2 \leq j \leq k-1$

$$\mathcal{C}(x_j(v)) = \bigcup_{i=1}^{j-1} x_i.(\downarrow \mathrm{in_set})^{j-i}(\downarrow v) \ \cup \ x_j.\downarrow \mathrm{out_set}(v) \qquad \text{(Inductive Hypothesis)} \tag{13}$$

Since the vertex v is a leaf, eq. (12) can be reduced to

$$\mathcal{C}(x_k(v)) \ = \ \{x_0\} \ \cup \ \emptyset \ \cup \ \bigcup_{i=1}^{k-1} x_i.\mathrm{in_set}(v) \ \cup \ \mathcal{C}(x_{k-1}.\mathrm{in_set}(v))$$

and, applying the inductive hypothesis 13, it becomes

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & \bigcup_{i=1}^{k-1} x_i.\mathrm{in_set}(v) \;\cup \\ & \bigcup_{i=1}^{k-2} x_i.(\downarrow\mathrm{in_set})^{k-1-i}(\downarrow\mathrm{in_set}(v)) \;\cup\; x_{k-1}.\downarrow\mathrm{out_set}(\mathrm{in_set}(v))\end{aligned}$$

which can be easily rewritten as

$$\begin{aligned}\mathcal{C}(x_k(v)) \;=\; & x_{k-1}.\mathrm{in_set}(v) \;\cup\; x_{k-1}.\downarrow\mathrm{out_set}(\mathrm{in_set}(v)) \;\cup \\ & \bigcup_{i=1}^{k-2}[x_i.\mathrm{in_set}(v) \;\cup\; x_i.(\downarrow\mathrm{in_set})^{k-i}(v)]\end{aligned}$$

By definition of "descendants" operator, and because of the fact that for any $v \in \mathrm{vert}(\mathcal{D})$ and for any $j \geq 1$ it trivially holds $\mathrm{in_set}(v) \subseteq\downarrow \mathrm{in_set}(v) \subseteq (\downarrow \mathrm{in_set})^j(v)$, it follows that

$$\begin{aligned}\mathcal{C}(x_k(v)) \;&=\; x_{k-1}.\downarrow\mathrm{in_set}(v) \;\cup\; \bigcup_{i=1}^{k-2} x_i.(\downarrow\mathrm{in_set})^{k-i}(v) \\ &=\; \bigcup_{i=1}^{k-1} x_i.(\downarrow\mathrm{in_set})^{k-i}(v)\end{aligned}$$

On the other hand, since v is a leaf, notice that $v =\downarrow v$ and $x_k.\downarrow\mathrm{out_set}(v)$ can be reduced to $\{x_0\}$, thus eq. (11) can be rewritten as

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i.(\downarrow\mathrm{in_set})^{k-i}(v)$$

Which proves the correctness of the assertion for all leaves of $\mathcal{D}$ and for any $k \geq 2$

Inductive Step: v is an internal vertex of $\mathcal{D}$. i.e. a vertex with at least one outgoing edge.

Assume that

$$\mathcal{C}(x_j(u)) = \bigcup_{i=1}^{j-1} x_i.(\downarrow\mathrm{in_set})^{j-i}(\downarrow u) \cup x_j.\downarrow\mathrm{out_set}(u) \qquad \text{(Inductive Hypothesis)} \quad (14)$$

holds $\forall u \in \mathrm{vert}(\mathcal{D})$ if $2 \leq j \leq k-1$, and $\forall u \in \mathrm{out_set}(v)$ if $j = k$, then, eq. (12) becomes

$$\mathcal{C}(x_k(v)) \;=\; \bigcup_{i=1}^{k} x_i.\mathrm{out_set}(v) \;\cup\; \mathcal{C}(x_k.\mathrm{out_set}(v)) \;\cup\; \bigcup_{i=1}^{k-1} x_i.\mathrm{in_set}(v) \;\cup$$

$$\left[\bigcup_{i=1}^{k-2} x_i.(\downarrow \text{in_set})^{k-1-i}(\downarrow \text{in_set}(v)) \ \cup \ x_{k-1}.\downarrow \text{out_set}(\text{in_set}(v))\right]$$

$$= \bigcup_{i=1}^{k} x_i.\text{out_set}(v) \ \cup \ \mathcal{C}(x_k.\text{out_set}(v)) \ \cup \ \bigcup_{i=1}^{k-1} x_i.\text{in_set}(v) \ \cup$$

$$\left[\bigcup_{i=1}^{k-2} x_i.(\downarrow \text{in_set})^{k-i}(v) \ \cup \ x_{k-1}.\downarrow \text{out_set}(\text{in_set}(v))\right]$$

By the associative rule of "dot" and "descendants" operators

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out_set}(v) \ \cup \ \mathcal{C}(x_k.\text{out_set}(v)) \ \cup \ \bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup \ x_{k-1}.\text{in_set}(v) \ \cup$$

$$\bigcup_{i=1}^{k-2} x_i.(\downarrow \text{in_set})^{k-i}(v) \ \cup \ x_{k-1}.\downarrow \text{out_set}(\text{in_set}(v))$$

$$= \bigcup_{i=1}^{k} x_i.\text{out_set}(v) \ \cup \ \mathcal{C}(x_k.\text{out_set}(v)) \ \cup \ \bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup$$

$$\bigcup_{i=1}^{k-2} x_i.(\downarrow \text{in_set})^{k-i}(v) \ \cup \ x_{k-1}.\downarrow \text{in_set}(v)$$

$$= \bigcup_{i=1}^{k} x_i.\text{out_set}(v) \ \cup \ \mathcal{C}(x_k.\text{out_set}(v)) \ \cup \ \bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup$$

$$\bigcup_{i=1}^{k-1} x_i.(\downarrow \text{in_set})^{k-i}(v)$$

Applying again the induction hypotesis 14 on $\mathcal{C}(x_k.\text{out_set}(v))$

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\text{out_set}(v) \ \cup$$

$$\bigcup_{i=1}^{k-1} x_i.(\downarrow \text{in_set})^{k-i}(\downarrow \text{out_set}(v)) \ \cup \ x_k.\downarrow \text{out_set}(\text{out_set}(v)) \ \cup$$

$$\bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup \ \bigcup_{i=1}^{k-1} x_i.(\downarrow \text{in_set})^{k-i}(v)$$

By the associative properties of "dot" and "descendants" operators applied to $x_i.(\downarrow \text{in_set})^{k-i}$ and $x_k.\text{out_set}$

$$\mathcal{C}(x_k(v)) = \bigcup_{i=1}^{k-1} x_i.\text{out_set}(v) \ \cup \ \bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup$$

$$\bigcup_{i=1}^{k-1} x_i.(\downarrow \text{in_set})^{k-i}[\downarrow \text{out_set}(v) \ \cup \ v] \ \cup$$

$$
\begin{aligned}
& x_k.\downarrow\text{out_set}(\text{out_set}(v)) \ \cup \ x_k.\text{out_set}(v) \\
= & \bigcup_{i=1}^{k-1} x_i.\text{out_set}(v) \ \cup \ \bigcup_{i=1}^{k-2} x_i.\text{in_set}(v) \ \cup \\
& \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in_set})^{k-i}(\downarrow v) \ \cup \ x_k.\downarrow\text{out_set}(v)
\end{aligned}
$$

Since for any $v \in \text{vert}(\mathcal{D})$ and for any $j \geq 1$ it trivially holds that

$$\text{out_set}(v) \subseteq \downarrow\text{in_set}(v) \subseteq (\downarrow\text{in_set})^j(v) \subseteq (\downarrow\text{in_set})^j(\downarrow v),$$

and similarly

$$\text{in_set}(v) \subseteq \downarrow\text{in_set}(v) \subseteq (\downarrow\text{in_set})^j(v) \subseteq (\downarrow\text{in_set})^j(\downarrow v),$$

the thesis follows:

$$\mathcal{C}(x_k(v)) \ = \ \bigcup_{i=1}^{k-1} x_i.(\downarrow\text{in_set})^{k-i}(\downarrow v) \ \cup \ x_k.\downarrow\text{out_set}(v)$$

■

For the sake of comparison, let us now consider the "context window" for the RCC model (eq. 5). The following theorem holds:

Theorem 2 *Given a DPAG $\mathcal{D}$, for any vertex $v \in \text{vert}(\mathcal{D})$, and for any index $k \geq 1$, the following equation holds*

$$\mathcal{C}_{RCC}(x_k(v)) = \bigcup_{i=1}^{k} x_i.\downarrow\text{out_set}(v). \tag{15}$$

Proof. For $k = 1$ the proof is given by Lemma 3, since eq. 5 is equivalent to eq. 6. For $k > 1$ the proof is similar to the proof given for Theorem 1 where, this time,

$$\mathcal{C}_{RCC}(x_k(v)) \ = \ \bigcup_{u \in \text{out_set}(v)} [x_1(u) \ \cup \ \mathcal{C}_{RCC}(x_1(u)) \cup \ \cdots \cup \ x_k(u) \ \cup \ \mathcal{C}_{RCC}(x_k(u))]$$

■

Given a state variable, this theorem allows us to directly compare the context of the CRCC model with the corresponding context in RCC. From this comparison it is very easy to see that any RCC context is strictly contained in the corresponding CRCC context. This is consistent with the fact that there are functions, i.e., contextual functions, which cannot be implemented by RCC, while CRCC is able to implement them. Additional computational results exploiting these theorems are described in [6].

5 Conclusion

In this paper we have given a formal characterization of the functional dependencies between state variables in recursive models based on Cascade Correlation for the processing of structured data. Specifically, we have defined the "context window" for a state variable as the set of state variables that contribute, directly or indirectly, to the determination of the considered state variable. Then, on the basis of this definition, we have been able to derive a compact expression describing the "context window" as a function of the state variables associated to specific sets of ancestors and descendants belonging to the considered vertex in the current input structure. This expression has been devised both for the *Recursive Cascade Correlation* model and the *Contextual Recursive Cascade Correlation* model.

The relevance of the results obtained in this paper is due to the possibility to use them in order to characterize the computational power of the above models (see [6]). Moreover, the approach used in this paper, which is independent from the specific neural implementation, can easily be applied to other recurrent and recursive models in order to compare them from a computational point of view.

Acknowledgment

This work has been partially supported by MIUR grants 2002093941_004 and 2001034475_006.

References

[1] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15(11):937–946, 1999.

[2] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786, 1998.

[3] C. Goller and A. Küchler. Learning task-dependent distributed structure-representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, pages 347–352, 1996.

[4] A. Micheli, D. Sona, and A. Sperduti. Bi-causal recurrent cascade correlation. In *Proc. of the Int. Joint Conf. on Neural Networks - IJCNN'2000*, volume 3, pages 3–8, 2000.

[5] A. Micheli, D. Sona, and A. Sperduti. Recursive cascade correlation for contextual processing of structured data. In *Proc. of the Int. Joint Conf. on Neural Networks - WCCI-IJCNN'2002*, volume 1, pages 268–273, 2002.

[6] A. Micheli, D. Sona, and A. Sperduti. Contextual processing of structured data by recursive cascade correlation. Submitted to IEEE Trans. on Neural Networks, 2003.

[7] A. Sperduti, D. Majidi, and A. Starita. Extended cascade-correlation for syntactic and structural pattern recognition. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture notes in Computer Science*, pages 90–99. Springer-Verlag, 1996.

[8] A. Sperduti and A. Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[9] H. Wakuya and J. Zurada. Bi-directional computing architectures for time series prediction. *Neural Network*, 14:1307–1321, 2001.