CHAPTER 3

ARTIFICIAL NEURAL NETWORKS

3.1 Introduction

An artificial neural networks (ANNs) model is a functional abstraction of the biological neural structures of the central nervous system. They are composed of many simple and highly interconnected computational elements, also called neurons, that operate in parallel and are arranged in patterns similar to biological neural nets. Figure 3.1 shows the typical diagram of a neuron. The neurons are connected by weighted links passing signals from one neuron to another. Each neuron receives a number of input signals through its connections. The output signal is transmitted through the neuron's outgoing connection. The outgoing connection, in turn, splits into a number of branches that transmit the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network.

It is generally thought that a neural network is highly sophisticated nonlinear dynamic system. Although each neuron is primitive both in architecture and in function, a network comprising many neurons is intricate. In addition to its nonlinear nature, neural network is a signal processing system. The inherent dynamic process can be classified as a fast process and a slow process. The former is a numerical process to evolve to an equilibrium status with given inputs. The latter is a learning process where the values of the connective weights between neurons are adjusted according to the environment. After learning, environmental information is stored on the connective weights.

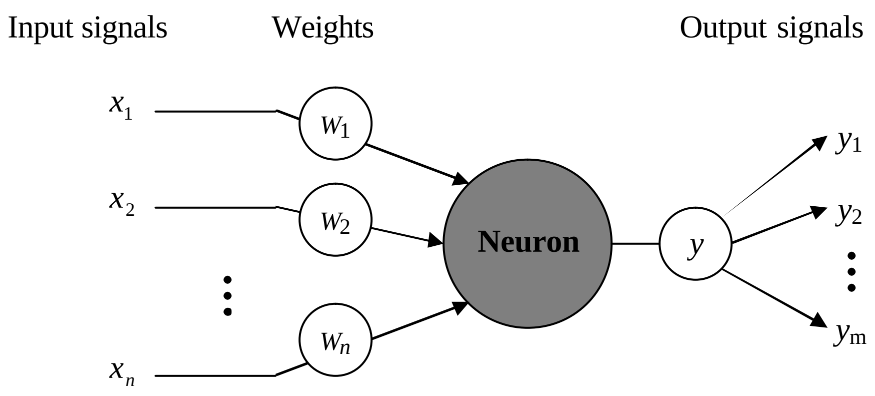

Figure 3.1 Diagram of a neuron

In 1943, McCulloch, a neurobiologist, and Pitts, a statistician, published a seminal paper [75] which inspired the development of the modern digital computer. At approximately the same time, Rosenblatt [76] was also motivated by this paper to investigate the computation of the eye, which eventually led to the first generation of artificial neural networks, known as the perceptron. Since then, the theory and design of ANNs have advanced significantly.

Over the last two decades, ANNs have found application in pattern recognition, signal process, intelligence control, system identification, optimization, etc. [48, 49, 77] because of their excellent learning capacity and their high tolerance to partially inaccurate data. They are suitable particularly for problems too complex to be modeled and solved by classical mathematics and traditional procedures. A good review article by Adeli [78] summarized the applications of the ANNs in civil engineering during the

20th century.

Artificial neural networks are typically characterized by their computational elements, their network topology, and the learning algorithm used. According to the learning approaches adopted, ANNs can be classified into two major groups: supervised and unsupervised. A supervised network is given both inputs and desired outputs pairs for training or learning. The network adjusts its weights until the errors between its outputs and the desired reach a predefined bound. An unsupervised network is commonly used for classification or clustering. Its weights are adjusted using predefined criteria until the network has performed a classification. Since these two types of networks are both adopted for signal processing and/or damage detection in this work, they are sequentially introduced in the following sections. The first introduced network model is a supervised neural network with L-BFGS learning algorithm, and the second one is an unsupervised neural network with fuzzy reasoning algorithm.

3.2 Supervised Neural Network With L-BFGS Learning Algorithm

3.2.1 Backpropagation Network (BPN)

Among the several different types of ANN, the feedforward, multilayered, supervised neural network with the error backpropagation algorithm, the backpropagation network (BPN) [79], is by far the most frequently applied neural network learning model, due to its simplicity.

The architecture of BP networks, depicted in Figure 3.2, includes an input layer, one or more hidden layers, and an output layer. Five components exist in such a network [48]: neuron (node); link; bias unit (threshold); weight; and transfer function. The nodes in each layer are connected to each node in the adjacent layer. Notably, Hecht-Nielsen [80] proved that one hidden layer of neurons suffices to model any solution surface of practical interest. Hence, a network with only one hidden layer is considered in this work.

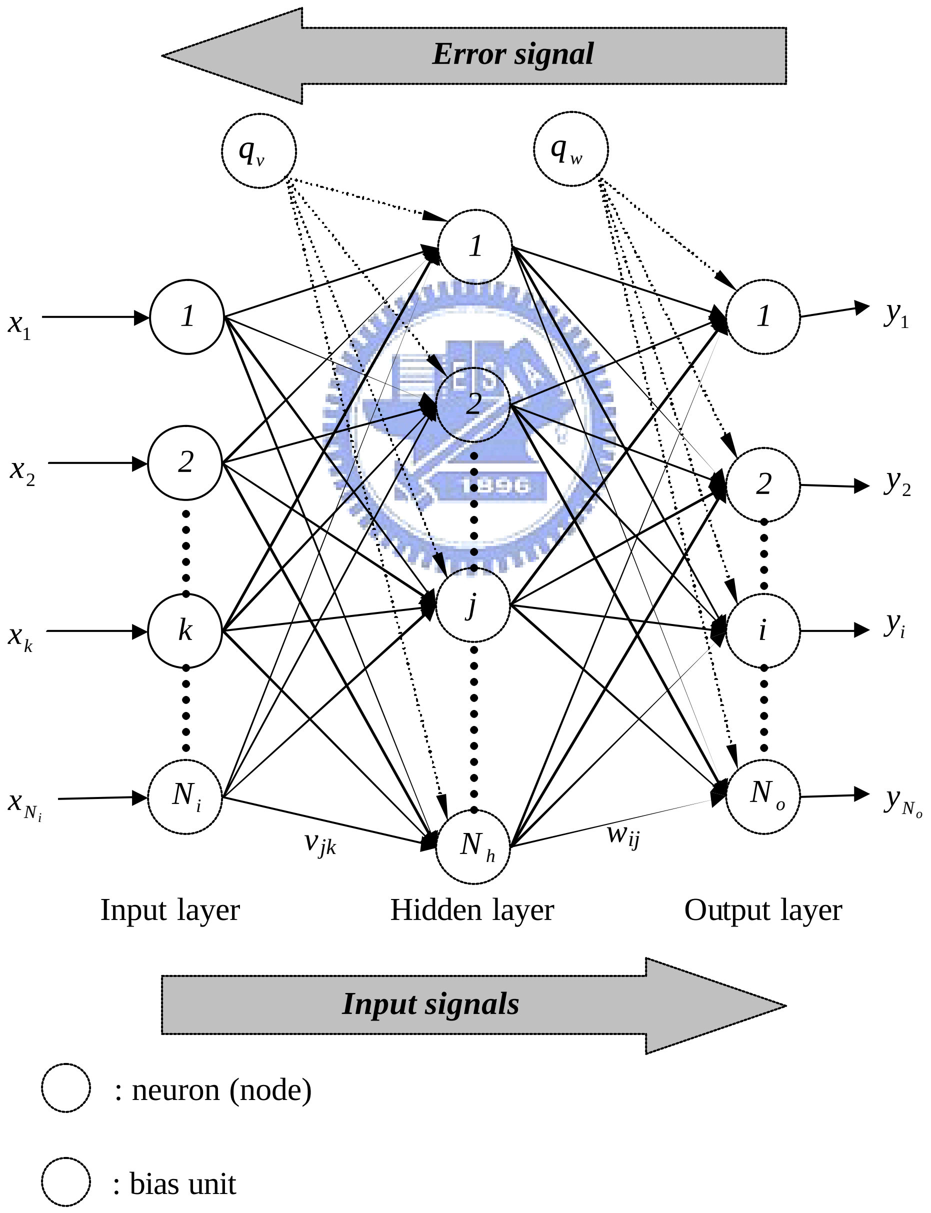

Figure 3.2 A typical three-layer neural network

Before an ANN can be used, it must be trained from an existing training set of pairs of input-output elements. The training of a supervised neural network using a BP learning algorithm normally involves three stages. The first stage is the data feed forward. The net input to a node (*j*) in hidden layer is

$$\text{net}_j = \sum_{k=1}^{N_i} v_{jk} x_k + q_{vj} \tag{3.1}$$

where v_{jk} is the connective weight between nodes in the input layer and those in hidden layer; x_k is the input of the *k*th node in input layer.; q_{nj} is bias term associated with the hidden layer; and N_i is the number of nodes in input layers.

Then the output to this node is

$$O_j = g(\text{net}_j) \tag{3.2}$$

where *g* is the transfer function in the node. The transfer function can be linear or nonlinear according to the problems of interest.

Similarly, the computed output of the *i*th node in output layer is defined as follows.

$$y_i = g\left(\sum_{j=1}^{N_h} \left(w_{ij}\, g\left(\sum_{k=1}^{N_i} v_{jk} x_k + q_{nj}\right) + q_{wi}\right)\right), \quad i = 1, 2, \cdots N_o \tag{3.3}$$

where w_{ij} is the connective weight between nodes in the hidden layer and those in the output layer; q_{wi} is bias term associated with the output layer. Terms N_h, and N_o are the number of nodes in hidden and output layers, respectively.

The second stage is error back-propagation through the network. During training, a system error function is used to monitor the performance of the network. This function is often defined as follows.

$$E(\mathbf{W}) = \frac{1}{2P}\sum_{p=1}^{P}(\tilde{\mathbf{Y}}_p - \mathbf{Y}_p)(\tilde{\mathbf{Y}}_p - \mathbf{Y}_p)^T \tag{3.4.a}$$

$$\tilde{\mathbf{Y}} = (\tilde{y}_1\ \tilde{y}_2 \cdots \tilde{y}_i \cdots \tilde{y}_{N_o}) \tag{3.4.b}$$

$$\mathbf{Y} = (y_1\ y_2 \cdots y_i \cdots y_{N_o}) \tag{3.4.c}$$

where **W** represents the collection of the weight matrices and the bias terms associated to the current network; and $\tilde{y}_i$ is the desired (or measured) value of output node *i*.

The final stage is the adjustment of the weights. The standard BP algorithm uses a gradient descent approach with a constant step length (learning ratio?to train the network.

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + \Delta\mathbf{W}^{(k)} \tag{3.5}$$

$$\Delta\mathbf{W}^{(k)} = -h\frac{\partial E}{\partial \mathbf{W}^{(k)}} \tag{3.6}$$

where h is the constant, general learning ratio in the range, [0, 1]. The superscript index, (k), indicates the kth learning iteration.

Backpropagation supervised neural network learning models, however, always take an extended period to learn. Moreover, the convergence of a BP neural network is strongly depends upon the use of a learning rate (h). Herein, a more effective adaptive L-BFGS learning algorithm [81], based on the limited memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi Newton second-order method [82] with an inexact line search algorithm, is employed.

3.2.2 L-BFGS Learning Algorithm

In the conventional BFGS method, the approximation, $\mathbf{H}_{k+1,}$ to the inverse Hessian matrix of the error function, E, is updated by,

$$\begin{aligned}\mathbf{H}_{k+1} &= (\mathbf{I} - ?_k \mathbf{s}_k \mathbf{z}_k^T)\mathbf{H}_k(\mathbf{I} - ?_k \mathbf{z}_k \mathbf{s}_k^T) + ?_k \mathbf{s}_k \mathbf{s}_k^T \\ &\equiv \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + ?_k \mathbf{s}_k \mathbf{s}_k^T\end{aligned} \tag{3.7.a}$$

where

$$?_k = 1/\mathbf{z}_k^T \mathbf{s}_k \tag{3.7.b}$$

$$\mathbf{V}_k = \mathbf{I} - ?_k \mathbf{z}_k \mathbf{s}_k^T \tag{3.7.c}$$

$$\mathbf{s}_k = \mathbf{W}^{(k+1)} - \mathbf{W}^{(k)} \tag{3.7.d}$$

$$\mathbf{z}_k = \mathbf{g}_{k+1} - \mathbf{g}_k \tag{3.7.e}$$

$$\mathbf{g}_k = \frac{\partial E}{\partial \mathbf{W}^{(k)}} \tag{3.7.f}$$

Rather than forming the matrix $\mathbf{H}_k$ in the BFGS method, the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ first define and then implicitly and dynamically update the Hessian approximation using information from the preceding few iterations. Therefore, the final stage of adjusting weights in a supervised ANN is modified as follows.

$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} + a_k \mathbf{d}_k \tag{3.8}$$

The search direction is given by,

$$\mathbf{d}_k = -\mathbf{H}_k \mathbf{g}_k + b_k \mathbf{d}_{k-1} \tag{3.9}$$

$$b_k = \frac{\mathbf{z}_{(k-1)}^T \mathbf{H}_{(k-1)} \mathbf{g}_{(k-1)}}{\mathbf{z}_{(k-1)}^T \mathbf{d}_{(k-1)}} \tag{3.10}$$

The step length, a_k, is mathematically adapted during the learning process, according to the inexact line search algorithm. This algorithm is used rather than a constant learning ratio in the L-BFGS learning algorithm. It is based on three sequential operations bracketing, sectioning, and interpolation. The bracketing operation brackets the potential step length, a, between two points, through a series of function evaluations. Sectioning then takes the two points of the bracket, for example a_1 and a_2, as the initial points, reduces the step size piecemeal, and determines the minimum between the points, to a desired degree of accuracy. Finally, quadratic interpolation approach takes the three points, a_1, a_2, and $(a_1+ a_2)/2$, to fit a parabola and thus determine the step length, a_k. Accordingly, the step length, a_k, must satisfy the following conditions in each iteration.

$$E(\mathbf{W}^{(k)} + a_k \mathbf{d}_k) \le E(\mathbf{W}^{(k)}) + b\,a_k (\nabla E(\mathbf{W}^{(k)})^T \mathbf{d}_k) \qquad b \in (0,1) \;\; and \;\; a_k > 0 \tag{3.11.a}$$

$$\nabla E(\mathbf{W}^{(k)} + a_k \mathbf{d}_k)^T \mathbf{d}_k \ge q (\nabla E(\mathbf{W}^{(k)})^T \mathbf{d}_k \qquad q \in (b,1) \;\; and \;\; a_k > 0 \tag{3.11.b}$$

$$\nabla E(\mathbf{W}^{(k)} + a_k \mathbf{d}_k)^T \mathbf{d}_{(k+1)} < 0 \tag{3.11.c}$$

Hence, the problem of trial and error selection of a learning ratio in the conventional BP algorithm is avoided in the adaptive L-BFGS learning algorithm.

3.3 Unsupervised Fuzzy Neural Network Reasoning Model

The unsupervised fuzzy neural networks (UFN) reasoning model was proposed by

Hung and Jan [83]. This model had been successful applied to the problems of preliminary design [84, 85] and control of building structures [86], and is further applied to the damage detection of structures [87]. The UFN reasoning model consists of an unsupervised neural network with a fuzzy computing process. The basic concept of the proposed model is that, the solution of a new instance can be solved by retrieved the similar instances from a collection of solved instances, named as ***instance base***, to a specified domain. The following is a brief review of the UFN reasoning model.

The UFN reasoning model is implemented in the following steps (Figure 3.3):

(i) measuring the similarities between new instance and existed instances;

(ii) generating the fuzzy set of similar instances; and

(iii) synthesizing the solution based on the fuzzy set of similar instances.

The three steps are introduced in the following sub sections.

3.3.1 Similarity Measurement

The first step involves searching for instances that similar to the new instance (**Y**) in the instance base (**U**$_j$) according to their inputs (Y_i and $U_{j,i}$). It is performed through a single-layered laterally-connected network with an unsupervised competing algorithm.

The similarity measurement is implemented by calculating the degree of difference between two instances. The function of degree of difference is defined as

$$d_{Yj} = diff(Y_{i,}U_{ji}) = \sum_{m=1}^{M} a_m (x^m - u_j^m)^2 \tag{3.12}$$

where d_{Yj} denotes the discrepancy between the inputs of the new instance **Y** and the jth instance **U**$_j$ in the instance base; a_m denotes predefined weight which is used to represent the degree of importance for the mth decision variable in the input.

After the values of d_{Yj} for all instances are calculated, the degree of similarity of instances **Y** and **U**$_j$ can be derived by the following fuzzy membership function.

$$m_{Yj} = f(d_{Yj}, R_{\max}, R_{\min}) = \begin{cases} 0 & if \quad d_{Yj} \geq R_{\max} \\ \dfrac{R_{\max} R_{\min} - R_{\min} d_{Yj}}{(R_{\max} - R_{\min}) d_{Yj}} & if \quad R_{\min} < d_{Yj} < R_{\max} \\ 1 & if \quad d_{Yj} \leq R_{\min} \end{cases} \tag{3.13}$$

The terms $R_{\max}$ and $R_{\min}$ define the upper and lower bounds of the degree of difference. In case the degree of difference is less than the upper bound $R_{\max}$, any two instances are treated as similar in some measure. Obviously, Equations (3.12) and (3.13) show that the smaller the discrepancy d_{Yj} is, the higher is the degree of similarity.

It is seen from equation (3.13) that the upper bound $R_{\max}$ heavily influences the measurement of similarity. A large $R_{\max}$ implies a loose similar relationship between instances. Consequently, a large number of instances are considered as similar instances. Since the solution of the new instance is based on the similar instances been taken, a selection of large $R_{\max}$ could result in inferior solution. On the other hand, a small $R_{\max}$ indicates that a strict similar relationship is adopted. Accordingly, most of the instances in the instance base are sorted as dissimilar to the new instance, and the UFN reasoning model could generate no solution. Therefore, a linear correlation analysis is employed to systematically determine the appropriate value of $R_{\max}$. More details about the correlation analysis process can be seen in Appendix A [85, 88].

3.3.2 Fuzzy Set Generation

The second step entails representing the fuzzy relationships among the new instance and its similar instances. The instances with the degree of difference smaller than $R_{\max}$ (the fuzzy membership value larger than zero, in other words) are extracted from the instance base as similar instances. Subsequently, the fuzzy set of "similar to **Y**" is then formed with the similar instances and their corresponding fuzzy membership values and is expressed by

$$S_{\sup,Y} = \{S_1(m_1), S_2(m_2), \ldots, S_p(m_p), \ldots\} \tag{3.14}$$

where S_p is the pth similar instance to instance **Y;** and m_p is the corresponding fuzzy membership value.

3.3.3 Solution Synthesis

Finally, the solution for instance **Y** is generated by synthesizing the outputs of similar instances according to their associated fuzzy membership value through the center of gravity (COG) method. As a result, the output Y_o of instance **Y** via the COG method is defined as follows:

$$Y_o = \frac{\sum_{k=1}^{p} m_k S_{k,o}}{\sum_{k=1}^{p} m_k} \tag{3.15}$$

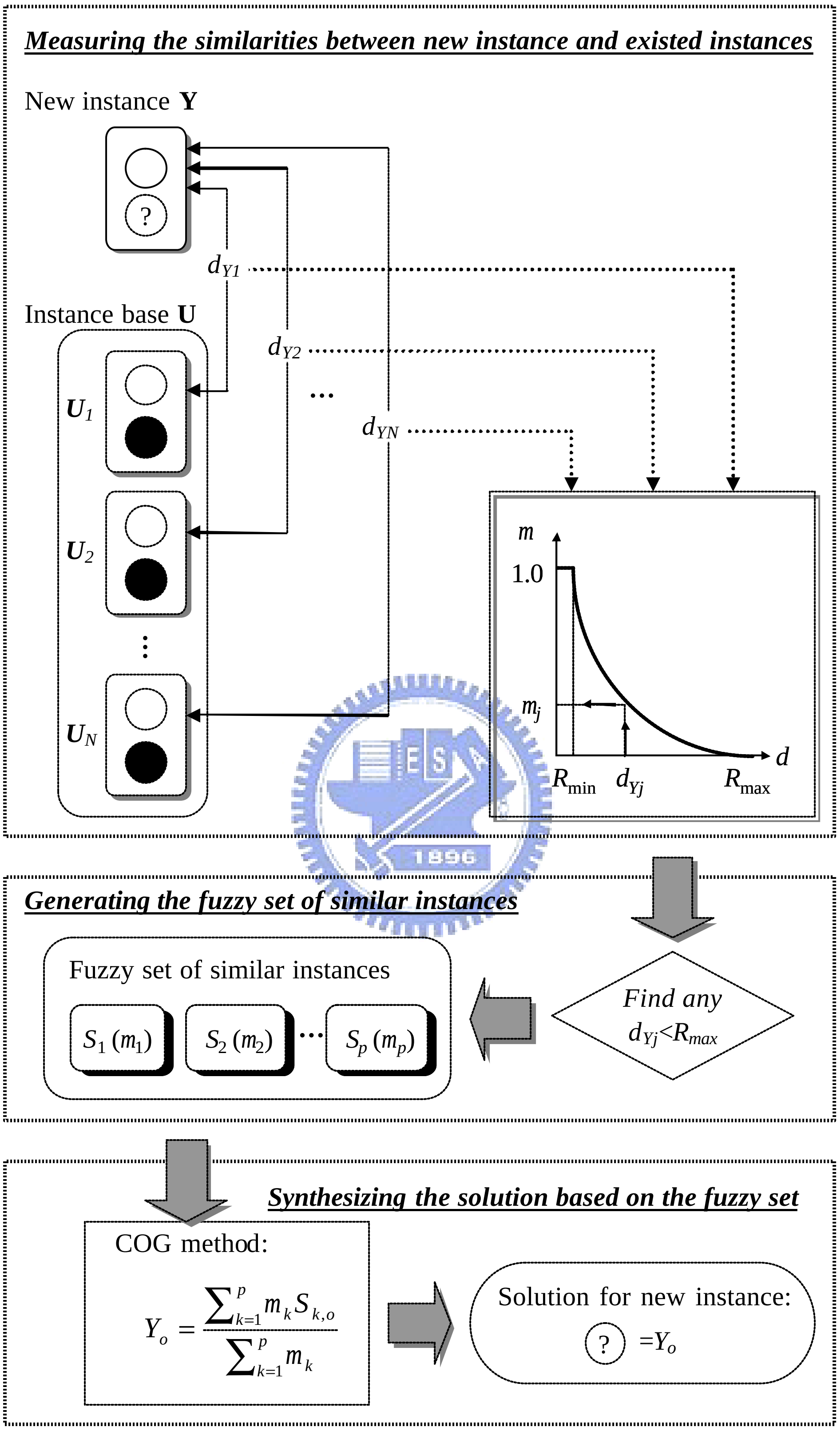
Measuring the similarities between new instance and existed instances
New instance Y
?
d_{Y1}
Instance base U
d_{Y2}
…
d_{YN}
U_1
U_2
U_N
m
1.0
m_j
d
R_{min}
d_{Yj}
R_{max}
Generating the fuzzy set of similar instances
Fuzzy set of similar instances
S_1 (m_1)
S_2 (m_2)
S_p (m_p)
Find any d_{Yj}<R_{max}
Synthesizing the solution based on the fuzzy set
COG method:
$Y_o = \frac{\sum_{k=1}^{p} m_k S_{k,o}}{\sum_{k=1}^{p} m_k}$
Solution for new instance:
? $=Y_o$

Figure 3.3 Process of the UFN reasoning