EXTENDED ABSTRACT

REACHABILITY MATRICES ASSIST IN DEFINING NEURAL NETWORK EXPERIMENTS

George G. Lendaris and David N. Todd
Systems Science Ph.D. Program
Portland State University
Portland, OR 97207-0751

BACKGROUND

The context for the present inquiry is a knowledge system with a large store of knowledge facts, where it is desired to implement a high-speed fact retrieval process. It is assumed that the 'conceptual graph' formalism [Sowa, 1985] is used for storing facts in the knowledge system (facts so represented are here called FactCGs [Fact Conceptual Graphs]). It is supposed that the query process will involve creating a QueryCG to represent the question, this will be presented to the system, and FactCGs from the store that bear appropriate relationships to the QueryCG will be fetched and presented to the user.

Prior to creating a knowledge system, a set of concept types and a set of relation types must be developed for encoding the knowledge. Of importance here is the fact that each concept type is related as a sub- or super-type of some other concept. For ease of visualization, these type relationships are often shown as a lattice structure (called a concept-type lattice).

A conceptual graph (CG) consists of what are called concept nodes, relation nodes, and connections between selected concept and relation nodes. Each FactCG and QueryCG will contain those relation types and concept types required to represent the given piece of knowledge. For a FactCG to be a candidate answer to a QueryCG, every relation node type in the QueryCG must be contained in the FactCG. In addition, to each of the concept nodes in the QueryCG, there must exist an appropriately related concept node in the FactCG...specifically (using the concept-type lattice as reference), the type of each concept node in the QueryCG must be above and connected to (or the same as) the type of at least one of the concept nodes in the FactCG.

In knowledge fact retrieval processes, the concept-type check is performed a large number of times. Since the above-and-connected-to part of the check is in effect a tree searching operation (and as is well known, tree searches are very expensive in terms of computation cycles) there is great incentive to make this operation as fast as possible. One way to speed up the operation is to develop a parallel mechanism for carrying it out.

NEURAL NETWORKS AND THE REACHABILITY MATRIX

Two formulations of the concept-type checking operation have been developed for the present investigation, and experiments carried out using neural networks (NNs) as the parallel implementations. For both formulations, a hypothetical concept-type lattice is created, and a corresponding adjacency (or, connection) matrix representation for the lattice is also created [Roberts, 1976]. Standard methods [Warfield, 1976] are used to generate a reachability matrix from the adjacency matrix. The reachability matrix is used in a fundamental way in both of the formulations.
Formulation 1

Given two concept types Ci & Cj, we ask the question "What is the relation of Ci to Cj in the concept-type lattice [shorthand: Ci \( \Rightarrow \) Cj]?" There are four possible answers: Ci is a super-type of Cj, a sub-type of Cj, the same type as Cj, and, is not comparable to Cj. These correspond to the following statements relative to the concept-type lattice: Ci is above Cj with a path connecting them; Ci is below Cj with a path connecting them; Ci and Cj are the same location in the lattice; and, there is no path that connects Ci and Cj in the (directed) lattice. The research approach is to train a neural network to answer the [Ci \( \Rightarrow \) Cj] question, where the NN has 4 output wires, one for each of the possible answers [Lendaris, 1989].

The "training" of neural networks proceeds along one of two major avenues, called respectively, supervised learning (the one used in the present investigations) and unsupervised learning. Supervised learning is a paradigm that incorporates a role called "teacher," where the teacher goes through the following sequence of steps: presents an input to the NN, lets the NN calculate the output appropriate to the current settings of its learning parameters (weights), compares the NN's output with what the teacher knows is the correct output [in our case, the latter information will be derived from the known hierarchy lattice], and finally, according to the results of the comparison, the teacher takes actions designed to have the NN converge to an input/output function that correctly maps each of the input patterns of interest [in our case, all possible pairs (Ci, Cj)] to the corresponding desired output.

A key component of the training process is the data set upon which the NN is trained. The training data set contains patterns to be used as inputs during training, and corresponding to each of these, the "correct" output. Depending on the application, development of such training data sets ranges from being easy to rather difficult. For the present application, this task leans toward the difficult. The difficulty is contributed to by the desire to investigate different strategies of presenting data to the NN...for the purpose of determining if it is possible/practical to take advantage of the a priori knowledge that there exists an underlying lattice structure for the input data set (such a priori knowledge of the input data is generally not available for NN applications.)

The reachability matrix is used as the vehicle for developing, in an organized way, the training data sets for the various experiments undertaken. This will be described in the presentation.

Formulation 2.

We begin here with a definition of upward closure** of a concept type Ci. This is the set of concept types contained in the path from Ci up to the top of the lattice [alternatively, the smallest set containing Ci and all of its supertypes].

The second formulation then proceeds as follows: Given two concept types Ci & Cj, we ask the question "Is Ci in the Upward Closure of Cj [shorthand: Ci \( \uparrow \) U Cj]?" Assume the lattice is constructed with the lines directed from higher nodes to lower nodes. Further, let the direction in the adjacency and reachability matrixes be from row to column [see Figure 1a & 1b]. With this representation, the upward closure of Cj is simply the collection of the row Ci's indicated by the presence of 1's in the Cj column
Reachability Matrices / Neural Networks

of the reachability matrix. If there are \( n \) concept types in the lattice, there are \( n \) possible [Ci ?U Cj] questions, each one answered via one of the columns in the reachability matrix (see Figure 1c). As this methodology is developed, the objective will be to allow compound questions involving multiple Cj's and/or Ci's. This application of the reachability matrix will also be demonstrated in the presentation.

![Connection Matrix](attachment://connection_matrix.png)

![Reachability Matrix](attachment://reachability_matrix.png)

![Upward Closures](attachment://upward_closures.png)

**Figure 1.** Coordinated examples of Connection (Adjacency) and Reachability Matrices, and use of Reachability Matrix for Upward Closure determination.

REFERENCES


** Use of this notion in the present context was suggested by a former student, James Marsh.