RECOGNIZING MATCHING AMONG TERMS: A NEURAL NETWORK APPROACH

PART A: BUILDING THE NETWORKS

Carlos Mareco *, Alberto Paccanaro * ‡

* Laboratorio de Electrónica Digital
Universidad Católica "Nuestra Señora de la Asunción"
Asunción, Paraguay
Tel: +595-21-334650
Fax: +595-21-310587
E-mail: *cmareco@ledip.py*

‡ Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Milano, Italy

ABSTRACT

This project aims at building Neural Networks that would receive as inputs (n+1) terms $\{T_1,..T_n, T\}$ and would suggest which T_is are matched by a subterm of T, with varying n. This first part of the paper, presents a particular net that can solve the problem for n=1, with a good degree of accuracy. This net was then used as a building block for larger nets that were applied to solve the problem for higher values of n.

1. INTRODUCTION TO THE PROBLEM

We assume the reader to be familiar with the basic concepts of First Order Logic (FOL) and Term Rewriting Systems (TRS). For a full coverage of the subject, see (Knuth & Bendix, 1970; Hullot, 1980; Hsiang & Rusinovitch, 1987; Dershowitz, 1989; Chang & Lee 1976).
In previous papers (Paccanaro & Willigs, 1994; Paccanaro, 1995) a particular Neural Network and a method for codifying terms and rules belonging to the complete TRS for the Group Theory were

presented; the net, given a term belonging to such theory, could suggest with a good degree of accuracy a rewrite rule belonging to the complete TRS that could reduce the term.
Notice that such net is therefore suggesting a rule of the TRS whose left hand side (lhs) is *matched* by a subterm of the term: it is actually identifying matching pairs.
The results obtained encouraged us to further investigate this area, and we focused on the Matching relation among FOL terms using Neural Networks.
Let us give a formal definition of Matching:

Matching: a term **t** is said to match a term **l** if there exists a substitution σ such that $\mathbf{l}\sigma = \mathbf{t}$.

The problem (1) we decided to investigate, can be stated as follows:

Let T(X,F) be the set of terms which can be built from a (finite) set of function symbols F (we assume constants to be 0-ary function symbols), and a (countable) set of variables X. Let $\{T_1, T_2, T_3, \ldots, T_n, T\}$ *where* T_i, $T \in T(X,F)$ *and n=1, 2, ..., be a set of terms such that there is only a term* T_k *that is matched by a subterm of T. The problem is to determine such* T_k.

We tried to find a class of neural networks that would receive as inputs (n+1) terms $\{T_1,\ldots,T_n,T\}$ and would suggest which T_k is matched by a subterm of T, with varying n.
It is worth to point out how the problem previously investigated in (Paccanaro & Willigs, 1994) turns out to be a particular case of this more general one: if we let the T_i be terms constituting the LHS of the rules of a completed TRS, and T the term to be rewritten with respect to the TRS, we get the previous problem back. Solving this more general problem would allow us to guide term rewriting with respect to any TRS.

2. EXPERIMENTAL SETUP

2.1 Databases of terms.

We decided to limit the problem to terms of length 7 or less, generated from a set of 3 function symbols: f, i and e with arity 2, 1 and 0 respectively.
Seven different databases of terms were built to be used in the experiments. Four of them were generated using a number of distinct variables varying between 1 and 4 (using the functions symbols described above and limiting the length of the terms to 7, a term may have at the most 4

distinct variables). In the following these databases will be indicated as DBi, where i = 0, ..., 4 stands for the number of distinct variables which were used for building the terms.

Then 3 more databases were generated from the ones containing 2, 3, and 4 variables by eliminating terms that could be made equal through variable renaming. This allowed to generate databases of terms which were smaller than the original ones leaving unchanged the total number of possible different variables present in each term.

In the following these databases will be indicated as DBRi, where i = 2, 3, 4 stands for the number of distinct variables which were used for building the terms before renaming.

The following table presents some relevant features of the composition of the databases which will be used in the subsequent discussions.

Database	# of terms	avg. # of var for term	avg. length of term
DB1	570	1.4228	6.4842
DB2	1875	2.0171	6.5968
DBR2	982	1.9257	6.5794
DB3	4556	2.3593	6.6620
DBR3	1074	2.0456	6.6024
DB4	9285	2.5878	6.7060
DBR4	1079	2.0547	6.6043

Table 1

It is worth to notice that DB4 can be considered as the "most general" database, being the largest and the one containing all the others.

2.2 Term Codification

Terms were codified as vectors of integer numbers of length 7 using the method that was found to yield best results in (Paccanaro & Willigs, 1994).

A positive integer number was assigned to each symbol and each element of the codified vector contained the number associated to the symbol in that position of the term. The vector was filled up starting from its last position, while a value of 0 was assigned to the elements not used for codifying the term.

In the following we shall present results of experiments which make use of a codification for terms which assigns the values of 500, 700 and 900 to the symbols "f", "i", and "e" respectively, while all the variables are assigned to 0. According to this codification, the vector corresponding to the term "f(i(x), f(e))" would turn out to be:

[0, 0, 500, 700, 0, 500, 900].

2.3 Networks Parameters

The learning algorithm used was the Normalized Cumulative Delta Rule. The processing elements used the hyperbolic tangent as transfer function. No noise was added to the inputs.

All nets were trained with a number of iterations between 200,000 and 300,000. The initial values used for the learning coefficient and momentum were 0.7 and 0.6 respectively. These values were gradually reduced as the learning process progressed.

Also, it is important to notice that the processing elements of the networks did not actually receive as inputs numbers such as 500 or 700, as they would soon saturate the neurons. To prevent this, the values used in the codification were scaled between -1 and 1, using linear scaling.

3. RECOGNIZING MATCHING AMONG PAIRS OF TERMS

We began trying to build a net, such that it could solve the problem for n=1. The net would receive as input 2 terms, T_1 and T, and should decide whether T_1 is matched by a subterm of T.

The net will have therefore 14 input elements (2 terms of length 7) and 1 output element.

For each database of terms a set of pairs of terms was built, according to the following procedure: for each term T in the database, two terms T_A, T_B were randomly chosen (from the same database), such that T_A was matched by a subterm of T while T_B was not; these terms were used to build 2 pairs [(T_A; T), 1], and [(T_B; T), 0]. From each set thus obtained, 75% of the pairs were randomly chosen to be used for training, while the remaining 25% were kept for testing purposes.

Various net architectures were tried. During testing the output of the net was considered as 1 when above 0.5, and 0 otherwise. The network architecture that gave best results is shown in Fig 1.

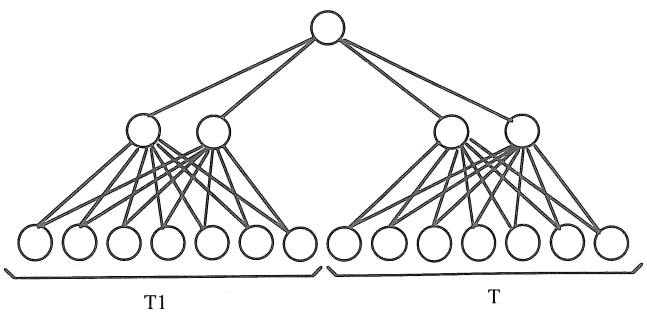

Figure 1

Table 2 presents the percentage of accuracy obtained testing the net with all the testing sets after being trained with all the training sets.

The notation N_DBi and N_DBRi identifies the net trained with the training sets from databases DBi and DBRi respectively. The notation T_DBi and T_DBRi denotes the test sets from databases DBi and DBRi respectively.

	T_DB1	T_DB2	T_DBR2	T_DB3	T_DBR3	T_DB4	T_DBR4	average
N_DB1	89.00	86.30	87.40	80.00	82.50	75.31	82.59	85.39
N_DB2	89.38	86.32	87.84	82.10	83.45	78.65	83.65	86.42
N_DBR2	88.65	85.35	87.84	80.97	83.09	76.91	82.41	85.20
N_DB3	89.39	85.07	86.86	81.84	83.98	78.75	82.41	84.22
N_DBR3	88.01	83.71	85.88	79.80	82.38	76.87	81.52	84.32
N_DB4	87.60	83.90	84.90	81.02	82.74	78.00	81.70	83.16
N_DBR4	88.35	84.75	86.47	79.63	81.49	76.61	82.59	84.53

Table 2

It is interesting to notice how for the different tests the net gave results which were relatively independent of the set it had been trained with.

This supports the belief that the net is learning to recognize the matching relation among terms: for example N_DB2 performs almost equally well when tested with T_DB1 or T_DBR4 though the tests come from databases with different statistical features.

We decided to analyze how errors were related to the number of variables and to the length of the terms involved.

The distribution of the error with respect to the number of variables and the length of the terms T_1 and T in the testing pairs was studied for every N_DBi, N_DBRi, and T_DBi, T_DBRi.

Fig. 2 shows how the error is distributed with respect to the number of variables (a) and the length (b) of the terms T_1 and T in the testing pairs separately, for N_DB2 and all the test sets.

The results of the net for the different tests were very similar. From this we can see that the degree of difficulty of the problem did not seem to be strongly related to the number of variables involved.

On the other hand the net did seem to encounter serious difficulties when the term T became large, nearly all the errors being found when such term has a length above 5.

Also an increasing degree of difficulty is related to longer T_1 terms, but not as strongly as to the length of T terms.

The others N_DBi, N_DBRi gave very similar results to the one shown for the N_DB2 case.

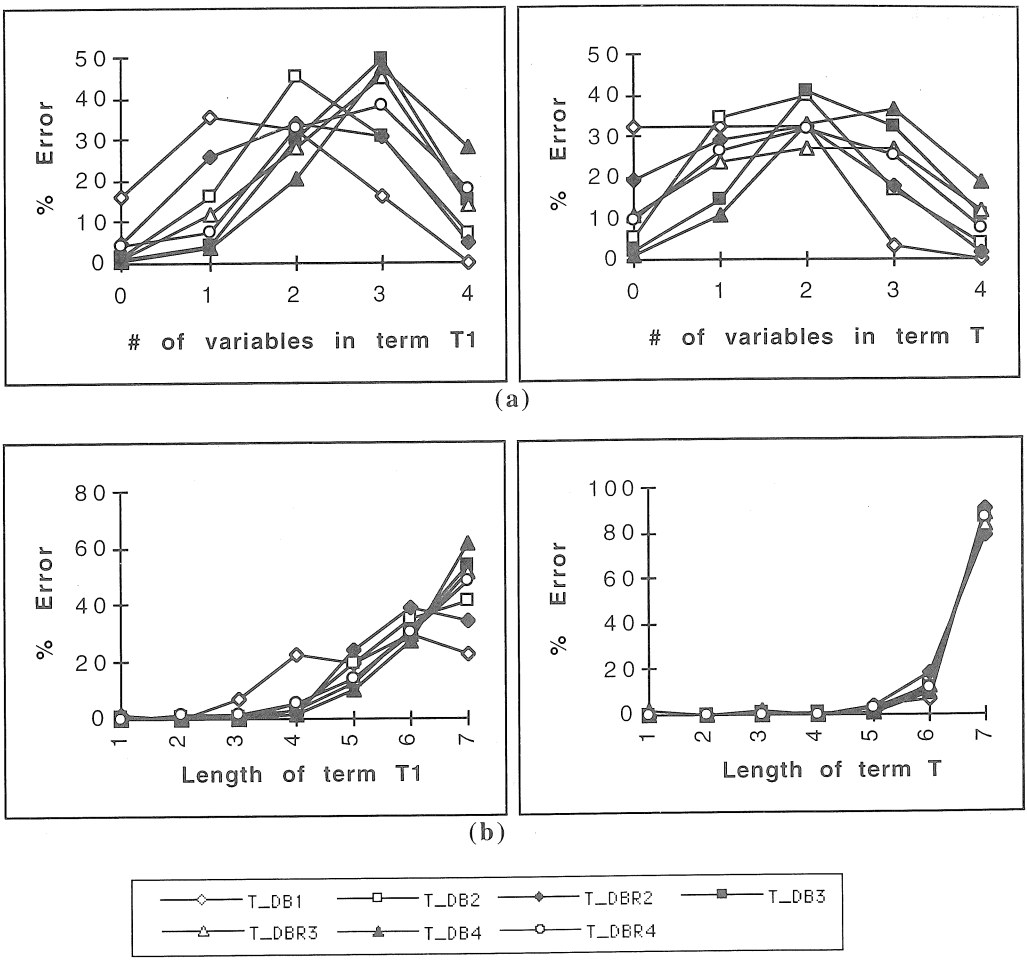

Figure 2

4. RECOGNIZING MATCHING IN THE GENERAL CASE

We then moved to the final goal stated at the beginning of this paper: to identify a class of nets that would work independently with respect to the number n of terms to be verified for matching against the term T.

Notice that increasing the number of input terms would lead to nets of increasing complexity (in terms of the number of connections); this would soon become impractical for training on the small machines (PC 486) we have available.

In order to solve the problem for higher values of n, larger nets were built utilizing the net which was used for solving the problem for n=1 as a building block.

The resulting net architecture for any value of n is shown in Fig. 3

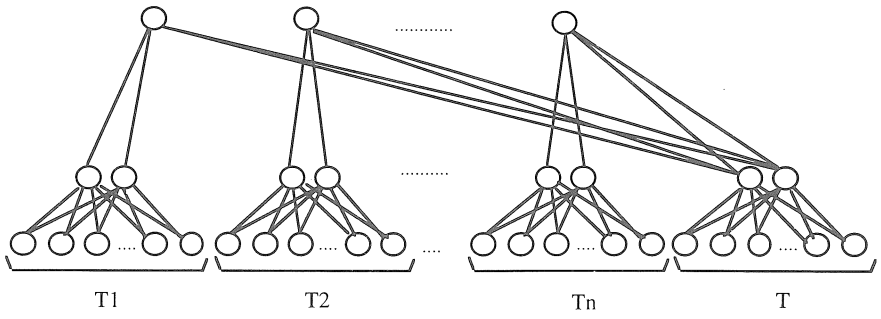

Figure 3

The connections from the T_is to the hidden layer and then to the output layer were assigned the same value as the respective connections from T_1 in the previous net after training. In the same way, connections from T to the hidden layer and then to the output layer were assigned the same value they had in the respective connection of the previous net after training.

It is important to notice how such bigger nets would not therefore have to be trained, thus avoiding the problem mentioned in the beginning of this section.

In the experiments presented in the following the values of the connections of N_DB2 were assigned to the connections of the new nets.

Such nets were then tested for problem(1) with varying n, using (n+1)-tuples of terms randomly built using terms found in DB4. The percentage of accuracy obtained for n=4, 8, and for a maximum length of the term T varying from 4 to 7 are shown in table 3.

	n = 4	n = 8
max. length 7	72.66	60.9375
max. length 6	90.716	81.1358
max. length 5	96.15	92.1154

Table 3

As expected, the percentage of accuracy decreases as n increases, but still the nets are quite reliable as long as the term T has a maximum length of 5.
It is worth to point out how a sequential solution of the problem would require an increasing amount of time as the number of terms to be verified for matching against a subterm of term T increases. On the other hand the amount of time using the approach proposed in this paper remains constant.

5. CONCLUSIONS

The purpose of this project was to build a Neural Network that would receive as inputs (n+1) terms $\{T_1,\ldots,T_n,T\}$ and would suggest which T_is are matched by a subterm of T, with varying n.
The nets presented in this paper can solve the problem for n=1 with a good degree of accuracy. These nets were then used as a building block for larger nets that were applied to solve the problem for higher values of n.
The greatest shortcoming of the method is constituted by the fact that the performance of the nets decrease for increasing size of n. The second part of this paper presents a way to improve these results, by specializing different nets to handle those cases which were found to be particularly difficult for the nets introduced here.

REFERENCES

C.L.Chang, R.C.T.Lee [1973], "Symbolic Logic and Mechanical Theorem Proving", Academic Press, New York

N.Dershowitz [1989], "Completion and its applications", in Resolution of equations in algebraic structures, pp. 31-48.

J.Hsiang, M.Rusinovitch [1987], "On Word Problems in Equational Theories", 14th Int. Conf. on Automata Languages and Programming, Karlsrhue.

J.M.Hullot [1980], "A catalogue of canonical term rewriting systems", Rept. CSL-113, SRI International, Menlo Park, CA.

D.E.Knuth, P.Bendix [1970], "Simple Word Problems in Universal Algebras", Proceedings of the Conf. on Computational Problems in Abstract Algebras, Pergamon Press, pp. 263-298.

A.Paccanaro , C.Willigs[1994], "Guiding Term Simplification in a Rewriting Based Automated Theorem Prover through a Neural Network", Proceedings of the XIV Int. Conf. of the Chilean Computer Science Society, pp. 315 - 322

A.Paccanaro [1995], "Guiding Term Simplification using Neural Networks: some results for the Group Theory", Proceedings of Neural, Parallel and Scientific Computations, v. 1, pp. 377 - 382