United States Department of Agriculture

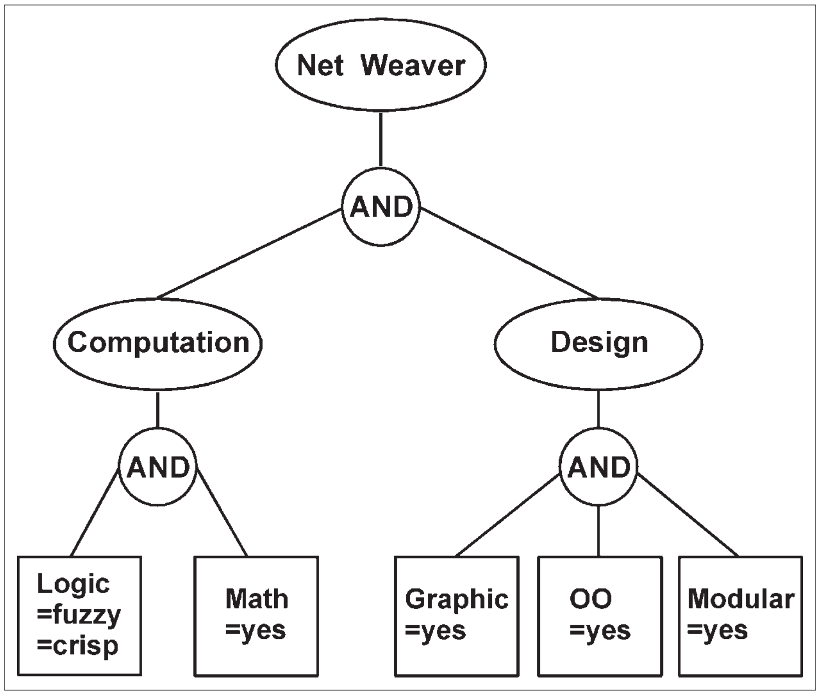
Forest Service

Pacific Northwest Research Station

General Technical Report PNW-GTR-471 October 1999

NetWeaver for EMDS User Guide (Version 1.1): A Knowledge Base Development System

Keith M. Reynolds



Author

KEITH M. REYNOLDS is a research forester, Forestry Sciences Laboratory, 3200 SW Jefferson Way, Corvallis, OR 97331.

Abstract

Reynolds, Keith M. 1999. NetWeaver for EMDS user guide (version 1.1): a knowledge base development system. Gen. Tech. Rep. PNW-GTR-471. Portland, OR: U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station. 75 p.

The guide describes use of the NetWeaver knowledge base development system. Knowledge representation in NetWeaver is based on object-oriented fuzzy-logic networks that offer several significant advantages over the more traditional rule-based representation. Compared to rule-based knowledge bases, NetWeaver knowledge bases are easier to build, test, and maintain because the underlying object-based representation makes them modular. The modularity of NetWeaver knowledge bases, in turn, allows designers to gradually evolve complex knowledge bases from simpler ones in small, simple steps. Modularity also allows interactive knowledge base debugging at any and all stages of knowledge base development, which expedites the development process. Finally, fuzzy logic provides a formal and complete calculus for knowledge representation that is less arbitrary than the confidence factor approach used in rule-based systems and much more parsimonious than bivalent rules.

Keywords: NetWeaver, knowledge base, fuzzy logic, decision support.

Contents

System Requirements

Component	Required	Recommended
Input device	Mouse	
Operating system	Windows 3.1	Windows 95 or Windows NT 4.0
Processor	486 CPU	Pentium CPU (90 MHz or better)
RAM	16 MB	
Screen resolution	640 by 480 pixels	800 by 600 pixels or better
Color resolution	256 colors	

Text Formatting Conventions

The following conventions are used for text formatting in this user guide:

PLAIN UPPERCASE TEXT indicates a folder (directory) or file name.

Italic text indicates a graphic object such as a button, menu choice, etc.

Plain bold text is used for emphasis and cross references.

Installation and Setup

NetWeaver is distributed by the USDA Forest Service as a component of the Forest Service version of the EMDS application (Reynolds and others 1996). If EMDS has been installed on your system, then NetWeaver is already installed as well.

Introduction

NetWeaver for EMDS (hereafter, NetWeaver) is a knowledge base development system developed for the Microsoft Windows® environment that provides a graphical environment in which to construct and evaluate knowledge bases.[1]

Knowledge base systems come in a variety of forms, but the dominant type currently in use is rule-based systems (Barr and others 1989, Jackson 1990, Schmoldt and Rauscher 1995). Knowledge representation in NetWeaver, in contrast, is based on object-oriented fuzzy-logic networks that offer several significant advantages over the more traditional rule-based representation. Compared to rule-based knowledge bases, NetWeaver knowledge bases are easier to build, test, and maintain because the underlying object-based representation makes these types of knowledge bases modular. The modularity of NetWeaver knowledge bases, in turn, allows the designer to gradually evolve complex knowledge bases from simpler ones in small, simple steps. Modularity also allows interactive knowledge base debugging at any and all stages of knowledge base development, which expedites the development process. Finally, fuzzy logic provides a formal and complete calculus for knowledge representation that is less arbitrary than the confidence factor approach used in rule-based systems and much more parsimonious than bivalent rules.

A Brief History

NetWeaver was developed at Pennsylvania State University by Michael Saunders and Bruce Miller. Initial development of the system began in 1987 and has steadily

[1] The use of trade or firm names in this publication is for reader information and does not imply endorsement by the U.S. Department of Agriculture of any product or service.

evolved since then. NetWeaver for EMDS was reengineered by Knowledge Garden, Inc., in the KnowledgePro language from the latest Pennsylvania State University version. Reengineering by a commercial software developer was undertaken primarily to enhance intuitiveness of the system interface. KnowledgePro was chosen in particular because its native hypermedia capabilities are well suited to integrating extensive help system functions into the application.

What is a Knowledge Base?

In recent times, the phrase "knowledge base" has come into popular usage and now generally means a body of knowledge about some problem domain. Originally, however, the term had a more precise meaning as a formal logical specification for the interpretation of information (Waterman 1986); that is, a knowledge base is a body of knowledge that has been organized within a formal syntactic and semantic framework that allows formal inferences about a problem. The term "knowledge base" is used throughout the NetWeaver user guide in this more formal sense.

Components of a Knowledge-Base System

By itself, a knowledge base does not actually do anything. Instead, it is a metadatabase used (by something or someone) to interpret data (Waterman 1986). In the case of a knowledge base system, the "something" is an inference engine, a program library that interprets external data according to the semantics built into a knowledge base by the knowledge base designer.

One more component is needed for a complete knowledge base system because the inference engine is just a program library. This final component is the interface, a program that allows the user to directly interact via screen objects in the interface to construct a knowledge base and to direct the processing of data by the knowledge base engine. The interface acts as an intermediary between the user and the engine. During construction of a knowledge base, for example, the inference engine builds the knowledge base in response to mouse and keyboard operations directed by the user via the interface. In the process of constructing a knowledge base, the engine enforces and constantly checks user actions to ensure that the content and structure of the knowledge base conforms to the syntax that the system requires. The user also interacts with the inference engine via the interface to control how data are evaluated.

Fuzzy Logic

Although the term "fuzzy logic" has a distinctly esoteric ring to it, the concept is actually fairly simple (Zadeh 1965). Fuzzy logic provides a metric for expressing the degree to which an observation about some variable belongs to a set that represents a concept. Alternatively, one might say that fuzzy logic is concerned with "aboutness." To make the concept clear, consider a simple example.

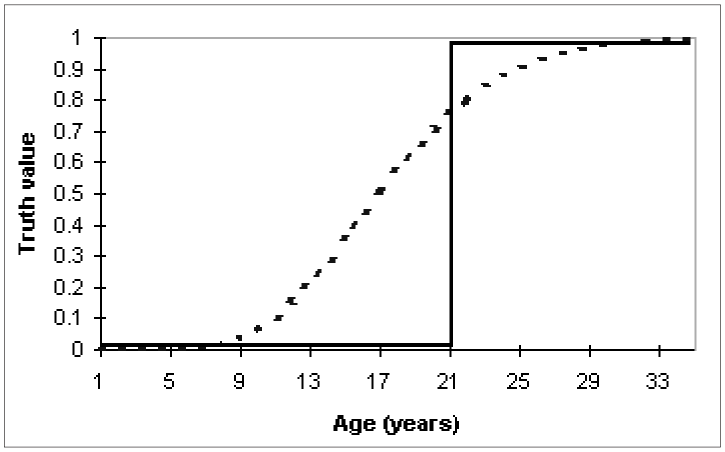


Figure 1—Representation of adultness with a bivalent argument versus a fuzzy argument.

Everyone has some concept of what it means to be an adult. For legal purposes, an adult, in western cultures at least, is often defined as a person who is 21 years old or older. A rule-based system dealing with legal issues can easily accommodate this bivalent definition: if a person is 20 years, 364 days old, he or she is not an adult, but if the person is one or more days older, he or she is an adult (fig. 1). This characterization of adultness is sufficient if the concept is limited to a simplistic legal one. However, if by adultness the real interest is to express something more complex, such as an individual's emotional maturity, then the simple bivalent rule is no longer adequate. Most people would agree that a 5-year-old has no, or at best minimal, adult qualities. In a 13-year-old, however, some early signs of adult characteristics may be seen. Many 18-year-olds demonstrate adult qualities (they act "grown up"). Conversely, most people can think of at least a few 45-year-olds who could not be called emotionally mature or adult. So, as a first step toward improving the characterization of adultness, one might construct a simple (fuzzy) curve that translates age into degree of membership in the set "adult" (fig. 1).

At the beginning of the chapter, it was indicated that fuzzy logic allows a more parsimonious knowledge representation than is possible with rule-based systems. The reason is simple. A single fuzzy curve is sufficient to express the full spectrum of adultness. In contrast, rule-based systems are inherently bivalent, meaning that a rule is either true or false. To more precisely characterize adultness in a rule based system, one would need to define, say, five age categories corresponding to different levels of adultness, and each category would require a rule. Moreover, if that rule base also dealt with intelligence and this attribute similarly had five categories (ranging from brilliant to dumb as a doorknob, for example), then to jointly consider both adultness and intelligence in the rule base could require as many as 25 additional rules. In contrast, in a fuzzy logic-based representation of this situation, only one more fuzzy curve and perhaps a new network object to jointly evaluate the two fuzzy curves are needed. So, in the example, two fuzzy curves have an expressive power that is equal to or better than 35 (10 + 25) rules. To summarize, the number of rules needed to adequately represent possible outcomes increases approximately exponentially, whereas the number of fuzzy curves and related objects needed to describe the same problem in an object oriented fuzzy logic representation increases linearly.

The NetWeaver Object Model

NetWeaver Components

A NetWeaver knowledge base has the following components:

- Dependency networks that represent topics of interest in a problem domain that are to be evaluated.
- Data links that request and evaluate data.
- Nodes that specify logical or mathematical relations among dependency networks and data links.
- Optionally, evaluation groups that represent collections of dependency networks and data links.

Each component is described in greater detail in later chapters. These components are introduced here to provide a general perspective of the NetWeaver architecture. Dependency networks (called networks hereafter) and data links are problem-specific NetWeaver objects that have both state and behavior. Nodes are predefined NetWeaver objects that also have state and behavior; their function in a knowledge base is to define the logical and mathematical dependencies among the problem-specific objects created by the user. Finally, evaluation groups are not objects at all, but collections of objects (usually dependency networks) defined by the user to conveniently view the evaluation of several things at one time. Members of an evaluation group can, but do not need to, have formally defined logical relations to one another.

The Object Concept

NetWeaver is a rigorous object-based system (Saunders and others 1989, 1990). Although it is not necessary to understand the NetWeaver object model in great detail to effectively use the system, a basic understanding of it makes the process of constructing a knowledge base more intuitive.

In NetWeaver, dependency networks and data links are programming objects that represent real-world objects or concepts. Data links typically request data about a real-world object, such as forest stand age, and evaluate the data. Dependency networks typically represent more abstract objects, such as elk habitat suitability, which are evaluated in terms of simpler abstract concepts and data. Regardless of how concrete or abstract real-world data and concepts may be, they can be thought of as objects; that is, things to be logically or mathematically manipulated. The key point here is that a one-to-one relation exists between real world objects and user-defined NetWeaver objects that makes knowledge representation in NetWeaver relatively straightforward and intuitive.

Although a collection of objects by itself is not very useful, the objects can be organized into a knowledge base from which useful conclusions can be drawn if logical or mathematical dependencies are specified among them. To construct a knowledge base in NetWeaver, one defines the objects of interest in the problem domain and the logical or mathematical dependencies among those objects.

A final point about objects is that, in a programming sense, they are reusable. That is, having once constructed an object for use in a particular knowledge base, it can be used many times simply by referring to it. The practical application of object reuse in NetWeaver is that a knowledge base developer needs to define an object only once, after which it can be referenced as many times as required.

The Network Concept

The most important object in a NetWeaver knowledge base is the dependency network (Stone and others 1986) because of its central role in formulating a representation of the problem. Everyone has at least an intuitive understanding of the concept of networks because there are many familiar real-world examples, such as road networks that interconnect towns and cities. In general, networks encapsulate understanding of how things are connected to one another.

A hierarchy is a special case of a network, and readers will have some intuitive understanding of hierarchies because they are routinely encountered even if they are not given much thought. Typological hierarchies are extremely common. For example, in a hierarchy of vehicles, an automobile is a type of vehicle, a truck is a type of automobile, a pickup is a type of truck, etc. A hierarchy is a special case of a network because the definition of a hierarchy imposes some topological restrictions on how objects can be connected. In the vehicle hierarchy, for example, pickups belong only to the truck superclass. Generally, hierarchical structures require that subclasses be connected only to one superclass for each instance in which they appear in the hierarchy.

Consider another typological hierarchy that also contains pickups. If the two hierarchies are linked, the composite violates the topological rule that a subclass can be connected to only one superclass for any instance of the subclass. This new composite structure is a network but not a hierarchy. NetWeaver knowledge bases are constructed similarly. A NetWeaver network structure is a simple hierarchy, but each such hierarchy can be interconnected with arbitrarily many other hierarchies at arbitrarily many points in their structure. This interconnectivity among hierarchies is a network topology that provides a powerful mechanism for evaluating complex dependencies among objects of interest in the real world.

Building Knowledge Bases With NetWeaver

Uses of Dependency Networks

Knowledge-based reasoning is a general modeling methodology in which phenomena are described in terms of abstract entities and their logical relations to one another. There are two basic reasons for using knowledge based reasoning:

- The entities or relations involved in the problem to be solved are inherently abstract so that mathematical models of the problem are difficult or perhaps even impossible to formulate.
- A mathematical solution is possible in principle, but current knowledge is too imprecise to formulate an accurate mathematical model.

Both cases are quite common. The first case naturally arises when the nature of the problem involves relatively abstract entities. These problems may be easier to solve with logic. The second case arises particularly when dealing with ecosystems, because there is an almost unlimited number of relations of potential interest. Agencies, academia, and others have developed numerous mathematical models to describe some of the important relations of interest to ecosystem management, but many relations have not been studied in sufficient detail to provide generally applicable mathematical models. Often, a wealth of human experience is available, however, in these same institutions that can be drawn on to develop useful, more qualitative, knowledge-based models to guide decisionmaking.

Another valuable aspect of knowledge-based reasoning that makes it ideal for use in environmental assessment is that such systems can reason with incomplete information. The NetWeaver knowledge base engine used in EMDS, for example, provides partial evaluations of ecosystem states and processes based on available information, and provides useful information about the influence of missing data that might be needed to improve an assessment.

NetWeaver Knowledge Bases and EMDS

In the context of the EMDS system, a NetWeaver knowledge base represents relations among concerns, ecosystem states and processes, and data requirements. Dependency networks in EMDS are used to:

1. Evaluate the truth value of assertions about ecosystem states and processes given existing data.

2. Identify data requirements for an analysis.
3. Rank missing data in order of relative importance to the analysis.

One of the virtues of a dependency network representation is that a single knowledge base may incorporate a wide variety of topics. This is particularly valuable in the context of ecological assessments, which EMDS is designed to support and in which topics of interest might include, for example, many different topics and subtopics related to terrestrial vegetation and wildlife, fish populations, recreation, water and air quality, aesthetic concerns, and commercial concerns. The number of topics and their interrelations that can be represented in a knowledge base are limited only by a computer's dynamic memory.

Evolving Knowledge Bases

NetWeaver is a rigorous object-based development system for knowledge base design, as discussed in the previous section. One of the more practical implications of object-based knowledge bases is that it is easy to start with a simple knowledge representation that gradually evolves into a large complex system, because object-based knowledge bases are extremely modular. A basic modeling principle that has motivated development and application of object-oriented technology in general is well captured by Gall (1986):

> A complex system that works is invariably found to have evolved from a simple system that worked.…A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.

NetWeaver is designed to build a knowledge base in an incremental and evolutionary way. The best possible advice for the novice user is to avoid designing large complex systems at the outset. Instead, start with a simple knowledge base built from a small number of dependency networks, and gradually develop this simple representation into a more complex representation of the problem.

It is usually best to start at the top with the highest level (primary) dependency networks that apply to the problem domain and develop the structure downward. To get started, create and document at least a few of the primary dependency networks (see the later sections "Creating Network Objects" and "Documenting a New Network Object"). Don't be too concerned about identifying an exhaustive list of primary networks at the outset. Because of the modular structure of NetWeaver knowledge bases, new dependency networks easily can be added later in development without upsetting the overall structure of the knowledge base.

For each primary network in the knowledge base, create antecedents. Unless it is an unusually simple knowledge base, at least one or two levels of antecedents usually will occur before a chain of dependencies terminates in a data link.

In a completed knowledge base, you will want to be sure that each chain of dependencies terminates in a data link. However, even though a knowledge base is under development, it is always possible to evaluate a network object, regardless of how complete the network structure is.

As the structure of a knowledge base evolves, occasions likely will occur when existing antecedents (both dependency networks and data links) can be used. Multiple occurrences of dependency networks and data links in a single knowledge base are not a problem because NetWeaver objects are reusable. The presence of a dependency network, in particular, in two or more other networks within the same knowledge base is an important mechanism interrelating networks.

Organization of the User Guide

Writing a user guide for an object-oriented application designed to build object-oriented knowledge bases presents certain challenges. In particular, the traditional cookbook approach often used for user guides is not sufficient by itself because cookbooks have a basic procedural orientation (first one does this, then one does that, and so on). In an application such as NetWeaver, however, the primary orientation is not the procedure, but the object. It is best to describe NetWeaver from both a procedural and an object perspective in subsequent chapters:

- “Getting Started” describes basic elements of the NetWeaver interface and file handling procedures related to designing knowledge bases.
- “Working With NetWeaver Objects” provides a procedural overview of methods for working with NetWeaver objects. In contrast to the final two chapters, the primary organizational unit in this chapter is the procedure.
- “NetWeaver Objects” is the first of two reference chapters. It describes the two central NetWeaver objects, dependency networks and data links, as well as evaluation groups, which formally are not objects, but collections of objects. The primary organizational unit of subsections in this chapter is the object or specializations of the object. Each description includes detailed procedures for using the NetWeaver interface to work with the object.
- “Relations Among Objects” is the second reference chapter and describes the logical nodes, which are themselves objects, that are used to build the dependency structure of a knowledge base. As with the previous chapter, the primary organizational unit is the object.

Getting Started

Launching NetWeaver

Windows 95 or NT

At installation, NetWeaver is installed under the EMDS program group on the *Task bar*. To launch the application from the Windows 95 *Task bar*:

1. Press the *Start button* on the *Task bar*.
2. Choose *Programs* from the *Start menu*.
3. Choose *EMDS* from the *Programs menu*.
4. Choose *NetWeaver* from the *EMDS menu*.

NetWeaver also can be launched from Windows Explorer by double clicking on the NetWeaver *application icon* in the \EMDS\BIN folder in an Explorer window.

Windows 3.1

At installation, NetWeaver is installed in the EMDS program group window in Program Manager. To launch the application from Program Manager:

1. Double click on the EMDS group icon to open the group window.
2. Double click on the NetWeaver *program item icon* in the group window.

NetWeaver also can be launched from File Manager by double clicking on the NetWeaver application name in the \EMDS\BIN directory in the File Manager window.

Online Documentation

The Main Help System

The NetWeaver help documentation is an online version of the user guide (fig. 2). To use online help, choose the *Help menu* from the *Application menu* bar (fig. 3), then choose one of the following help topics from the *Help menu*:

- Contents
- Search
- How to
- Using Help
- About

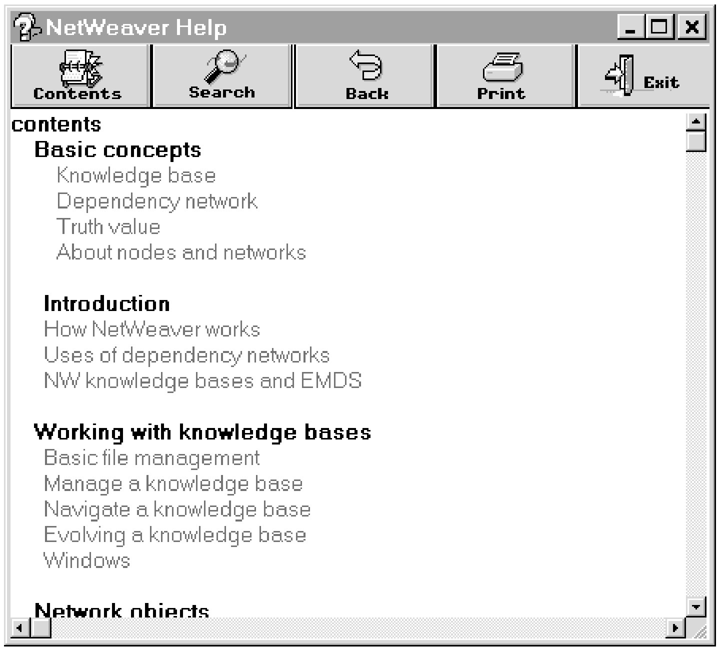
All text highlighted in blue within the Help System is hypertext that can be selected to move to the indicated topic. The following *buttons* are displayed across the top of the Help System window:

- *Contents* returns to the table of contents.
- *Search* displays an index of help topics (see “Searching for Help Topics,” below).
- *Back* moves back one topic in the list of help topics that have been called (the help system maintains a list of topics that you have visited).
- *Print* prints the currently displayed help topic to the default printer.
- *Help* displays help about using the Help System.
- *Exit* quits the Help System.

Searching for Help Topics

When the *Search button* is pressed (fig. 2), the Help System displays the *Search dialog window* (fig. 4). A search topic can be typed in the *edit line* at the top of the window. As the word is typed, the contents of the *Search topics list box* scrolls to matching entries in the search index, and the first match is highlighted.

The *Search topics list box* also can be scrolled with the *button* on the *list box scroll bar*. Select a topic from the index with the mouse. When the desired topic is highlighted, press the *OK button* to go to the topic. Press the *Cancel button* to abort the search and return to the previous help topic.



contents

Basic concepts

Knowledge base
Dependency network
Truth value
About nodes and networks

Introduction

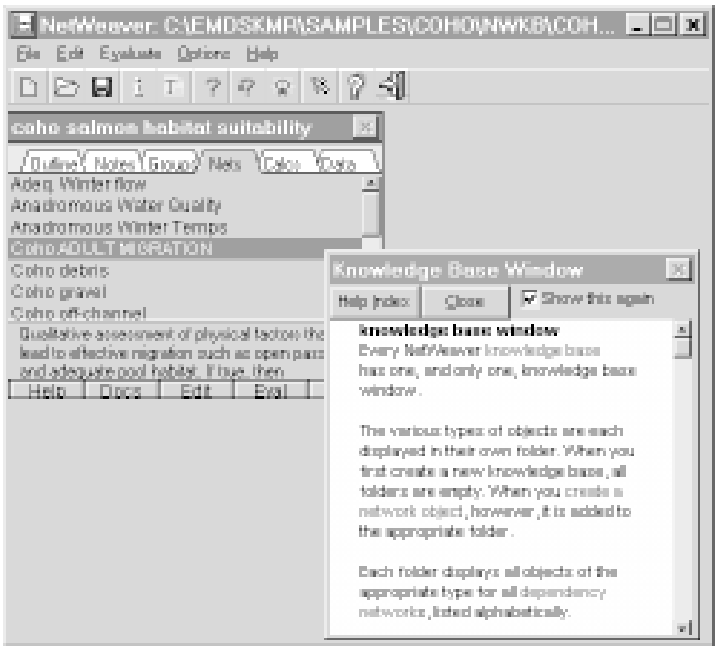
How NetWeaver works
Uses of dependency networks
NW knowledge bases and EMDS

Working with knowledge bases

Basic file management
Manage a knowledge base
Navigate a knowledge base
Evolving a knowledge base
Windows

Network objects

Figure 2—The main NetWeaver help system.



Adeq. Winter flow
Anadromous Water Quality
Anadromous Winter Temps
Coho ADULT MIGRATION
Coho debris
Coho gravel
Coho off-channel

Qualitative assessment of physical factors the
lead to effective migration such as open pass
and adequate pool habitat. If true, then

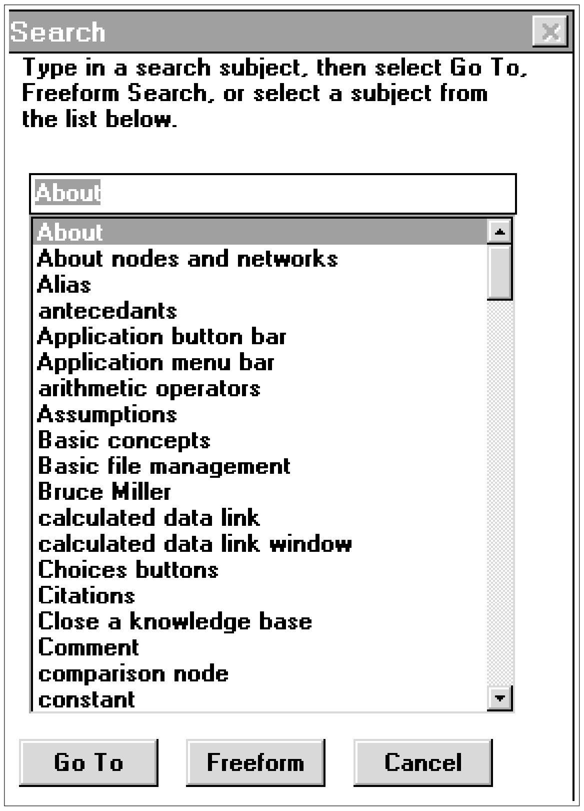
knowledge base window

Every NetWeaver knowledge base has one, and only one, knowledge base window.

The various types of objects are each displayed in their own folder. When you first create a new knowledge base, all folders are empty. When you create a network object, however, it is added to the appropriate folder.

Each folder displays all objects of the appropriate type for all dependency networks, listed alphabetically.

Figure 3—The main application window of NetWeaver with cue card help enabled.


Search

Type in a search subject, then select Go To, Freeform Search, or select a subject from the list below.

About

About
About nodes and networks
Alias
antecedants
Application button bar
Application menu bar
arithmetic operators
Assumptions
Basic concepts
Basic file management
Bruce Miller
calculated data link
calculated data link window
Choices buttons
Citations
Close a knowledge base
Comment
comparison node
constant

Figure 4—Search dialog window of the NetWeaver help system.

Cue Card Help

In addition to the standard help system, NetWeaver can display cue cards that provide abbreviated instructions about how to work with the dialog window that was most recently opened in the main application window of NetWeaver (fig. 3). Selecting hypertext in cue cards leads to a detailed explanation of the selected help topic in the main NetWeaver help system.

Cue card help is turned on by default when NetWeaver is initially installed. A check mark next to the *Cue card item* on the *Options menu* of the *Application menu bar* indicates that cue card help is enabled. To completely disable the cue card feature, choose the *Cue card item* on the *Options menu* to remove the check mark next to the option. Select the *Cue card item* again to fully restore cue card help.

The cue card associated with each dialog window can be turned off indefinitely by clicking on the *check box* labeled “Show this again” in the cue card window to remove its check mark. The cue card for the associated dialog window will not be shown again until the *check box* is reselected on the cue card. To temporarily restore cue card help for the dialog window currently active in the main application window, press the *Help button* in that window. To permanently restore cue card help for the current dialog window, reselect the *check box* labeled “Show this again” so that it is again checked. To permanently restore automatic display of all cue cards that have been turned off, first deselect the *Cue card item* on the *Options menu* to turn off all cue card help, then reselect the *Cue card item*. This procedure resets all cue cards to the enabled state.

Basic File Management

Create a New Knowledge Base

To create a new knowledge base, choose *New* on the *File menu* of the *Application menu bar*, or press the *New file button* on the *Application button bar* (fig. 3). Either action opens a new *Knowledge base window*. If a knowledge base is already open, NetWeaver first prompts, asking if the currently opened knowledge base should be saved before opening the new one.

Open an Existing Knowledge Base

To open an existing knowledge base, choose *Open* on the *File menu* from the *Application menu bar* or press the *Open file button* on the *Application button bar*. Either action opens a *Filename dialog window* from which to select a NetWeaver knowledge base file to open. By default, the *dialog window* displays all knowledge base files in the \project name\NWKB folder. If a knowledge base is already open, NetWeaver first prompts, asking if the currently opened knowledge base should be saved before opening the new one.

Save a Knowledge Base

To save the current state of a knowledge base, choose *Save* on the *File menu* of the *Application menu bar* or press the *Save file button* on the *Application button bar*. To save the knowledge base under a new file name, choose *Save as* on the File menu.

Close a Knowledge Base

To close the knowledge base currently open in NetWeaver, choose *Close* on the *File menu*. If any changes have been made prior to last saving the knowledge base, a dialog window will open and prompt to save changes. Choose the *Yes* option to save the changed version, or choose *No* to discard the changes.

Revert to Saved Version of Knowledge Base

Choose *Revert to saved* on the *File menu* to discard any changes made to the currently open knowledge base, and open the last saved version of the knowledge base.

Print a Knowledge Base

To print an ASCII script of the contents of the knowledge base currently loaded in NetWeaver, choose *Print* on the *File menu*.

NetWeaver Interface

Application Button Bar

New button: Creates a new, empty knowledge base. If a knowledge base is already open and it has been edited since the last save operation, NetWeaver puts up a *dialog window*, asking if you want to save the currently open knowledge base first.

Open button: Opens a *file name dialog window* from which to select a knowledge base to open. The default directory location is the \project_name\-NWKB subdirectory in the C:\EMDS directory if default directory locations were used when EMDS was installed. If a knowledge base is already open and it has been edited since the last save operation, NetWeaver puts up a *dialog window*, asking if you want to save the currently open knowledge base first.

Save button: Saves the current state of a knowledge base. If the knowledge base has not previously been saved, then *Save* functions in the same way as *Save As* (see the later section on the *File menu* for important notes about saving knowledge bases).

Get info button: Opens the knowledge base *Information window* for editing the name of the knowledge base.

New Note button: Opens a *Notes window* to create a new note.

New Group button: Opens the *Documentation window* and then, after the window is closed by the user, the *Evaluation group window* to create a new evaluation group.

New Network button: Opens the *Documentation window* and then, after the window is closed by the user, the *Network window* to create a new dependency network.

New Calculated Data Link button: Opens the *Documentation window* and then, after the window is closed by the user, the *Data link window* to create a new calculated data link.

New Simple Data Link button: Opens the *Documentation window* and then, after the window is closed by the user, the *Data link window* to create a new simple data link.

Help button: Opens *help window* with help documentation for the currently active NetWeaver window.

Exit button: Closes the NetWeaver application window.

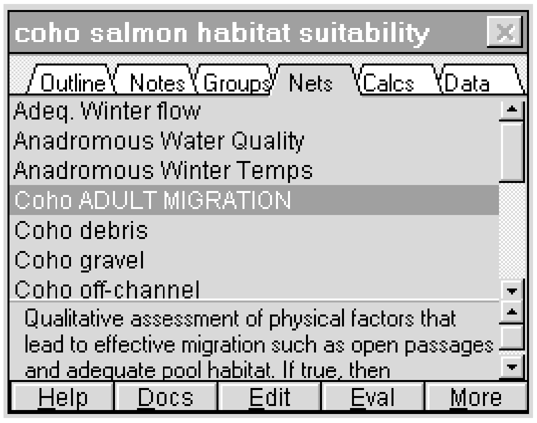
Knowledge Base Window

Every NetWeaver knowledge base has one, and only one, *Knowledge base window* (fig. 5). The various types of NetWeaver objects are each displayed in their own folder in this window. When a new knowledge base is first created, all folders are empty. Objects are added to the appropriate folder as soon as they are created.

The *Outline folder* displays the basic organizational structure of a knowledge base as a hierarchy of topics. Each of the remaining folders in the *Knowledge base window* displays an alphabetical list of all objects of the corresponding object type that have been created in the knowledge base.

***Outline folder*—**The *Outline folder* in the *Knowledge base window* displays the complete network structure of a knowledge base, decomposed into a simple hierarchy of network objects (fig. 5). This view of knowledge base structure is synoptic in the sense that it omits details of the logical relations among objects, but it is particularly useful as an overview of the complete hierarchical structure of a knowledge base.

The hierarchy displayed in the *Outline folder* can be expanded and collapsed. To expand the outline underneath a specific evaluation group or dependency network, click on the *right-pointing arrow icon* in front of the object name. After expanding a given group or network of the knowledge base hierarchy, the associated *arrow icon* changes to a *down-pointing arrow icon*, and the immediate antecedents of the group or network are displayed, indented underneath the item. To collapse the expanded hierarchy back to a specific network or group, click on its *down-pointing arrow icon*.



Adeq. Winter flow
Anadromous Water Quality
Anadromous Winter Temps
Coho ADULT MIGRATION
Coho debris
Coho gravel
Coho off-channel

Qualitative assessment of physical factors that lead to effective migration such as open passages and adequate pool habitat. If true, then

Figure 5—Knowledge base window.

Object folders in the Knowledge base window—Each of the remaining folders in the *Knowledge base window* displays an alphabetical list of all objects of the corresponding object type that have been created in the knowledge base (fig. 5). In addition to the *Outline folder*, there are specific *folders for Notes*, evaluation groups (*Groups*), networks (*Nets*), calculated data links (*Calcs*), and simple data links (*Data*).

Using the Knowledge base window in conjunction with menus and buttons— To perform an action on an object displayed in one of the folders of the Knowledge base window, click on the name of an object in a *folder* to highlight it, and then either choose one of the *menu items* on the *Edit menu* or *Evaluate menu*, or press one of the *buttons* on the *Knowledge base window button bar*.

Menu items that operate on selected objects are *Documentation*, *Edit object*, and *Delete* on the *Edit menu*, and *Evaluate object* on the *Evaluate menu*. For more detailed information about each *menu item*, see the later section, “Application menu bar.” The *menu bar* at the bottom of the *Knowledge base window* includes *buttons* labeled *Docs*, *Edit*, and *Eval* that duplicate the functions of the *Documentation*, *Edit object*, and *Evaluate object menu items*.

Viewing additional information about network objects—The *Knowledge base window* actually consists of three panes. The topmost pane displays the contents of the *Notes*, *Groups*, *Nets*, *Calcs*, and *Data folders*. A small pane immediately underneath displays any comments entered in the *Documentation window* that are associated with the selected object (fig. 5). Both panes are always visible.

A *details pane* that contains further information about objects selected in the folders of the topmost pane can be opened by pressing the *More button* on the *Knowledge base window button bar*. Folders in the *details pane* display information about objects that are antecedent to the selected object (*Antes*), objects that are dependent on the selected object (*Deps*), as well as selected information from the *Documentation window* (fig. 6).

Shortcuts—Double clicking on objects in the *object folders* of the topmost pane can be used as a shortcut for invoking certain operations:

- Double click on an evaluation group to display the *Evaluation window* and begin an evaluation of the group.
- Double click on a simple data link to display the *Data link evaluation window* and enter a data value for the data link or edit an existing value.
- Double click on a network to open the *Network window* and edit the structure of a network.
- Double click on a calculated data link to open the *Calculated data link window* and edit the functional structure of the data link.

Application Menu Bar

File menu—

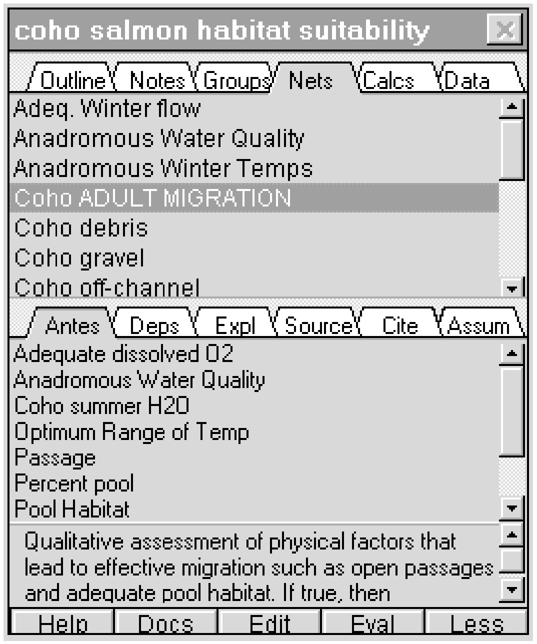
New: Creates a new, empty knowledge base. If a knowledge base is already open and it has been edited since the last save operation, NetWeaver displays a *dialog window*, asking if the open knowledge base should be saved.

Open: Displays a *File name dialog window* from which to select a knowledge base to open. The default directory location is the \projectname\NWKB. If a knowledge base is already open and it has been edited since the last save operation, NetWeaver displays a *dialog window*, asking if the open knowledge base should be saved.

Get info: Displays the knowledge base *Information window* for editing the name of the knowledge base.

Close: Closes the currently open knowledge base. If the current knowledge base has been edited since the last save, the NetWeaver prompts, asking if the knowledge base should be saved.

Save: Saves the current state of a knowledge base. If the knowledge base has not been saved previously, then *Save* functions like *Save As*, below.



Adeq. Winter flow
Anadromous Water Quality
Anadromous Winter Temps
Coho ADULT MIGRATION
Coho debris
Coho gravel
Coho off-channel

Adequate dissolved O2
Anadromous Water Quality
Coho summer H2O
Optimum Range of Temp
Passage
Percent pool
Pool Habitat

Qualitative assessment of physical factors that lead to effective migration such as open passages and adequate pool habitat. If true, then

Figure 6—The More button in the Knowledge base window displays a new pane with additional information about the object selected in the upper pane.

Save As: Displays a *File name dialog window* in which to enter a file name for the knowledge base. The default directory location is the \projectname\NWKB.

Revert to Saved: Discards any changes made to the knowledge base since it was last saved and restores the previous version of the knowledge base.

Print: Prints an ASCII script of the currently loaded knowledge base.

Exit: Exits NetWeaver. If the current knowledge base has been edited since the last save, NetWeaver prompts, asking if the knowledge base should be saved.

Edit menu—

New object: This *menu item* cascades to a *submenu* whose items are identified in the following bulleted list. When any of these items are chosen, NetWeaver first displays the *Documentation window* and then displays another window specific to the type of object being created.

- *New Note*: Displays the *Notes window* to create a new note.
- *New Group*: Displays the *Evaluation group window* to define the contents of a new evaluation group.

- *New Network*: Displays the *Network window* to design the hierarchy for a new dependency network.
- *New Calculated Data Link*: Displays the *Data link window* to create a new calculated data link, and then displays the *Calculated data link window*, which is essentially a network window because it is used to construct a graphic representation of the required calculations.
- *New Simple Data Link*: Opens a documentation window and then a data link window to create a new simple data link.

Object: This *menu item* displays the appropriate *edit window* for the type of object that has been selected in the *Knowledge base window* (see *New object*, above, for the names of edit windows that correspond to a particular NetWeaver object).

Documentation: Displays the *Documentation window* for the currently selected object in the knowledge base window.

Clear All: Resets all data links and networks to their unevaluated states.

Delete: Deletes the currently selected object in the knowledge base window from the knowledge base. NetWeaver will not delete a network object if it is referenced by any networks. To delete an object, all references to it must first be deleted.

Evaluate menu—Choose *Evaluate object* on the *Evaluate menu* to perform an evaluation of the object currently selected in the *Knowledge base window*. For groups, networks, and calculated data links that have an argument, NetWeaver displays the *Evaluation window* to run an evaluation on the object. For simple data links, NetWeaver displays the *Data link evaluation window*. For calculated data links whose values are used directly rather than evaluated against an argument, NetWeaver displays the *Calculated data link window*.

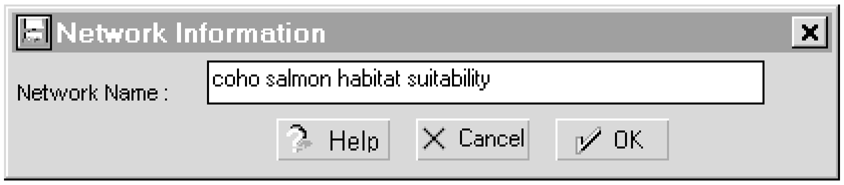
Window menu—

Arrange icons: Arranges any iconized NetWeaver windows neatly across the bottom of the NetWeaver application window.

Minimize All: Minimizes (iconizes) all NetWeaver windows that are currently open in the NetWeaver application window.

Options menu—

Cue cards: Toggles display of cue cards for NetWeaver windows on and off. A check mark next to the *menu item* indicates that cue card display is enabled.



coho salmon habitat suitability

Figure 7—The Information window is used to enter or edit a descriptive name for the knowledge base.

Autoload: Automatically loads the last knowledge base opened in the previous NetWeaver session.

Help menu—

Contents: Displays the table of contents for the help system.

Search: Displays a dialog window containing an index of help topics (see "Searching for Help Topics" for additional information).

How to Use NetWeaver: Covers many of the basic procedures used to construct knowledge bases in NetWeaver.

Using Help: Provides instructions for using features of the help system.

About: Displays information about the current version of NetWeaver and its developers.

Windows

NetWeaver displays many different windows to handle all the various tasks associated with designing a knowledge base. Most windows are specific to particular NetWeaver objects. Descriptions of these windows and discussions about their use are presented in later chapters. Three windows are relevant, however, to a discussion of the NetWeaver interface: neither the *Information window* nor the *Notes window* are associated with network objects. The *Documentation window* also is discussed here because it is associated with all network objects.

Information window—

Usage—This window (fig. 7) associates a descriptive name with a knowledge base. Naming a knowledge base is optional. The file name associated with a knowledge base is displayed in the *title bar* of the NetWeaver application window. New knowledge bases not yet saved are named "Untitled." Naming a knowledge base via the *Information window* allows specification of a longer, more descriptive name than is possible with a knowledge base filename limited to eight characters. If a name is

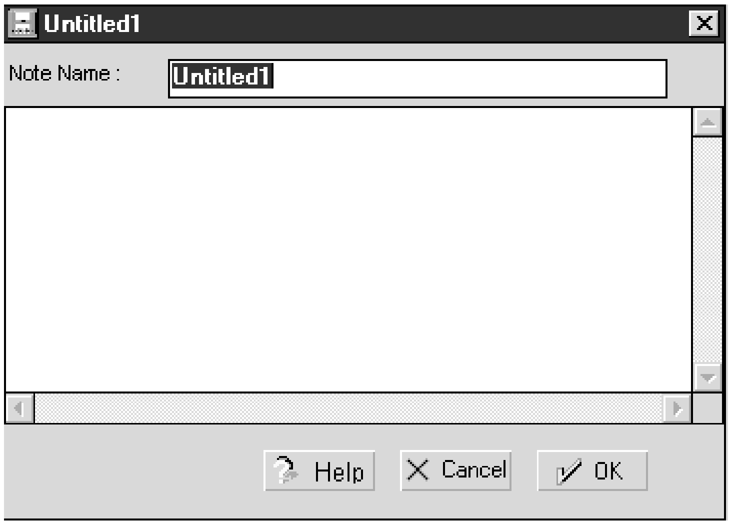


Figure 8—The Notes window is used to enter or edit notes about the knowledge base.

entered on the *edit line* of the *Information window*, that name is displayed in the *title bar* of the *Knowledge base window*.

Access—To open the *Information window*, press the *Information button* on the *Application button bar* or choose *Get info* on the *File menu* of the *Application menu bar*.

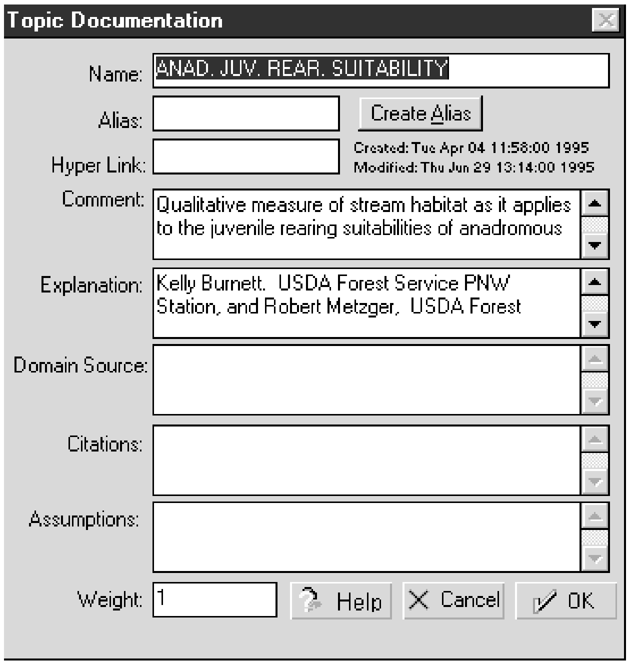
Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Notes window—

Usage—Use the *Notes window* (fig. 8) to add information not directly related to specific networks objects. While a knowledge base is under construction, for example, ideas may come up that you are not ready to deal with at that moment, but you would like to at least make a brief annotation so the thought is not lost. In a completed knowledge base, notes also can be used to provide some general documentation of the construction and use of the knowledge base.

Access—

To create a new note, press the *New Note button* on the *Application button bar*, or choose *New Note* on the *New object submenu* of the *Edit menu*.



Name: ANAD. JUV. REAR. SUITABILITY

Alias:

Hyper Link:

Created: Tue Apr 04 11:58:00 1995

Modified: Thu Jun 29 13:14:00 1995

Comment: Qualitative measure of stream habitat as it applies to the juvenile rearing suitabilities of anadromous

Explanation: Kelly Burnett. USDA Forest Service PNW Station, and Robert Metzger, USDA Forest

Domain Source:

Citations:

Assumptions:

Weight: 1

Figure 9—The Documentation window is used to enter or edit the name of a network or data link, and to provide optional documentation about the object.

To review existing notes:

1. Press the *Notes tab* in the *Knowledge base window*.
2. Click on the name of the desired note in the list displayed in the *Notes folder* to highlight the name.
3. Choose *Object* on the *Edit menu* of the *Application menu bar* or press the *Edit button* in the *Knowledge base window*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Documentation window—

Usage—Use this *dialog window* (fig. 9) to provide documentation for a new object when it is created or to edit the documentation. For detailed suggestions about documenting an object, see the later section, “Documenting a New Network Object.”

Access—The documentation window automatically is displayed when a new object is created. To edit documentation for an object that is displayed in the *Knowledge base window*:

1. Click on the object name in the appropriate folder to highlight the name.
2. Choose *Documentation* on the *Edit menu* of the *Application menu bar* or press the *Docs button* in the *Knowledge base window*.

Documentation of network objects also can be edited from any *Network window* that displays a reference to a network object. To edit object documentation in the network window:

1. Right click on the graphic representation of the referenced object in the *Network window*.
2. Choose *Documentation* from the *pop-up menu*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Working With NetWeaver Objects

Introduction to NetWeaver Objects

The following two reference chapters provide more detailed information about NetWeaver objects and methods for working with them in the NetWeaver interface. Here, though, is a brief summary of the collection of objects that make up a NetWeaver knowledge base. There are two basic categories: user-defined objects and predefined objects.

User-Defined Objects

A dependency network is an object that represents a proposition about some topic of interest in the problem domain for which the knowledge base is being constructed. The key attribute of a network is its truth value, which is a measure of the degree to which subordinate (antecedent) networks and data links support or refute the proposition of the dependent network. Each network specifies a dependency hierarchy in terms of other antecedent networks, data links, logical nodes, and other relational nodes.

An evaluation group is a user-defined collection of networks (it is not a NetWeaver object in the strict sense). A group specifies a collection of networks that are conceptually relevant to one another and that the knowledge base designer wants to see evaluated as a set.

A simple data link is responsible for reading a single datum and optionally evaluating it. Because a simple data link may optionally evaluate a datum, it conceptually is an elementary form of a network. Simple data links may either evaluate a datum against a simple or fuzzy argument or simply pass their data value to another object.

A calculated data link shares properties of both simple data links and networks. It reads and optionally evaluates data like a data link, and it specifies a mathematical expression in a functional dependency structure that reads data from one or more data links to perform some transformation on the data.

Predefined NetWeaver Objects

Logic nodes, such as AND and OR, are objects used to specify the logical dependency of a network on its antecedents. Because these objects are predefined, they do not have to be created by the user. Instead, they are simply inserted into the dependency hierarchy of a network.

Functional nodes, such as “*” (multiplication) and “/” (division), are objects used to construct mathematical expressions for use in calculated data links and comparison nodes.

NetWeaver also includes a predefined set of special-purpose objects referred to as other relational objects. Examples include switch, comparison, and fuzzy nodes. These predefined objects generally operate in conjunction with networks and data links. A switch node, for example, selects among alternative network pathways based on the data value passed from an associated data link.

Creating Objects

Evaluation groups can be created only from the *Application button bar*. Data links and networks may be created from either the *Application button bar* or the *Network window button bar* in an open *Network window*. There is a small but important difference between the two methods. In the first case, only the object is created. In the second case, a reference to the object also is created and attached to the node currently highlighted in the *Network window*.

Application Menu Bar

Press the appropriate *button* (group, network, simple data link, or calculated data link) on the *Application button bar* or choose *New object* on the *Edit menu* of the *Application menu bar*, then choose from the list of object types to create (evaluation group, network, simple data link, or calculated data link).

Network Window

Press the appropriate *object button* (network, simple data link, or calculated data link) on the *Network window button bar* and choose *New* on the *pop-up menu*. This creates an object as well as a reference to a network object at the same time.

What Happens Next

When a new NetWeaver object is created via either the *Application menu bar* or the *Network window button bar*, NetWeaver automatically displays the *Documentation window* (see "Documenting a New Network Object," below). After entering any pertinent documentation in the *Documentation window*, close the window. NetWeaver then automatically displays a window in which to provide specifications for the new object:

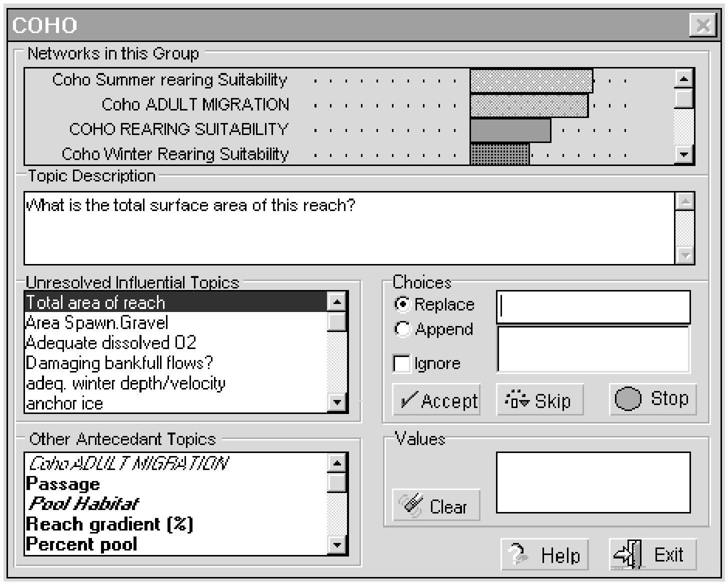
- The *Evaluation group window* (fig. 10) specifies the networks included in an evaluation group.
- The *Data link window* (fig. 11) specifies the arguments for a data link.
- The *Network window* (fig. 12) specifies the dependency hierarchy of a network.

Documenting a New Network Object

NetWeaver displays the *Documentation window* as soon as a *menu item* is chosen or a *button* is pressed to create a new object (fig. 13). Other than entering a name for the new object in the name field of the *Document window*, documentation is strictly optional but also extremely valuable. Documenting a network can be streamlined by putting detailed information at the highest applicable level of the network hierarchy and briefly referencing its name subsequently in the documentation for antecedents.

The *Documentation window* contains *edit boxes* for the following object attributes:

- Name
- Alias
- Hyperlink
- Comment
- Explanation
- Domain source
- Citations
- Assumptions
- Weight


COHO

Networks in this Group

Coho Summer rearing Suitability

Coho ADULT MIGRATION

COHO REARING SUITABILITY

Coho Winter Rearing Suitability

Topic Description

What is the total surface area of this reach?

Unresolved Influential Topics

Total area of reach
Area Spawn.Gravel
Adequate dissolved O2
Damaging bankfull flows?
adeq. winter depth/velocity
anchor ice

Choices

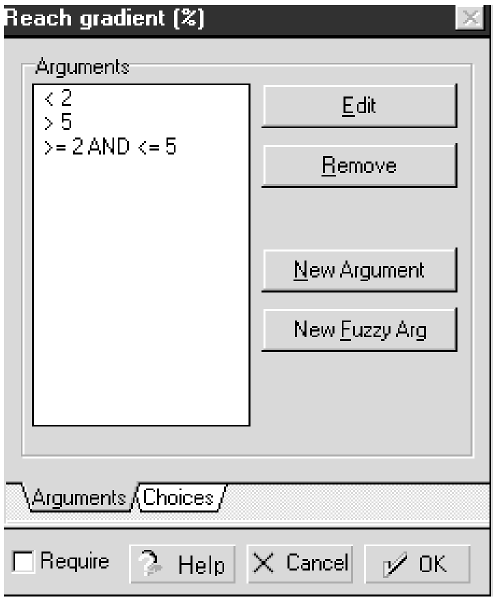
Replace
Append
Ignore

Other Antecedant Topics

Coho ADULT MIGRATION
Passage
Pool Habitat
Reach gradient (%)
Percent pool

Values

Figure 10—The Evaluation window is used to view the evaluation of an evaluation group (a collection of networks) or individual networks.


Reach gradient (%)

Arguments

< 2
> 5
>= 2 AND <= 5

Require

Figure 11—The Data link window is used to initially enter, or subsequently edit, the arguments of a data link. The Simple argument and Fuzzy argument windows are opened from this window.

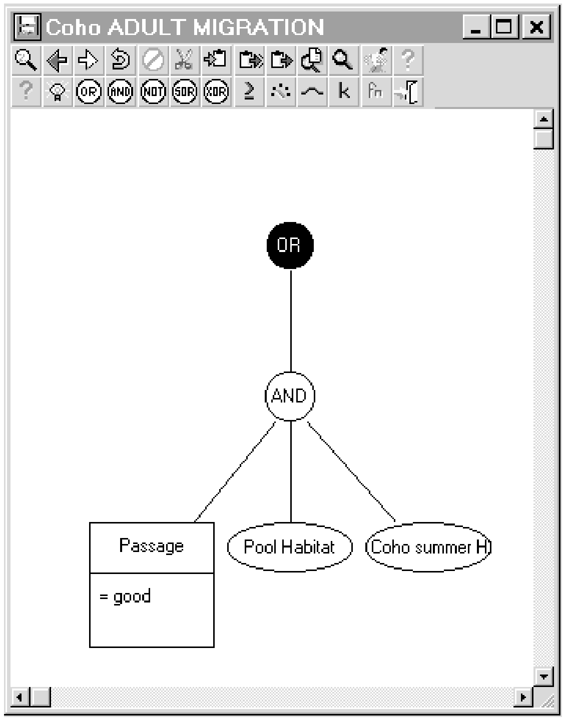
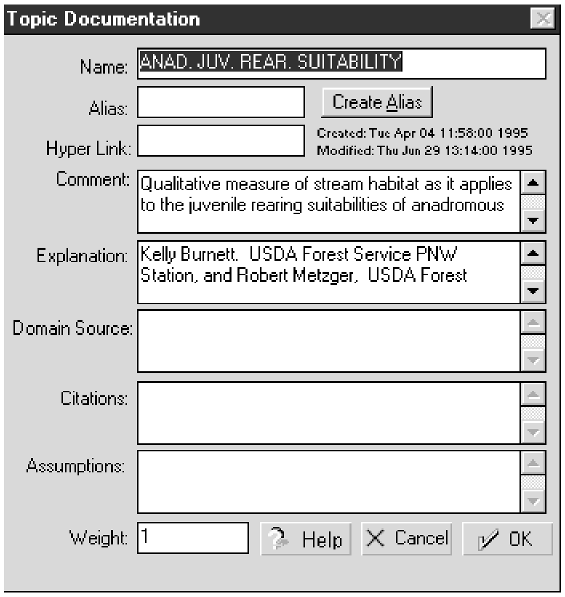


Figure 12—The Network window is used to initially specify, or subsequently edit, the dependency hierarchy of a network object.


Topic Documentation

Name:	ANAD. JUV. REAR. SUITABILITY
Alias:	
Hyper Link:	
Comment:	Qualitative measure of stream habitat as it applies to the juvenile rearing suitabilities of anadromous
Explanation:	Kelly Burnett. USDA Forest Service PNW Station, and Robert Metzger, USDA Forest
Domain Source:	
Citations:	
Assumptions:	
Weight:	1

Created: Tue Apr 04 11:58:00 1995
Modified: Thu Jun 29 13:14:00 1995

Figure 13—The Documentation window is used to enter or edit the name of a network or data link, and to provide optional documentation about the object.

Documentation Attributes of an Object

Name—When naming a dependency network or data link, it is important to keep in mind that its basic function is to test a proposition about the topic of interest (data links that use data directly are an exception). Long names may be rendered in such a small font that the name is not readable on a graphic object in the network window. However, when the mouse pointer is positioned over a graphic object in the network window, the object's name is displayed on the *status bar* at the bottom of the main application window.

Alias—A dBase-compliant column name used to associate the object with a field (column) in a database table. dBase compliant means:

1. A maximum of 10 characters.
2. No spaces.
3. All uppercase.
4. First character is alphabetic.
5. Alphanumeric or underscore for other characters.

Hyperlink—Hyperlink specifies a path and file name with additional or illustrative information about the object. If no path name is given, the file is assumed to reside in the NWKB subdirectory.

File types currently supported are .TXT, .BMP, and .HTM.

If the file name is unique within the directory specified by the path, then the file name extension can be omitted.

To specify multiple file names, delimit the names with either a comma or a pipe (|) (for example, mybmp|yourbmp|ourbmp).

Comment—The comment field provides a brief description of an object. Comments are displayed in the central *Information pane* of the *Evaluation window* when a data link or dependency network is selected from the list of available objects. Comments are particularly useful as prompts for data input to data links during network evaluation and should indicate the units (for example, meters, feet, etc.) expected by the data link. The comment field also is the best place to explicitly define the proposition associated with a network.

Explanation—The explanation field provides space for more extensive documentation about an object. Here, one might explain the rationale for representing an object in the network and perhaps something about the structure of underlying dependencies.

Domain source—This field documents the individuals who developed this portion of network structure. As an example of streamlining documentation, if three people have designed a large network hierarchy as a team, this is an item that you might enter in its entirety only once for the top-level network. Domain source documentation for some or all antecedents then might contain only a brief reference to the top-level network's domain source field.

Citations—Provide any literature references that might be relevant to how the network is structured with respect to this object and its antecedents.

Assumptions—Virtually all models contain underlying assumptions. Unfortunately, many modeling applications provide little or no on-line documentation about those assumptions. If the model developer is also the sole user of the system, there is no problem. But, when the user of a knowledge base is not the developer, there is potential for serious misapplication if the model assumptions are not clearly understood.

Weight—All weights have a default value of 1.0. The practical implication of equal weights on all antecedents of a given dependency network is that such antecedents are presumed to contribute equally to the truth value of the assertion. Unless there is a compelling reason to depart from this assumption, use of the default weight is recommended. The one exception to this advice is in the context of antecedents to a fuzzy SOR node (see the section, "Fuzzy SOR Node," in "Relations Among Objects," below).

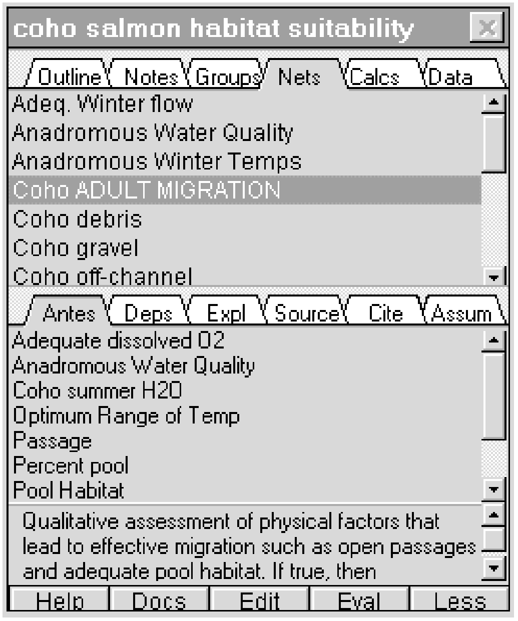
Editing Object Documentation

Knowledge Base Window

1. Click on the name of an object in the appropriate *folder* of the *Knowledge base window* to highlight the item.
2. Choose *Documentation* from *Edit menu* on the *Application menu bar* or press the *Docs button* on the *Knowledge base window button bar*.

Network Window

1. Right click on the object's graphic representation in the *Network window*.
2. Choose *Documentation* from the *pop-up menu*.


coho salmon habitat suitability

Adeq. Winter flow
Anadromous Water Quality
Anadromous Winter Temps
Coho ADULT MIGRATION
Coho debris
Coho gravel
Coho off-channel

Adequate dissolved O2
Anadromous Water Quality
Coho summer H2O
Optimum Range of Temp
Passage
Percent pool
Pool Habitat

Qualitative assessment of physical factors that lead to effective migration such as open passages and adequate pool habitat. If true, then

Figure 14—The More button in the Knowledge base window displays a new pane with additional information about the object selected in the upper pane.

Viewing Documentation and Other Attributes of an Object

To view documentation associated with a particular object:

1. Click on the name of an object in the appropriate *folder* of the *Knowledge base window* to highlight the item.
2. Press the *More button* on the *Knowledge base window button bar*.

More opens another set of *folders* in an additional lower *pane* of the *Knowledge base window*. *Folders* in the lower *pane* display documentation attributes that have been provided about an object via the *Documentation window* for an item that is currently highlighted in the upper *pane* of the *Knowledge base window* (fig. 14). Documentation presented in these additional folders can be viewed but not edited. In addition to documentation attributes, *More* displays two additional *folders* that list the antecedents and dependents of the item currently selected in the upper *pane*.

Deleting a Network Object

The distinction between objects and references to objects is important in the context of deletion. A user-defined NetWeaver object can be deleted from the knowledge base only in the *Knowledge base window*. Deleting the graphic representation of an object from a *Network window* (see “Deleting References,” below) only deletes the reference to the object from the network in which it was displayed. The object itself remains in the knowledge base until it is deleted from the *Knowledge base window*.

Table 1—User-defined objects, their editable attributes, and applicable windows

Object	Editable attributes	Applicable window
Evaluation group	Networks in group	Evaluation group window
Dependency network	Dependency hierarchy	Network window
Data link	Available arguments	Data link window
	Simple argument definition	Simple argument window
	Fuzzy argument definition	Fuzzy argument window
Calculated data link	Functional hierarchy	Calculated data link window

Furthermore, to avoid creating havoc in a knowledge base, NetWeaver does not allow an object to be deleted if it is referenced in one or more dependency networks. To remove an object, all references to it first must be deleted from all networks in which the object is referenced. Assuming that there are no references to an object:

1. Click on the object in the appropriate *folder* of the *Knowledge base window* to highlight the item.
2. Choose *Delete* on the *Edit menu* of the *Application menu bar* to delete the object.

Editing a Network Object

All NetWeaver objects have documentation attributes that provide useful, and often necessary, information about the object (see “Editing Object Documentation,” above). All user-defined NetWeaver objects also have other attributes that can be edited. Table 1 summarizes associations between user-defined objects, relevant attributes that can be edited, and the applicable NetWeaver window.

The *Data link window* is used to create and edit arguments for both simple and calculated data links. Calculated data links additionally have their *Network window*, which is referred to as the *Calculated data link window* to distinguish it from the standard *Network window*. Both the *Simple argument window* and the *Fuzzy argument window* are accessed from the *Data link window* to create or edit simple and fuzzy arguments, respectively.

Evaluating a Network Object

Knowledge-Base Window

To open an *Evaluation window* for an evaluation group, an individual dependency network, or a data link:

1. Click on the name of the object in the appropriate *folder* of the *Knowledge base window* to highlight the item.
2. Choose *Evaluate object* from *Evaluate menu* on the *Application menu bar* or press the *Eval button* on the *Knowledge base window button bar*.

As a short-cut to opening the *Evaluation window* for evaluation groups in particular, double click on the evaluation group name in either the *Outline folder* or the *Group folder* of the *Knowledge base window*.

Network Window

To open an *Evaluation window* for an individual dependency network or a data link displayed in a *Network window*:

1. Right click on the referenced object's graphic representation in the *Network window* to highlight the item.
2. Choose *Evaluate* from the *pop-up menu*.

Navigating a Knowledge Base

There are several ways to navigate the structure of a knowledge base:

- The *Outline folder* of the *Knowledge base window* can be a useful starting point for navigation because it displays a simplified view of the complete knowledge base structure. Browse through the hierarchy in this *folder* and then either double click on a network to open its *Network window* or click on the network name to highlight it in the *folder*; then *Edit button* on the *Knowledge base window button bar*.
- In the *Nets folder* of the *Knowledge base window*, either double click on the network name or click on the name of a network to highlight it, then press the *Edit button* on the *Knowledge base window button bar*.
- During an evaluation, networks displayed in any of the *panes* of the *Evaluation window* can be opened by double clicking on the name of the network.

- Once any *Network window* is open, it is possible to navigate to other dependency networks displayed within the currently active *Network window* by double clicking on the graphic representation of the network in the network window, or by right clicking on the graphic and choosing *Edit* from the *pop-up menu*.

Referencing Network Objects

Data links and dependency networks are reusable NetWeaver objects. Although an object may appear in multiple networks, there is only one copy of the actual object. Graphic representations of an object in network windows are not copies of the object, but references to the original object.

Adding References to a Network

When an object is first created via one of the *new object buttons* on the *Application button bar*, there initially are no references to that object in the knowledge base. The reference is created when the object is added to a specific network in a *Network window*. To add a reference to a data link or dependency network in the current network window:

1. Click on the network node to which the reference is to be attached to highlight the node.

2. Press the appropriate *object button* on the *Network window button bar*.

3. Choose the object name from the *pop-up menu* of existing objects of that type.

When an object is created within a *Network window*, NetWeaver invokes its standard methods for creating a new object (see “Creating Objects,” earlier in this chapter) and then inserts a reference to the object at the currently highlighted node in the *Network window*. If a network has been created, its *Network window* can be opened to edit the network’s dependency hierarchy by double clicking the graphic representation of its newly added reference. If a data link has been created, NetWeaver automatically opens the *Data link reference window* for the new data link so that its arguments can be defined.

Deleting References

NetWeaver does not allow an object to be deleted from the knowledge base if the object is referenced (that is, in use) in one or more dependency networks. To remove an object from the knowledge base, all references to it first must be deleted from all networks in which it occurs.

To delete a referenced NetWeaver object from a network window:

1. Click on the graphic representation of the referenced object in the *Network window* to highlight the item.

2. Press the *Delete button* on the *Network window button bar*.

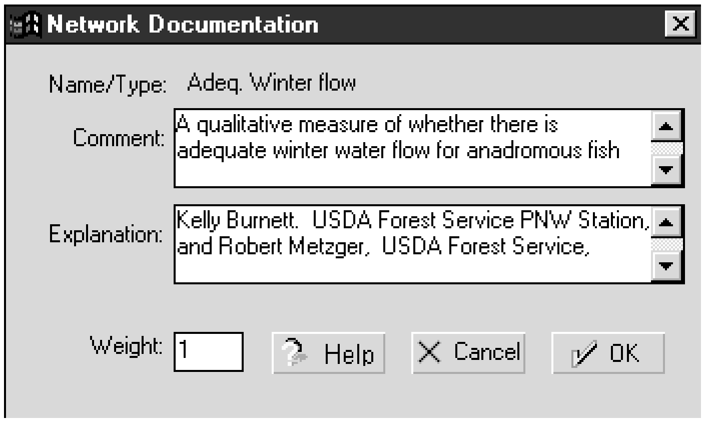
Editing a Reference

References to dependency networks and data links that are displayed within a *Network window* can be edited. Each type of reference has its own *reference window*.

Dependency network references—Each reference to a dependency network has its own *Comment*, *Explanation*, and *Weight attributes* that can be edited in the *Network reference window* (fig. 15). Attributes displayed in this window are specific to that reference and are distinct from the attributes of the object in the *Documentation window*. The specification of a network's dependency hierarchy cannot be edited in a *Network reference window* because the specification of a network object has global scope within a knowledge base. In my experience, this window rarely is used, but there can be situations in which a special comment about the use of network in a specific part of a knowledge base is warranted.

Data link references—Each reference to a data link has its own *Comment*, *Explanation*, and *Weight attributes* that can be edited in the *Data link reference window* (fig. 16). These attributes are specific to that reference and are distinct from attributes of the same name that may have been entered for the data link object itself in the *Documentation window*. The rationale for having attributes specific to a data link reference is the same as that for network references (see previous paragraph).

In addition, the *Data link reference window* includes an interface and methods for assigning a specific argument or set of arguments to be used by the data link reference. Similarly, a data link object has global scope in the knowledge base, but specific references to a data link may use different arguments. It is, therefore, quite possible for a data link reference to evaluate to true in one instance and false in another because data link references can take different arguments in different locations in a knowledge base. A related point is that, although there may be multiple references to a data link object, it is the data link object itself that is responsible for requesting a data value, so data input for a data link is read only once, but it is evaluated once for each instance in which the data link is referenced.


Network Documentation

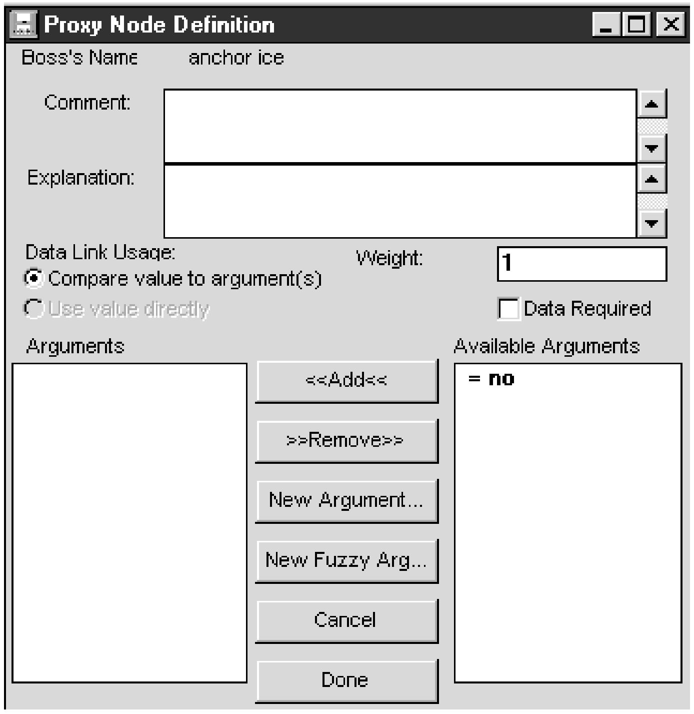
Name/Type: Adeq. Winter flow

Comment: A qualitative measure of whether there is adequate winter water flow for anadromous fish

Explanation: Kelly Burnett. USDA Forest Service PNW Station, and Robert Metzger, USDA Forest Service,

Weight: 1

Figure 15—Network reference window.


Proxy Node Definition

Boss's Name anchor ice

Comment:

Explanation:

Data Link Usage:
[✓ Compare value to argument(s)]
Use value directly

Weight: 1

Data Required

Arguments

Available Arguments
= no

Figure 16—Data link reference window.

NetWeaver Objects

Dependency Networks

A dependency network is a formal logical representation of how system states at one level of a conceptual model are affected by, or dependent on, other antecedent states. Networks represent things such as system states and processes. A single dependency network is hierarchical in structure (fig. 17), but NetWeaver knowledge bases generally are networks of such hierarchies (fig. 18) in which a network is composed of other networks whose logical relation to the parent network is defined by relational nodes (fig. 19). Data links are special cases of networks whose state depends only on data. Thus, a data link (see “Data Links,” below) can be thought of as an elementary dependency network.

The dependency hierarchy under any given network is built graphically in the *Network window* (fig. 19). The *Network window* automatically opens after the *Document window* is closed when a network is created.

Truth Value

The concept of a truth value comes from the discipline of cognitive science, which is basically the science of how we know what we know. In cognitive science, the concepts of proposition and truth value are directly related. A proposition is considered to be the smallest unit of thought that can be assigned a measure of truth (Stillings and others 1991). NetWeaver knowledge bases are based on propositional logic and every NetWeaver network asserts some proposition concerning the topic it is constructed to evaluate. The key attribute, or state variable, of a network is its truth value, which expresses the degree to which antecedent information supports or contradicts the proposition that the network is designed to test.

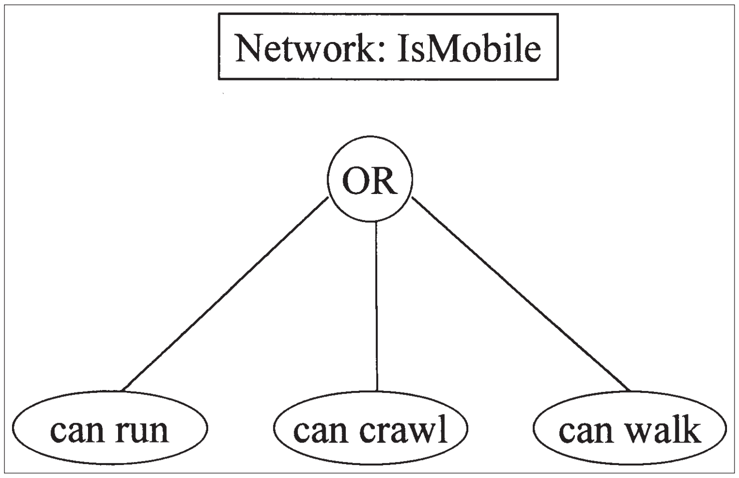


Figure 17—A single dependency network has a simple dependency hierarchy.

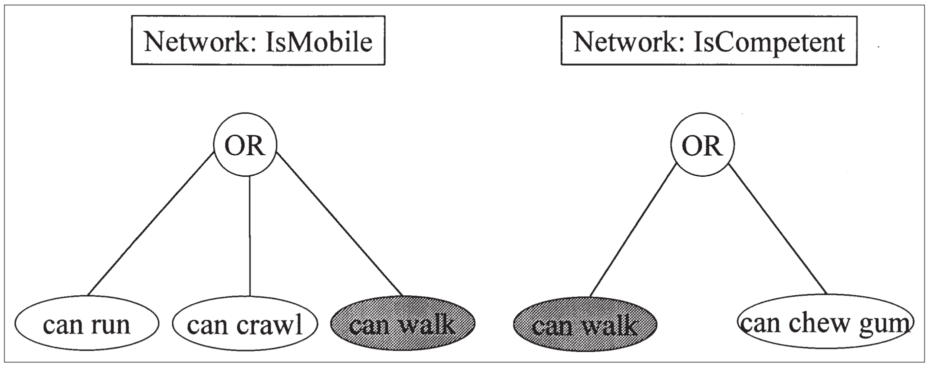


Figure 18—A knowledge base containing two or more dependency networks linked by common networks no longer has a simple hierarchical structure.

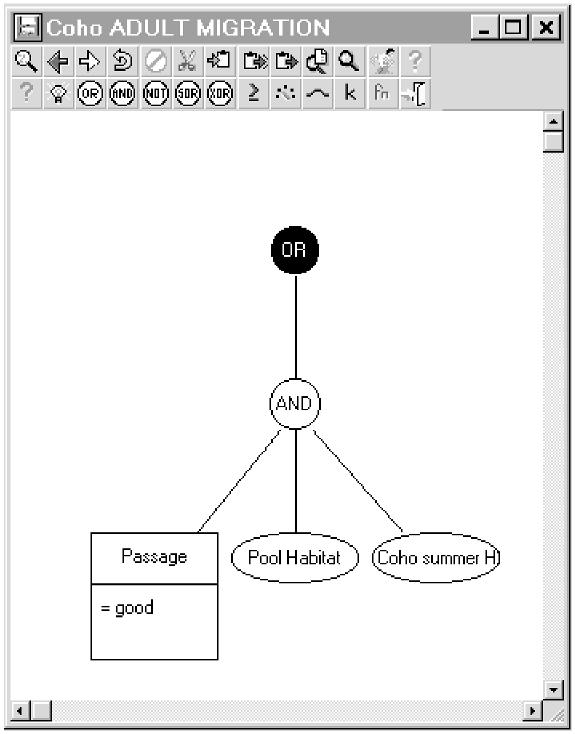


Figure 19—The Network window is used to initially specify, or subsequently edit, the dependency hierarchy of a network object.

If all evidence antecedent to a proposition supports the assertion, then the truth value for the network is 1 (completely true). If all evidence antecedent to an assertion is contrary to that assertion, then the truth value for the network is -1 (completely false). If there is no evidence for or against the assertion, then the truth value is 0 (undetermined). Truth values also may be partially true or partially false. Three conditions cause this condition in NetWeaver:

1. Some data needed to fully evaluate the node or dependency network have not yet been supplied when evaluation of a network is being performed.
2. Data are missing and cannot be supplied.
3. One or more data items that influence the truth value of a dependency network have been evaluated against a fuzzy argument and found not to have full membership in the fuzzy set defined by the fuzzy argument.

Dependency

As a noun, the term antecedent literally means “something that comes before something else.” An antecedent network, for example, is one that another network depends on for its truth value. The term antecedent is equally applicable to dependency networks, relational nodes, and data links. An antecedent network, node, or data link logically precedes its dependent network or node in the sense that the value of the antecedent must be known to evaluate a network or node that depends on it.

Although the networks of hierarchies in a NetWeaver knowledge base are more general relational structures than simple hierarchies, there are some restrictions on the structure of knowledge bases because they are based on propositional logic. In particular, the NetWeaver inference engine monitors the construction of knowledge bases and will not allow a dependency structure that implements circular reasoning. For example, it is not permissible for network A to depend on network B if network B depends on network A because this clearly leads to circular reasoning.

Behavior of Dependency Networks

Dependency networks have three basic behaviors:

1. They query antecedent networks on which they depend to determine the state of the latter.

2. They evaluate their own state, given the state of their antecedent networks.

3. They inform higher level networks that depend on them about their state.

Network Window

Usage—Use this window (fig. 19) to define, edit, or simply view the logical dependencies of a specific dependency network or calculated data link. The structure of network dependencies defined in this window is the network's object specification. It is not necessary to define the logical dependencies of a network at the time it is created. To defer defining the logical structure, just close the *Network window*.

One of NetWeaver's powerful features is that many windows are dynamically linked to one another. For example, when an evaluation group or network is evaluated in the *Evaluation window*, the states of network objects displayed in any open *Network windows* are dynamically updated, concurrent with the evaluation. Object references are color coded in bright red if the truth value of the referenced object is completely false, bright green if the truth value is completely true, and graduated shades for intermediate truth values. The actual truth value for the object also is displayed to the right of the *Network window* button bar when the mouse pointer is moved over the reference.

Access—The Network window automatically opens after the *Documentation window* is closed when a new network is created. There are several ways to open the *Network window* for an existing network. From the *Knowledge base window*:

1. In either the *Outline folder* or the *Nets folder*, click on the network name to highlight it.

2. Either press the *Edit button* on the *Knowledge base window button bar* or choose *Edit object* from the *Edit menu*.

In either the *Knowledge base window*, the *Evaluation window*, or even another *Network window* that displays a reference to a network, double click on the name of the network object or the graphic representation of its reference to open its *Network window*.

Finally, in a *Network window* that displays a reference to a network, right click on the graphic representation of the reference and choose *Edit* from the *pop-up menu*.

Defining the logical dependencies of a new network—When a network is first created and the *Network window* is opened, the window initially only contains an OR node that represents the root of the network. The *Network window button bar* (see below) provides access to existing networks and data links that may be referenced as antecedents of the current network. It is also possible to insert a new network in the *Network window* if a desired antecedent has not been previously created.

Inserting nodes and dependency networks—If the subject network's dependency on its immediate antecedents involves an OR relation, then new or existing networks may be attached directly to the root OR node of the current network. If the subject network's dependency on its immediate antecedents involves an AND relation or some other relation (see "Fuzzy Logic Nodes," below), then:

1. Press the *button* on the *Network window button bar* for the type of logical node (AND, XOR, SOR, etc.) to attach the node to the OR node.

2. Click on the newly inserted logical node so that it is now highlighted.

3. Press the *Network button* on the *Network window button bar*, and choose *New* or the name of an existing network from the *pop-up menu* to attach a reference to the new or existing network to the node highlighted at step 2.

If a previously existing network is inserted into a dependency network in the open *Network window* at step 3, then antecedents cannot be attached to the inserted network, because the specification of a network has global scope within a knowledge base. Thus, when an existing network is inserted into the structure of another network's hierarchy, you are actually inserting a reference to an existing object whose dependency hierarchy is defined in its own *Network window*.

If, at step 3, a new network is created for insertion into the current network hierarchy, then NetWeaver displays the *Documentation window* for the new network. After the *Documentation window* is closed, a reference to the new network is inserted into the hierarchy. NetWeaver does not automatically open a new *Network window* for the newly created network, but the graphic representation of its reference can be double clicked to open a new *Network window* and specify the dependency hierarchy of this new network if desired.

Inserting simple and calculated data links—The steps for inserting data links into a network are essentially the same as those described in the previous section for inserting a network. If an existing data link is inserted, then NetWeaver displays the *Data link reference window* to define arguments to be used for this specific reference to the data link. If a new data link is inserted, then NetWeaver first displays the *Documentation window*, and then, on closing the latter, the *Data link reference window*. For details about use of the *Data link reference window*, see the later section, "Data Links."

Calculated data links are similar to networks in that a *Network window* is used to graphically construct its functional structure. Later, this particular *Network window* is referred to as the *Calculated data link window*, because, although it is essentially the same as the standard *Network window*, special *buttons* on the *Network window button bar* for constructing functions are enabled only when the object is a calculated data link. NetWeaver does not automatically open the *Calculated data link window* when a new calculated data link is inserted in a network. To open the window for the calculated data link, just double click on the graphic representation of its reference.

***Network window button bar*—**Various editing operations can be performed on items that have been highlighted in a network hierarchy. In general, click on the graphic representation of the reference to which some action is to be applied, and press one of the following buttons:

Shift left: Shifts the selected object to the left.

Shift right: Shifts the selected object to the right.

Undo the last edit action.

Delete the object from the network, but does not put it on the Windows Clipboard (compare to *Cut*, next).

Cut removes the object from the network and puts it on the Windows Clipboard for subsequent *Paste* action (compare to *Delete*, previous).

Copy object and (*children*) *antecedents to clipboard*: Copies the selected object and any of its children to the Windows Clipboard.

Paste object and children from clipboard: Pastes the selected object and any of its children from the Windows Clipboard to the currently selected network object (compare with *Paste children only*, next).

Paste children only is similar to *Paste object and children from clipboard*, except that only the antecedents (and not their parent) are pasted to the selected object.

Documentation opens the *Document window*.

Local edit opens the *Reference window*.

Evaluate runs an evaluation on the object.

Data link creates and attaches a new data link or attaches an existing data link to the currently selected network object. Valid objects are fuzzy logic nodes, comparison nodes, switches, fuzzy curve nodes, and mathematical operators.

Network creates and attaches a new network or attaches an existing network to the currently selected network object. Valid objects are fuzzy logic nodes, comparison nodes, switches, fuzzy curve nodes, and mathematical operators.

OR attaches a fuzzy OR node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and fuzzy curve nodes.

AND attaches a fuzzy AND node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and fuzzy curve nodes.

NOT attaches a fuzzy NOT node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and fuzzy curve nodes.

SOR attaches a fuzzy SOR node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and fuzzy curve nodes.

XOR attaches a fuzzy XOR node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and fuzzy curve nodes.

Comparison node attaches a comparison node to the currently selected network object. Valid objects are fuzzy logic nodes, other comparison nodes, switches, and fuzzy curve nodes.

Switch node attaches a switch to the currently selected network object. Valid objects are fuzzy logic nodes, comparison nodes, other switches, and fuzzy curve nodes.

Fuzzy curve node attaches a fuzzy curve node to the currently selected network object. Valid objects are other fuzzy logic nodes, comparison nodes, switches, and other fuzzy curve nodes.

Constant attaches a constant node to the currently selected network object. Valid objects are function nodes, comparison nodes, switches, and fuzzy curve nodes.

Function attaches a function node to the currently selected network object. Valid objects are calculated data links, and other function nodes .

Four additional buttons on the *Network window button bar* operate independently of items highlighted in a *Network window*.

Zoom enlarges or shrinks the contents of the window display area.

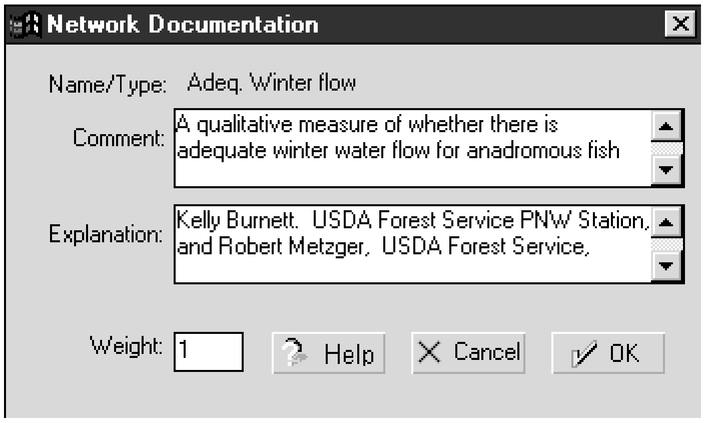
Cue card displays the cue card for the *Network window*.

Help opens NetWeaver help documentation for the *Network window*.

Exit closes the *Network window*.

Viewing logical dependencies of a network—Networks may extend many levels deep; however, the network window displays only the immediate antecedents of the subject network being displayed. A simple view of the entire network structure of a knowledge base is available in the *Outline folder* of the *Knowledge base window*. This view shows the hierarchy of dependencies among networks without the detailed information about logical connections.

From a given network window, it is also possible to navigate a network hierarchy downward by double clicking on antecedent networks referenced in a succession of network windows.


Network Documentation

Name/Type: Adeq. Winter flow

Comment: A qualitative measure of whether there is adequate winter water flow for anadromous fish

Explanation: Kelly Burnett. USDA Forest Service PNW Station, and Robert Metzger, USDA Forest Service,

Weight: 1

Figure 20—Network reference window.

Network Reference Window

Usage—You may never need to use the network reference window (fig. 20), but it is provided primarily to enter a comment or explanation specific to the referenced network concerning its use as an antecedent in the current network window. Often, no other commentary, besides that provided for the object itself in the *Documentation window*, is really needed. On occasion, however, there may be some need to provide additional comment and explanation concerning the use of a referenced network in a specific place in a knowledge base.

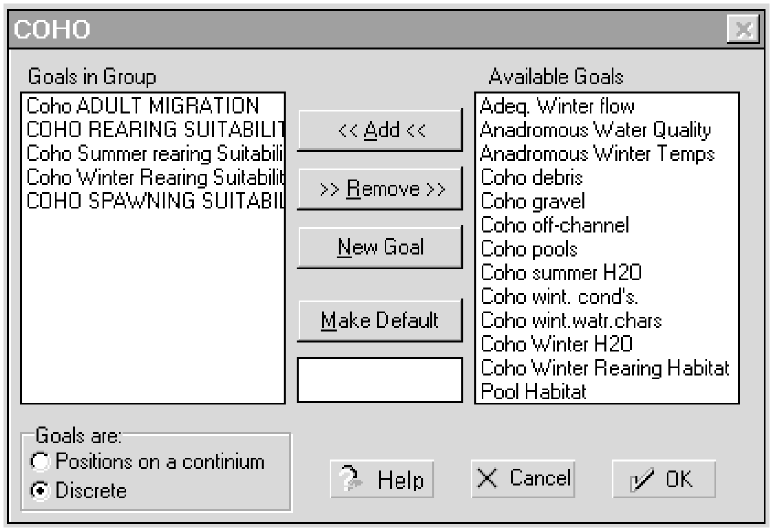
The *Network reference window* also can be used to assign a new weight to the referenced network specifically for use in the current network. I generally recommend that users use the default weight of 1.0, which is assigned in the *Documentation window*. One instance in which weighting is recommended is in conjunction with networks attached to a SOR node (see “Fuzzy SOR Node,” in “Relations Among Objects,” below).

Access—To open the *Network reference window*, right click on the graphic representation of the referenced network to highlight it; then choose *Local edit* from the *pop-up menu*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Evaluation Groups

Multiple networks may be defined as an evaluation group to simultaneously view the evaluation of several networks side by side. A NetWeaver knowledge base might contain only a few dependency networks, but more typically a knowledge base will contain many dependency networks representing a variety of components of the


COHO

Goals in Group
- Coho ADULT MIGRATION
- COHO REARING SUITABILIT
- Coho Summer rearing Suitabili
- Coho Winter Rearing Suitabilit
- COHO SPAWNING SUITABIL

Available Goals
- Adeq. Winter flow
- Anadromous Water Quality
- Anadromous Winter Temps
- Coho debris
- Coho gravel
- Coho off-channel
- Coho pools
- Coho summer H2O
- Coho wint. cond's.
- Coho wint.watr.chars
- Coho Winter H2O
- Coho Winter Rearing Habitat
- Pool Habitat

Goals are:
( ) Positions on a continium
(●) Discrete

Figure 21—Use the Evaluation group window to specify the networks to include in an evaluation group.

overall problem being modeled. The set of dependency networks comprising a single knowledge base is conceptually related, at least to the extent that it represents different aspects of the same problem. Some of these dependency networks may represent useful subsets whose simultaneous evaluation is particularly useful because of the manner in which they are conceptually, if not logically, related.

Sets of dependency networks comprising an evaluation group may either be discrete or represent segments on a continuum. An example of a group representing points on a continuum would be the set of networks representing the outcomes “viability low,” “viability medium,” and “viability high.” These outcomes are obviously highly interdependent. An example of a group consisting of discrete networks would be one containing “adequate habitat suitability” and “adequate population.” Although the two outcomes may be conceptually related, they do not represent alternative outcomes as in the previous example.

The collection of networks comprising an evaluation group is specified in the *Evaluation group window*. The conduct of an evaluation is observed and controlled via the *Evaluation window*. The *Evaluation group window* automatically opens after the *Document window* is closed when a group is created. See the description of these windows in the earlier section, “NetWeaver Interface” in “Getting Started.”

Evaluation Group Window

Usage—Use this *dialog window* (fig. 21) to assign a set of networks to an evaluation group when the group is initially created or to edit the list of networks in the group.

Access—When a new evaluation group is created, the group window is automatically displayed after the documentation window is closed. To edit an existing group, click on its name either in the *Outline folder* or in the *Group folder* of the Knowledge *base window* to highlight it, then choose *Edit object* on the *Edit menu* of the *Application menu bar*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* without assigning any additional dependency networks to the evaluation group.

Adding existing dependency networks to an evaluation group—If the knowledge base already contains dependency networks when the evaluation group is created, one or more networks can be selected from the *Available networks list box* and then added by pressing the *Add button*. The selected items are removed from the *Available networks list box* and added to the *Networks in group list box*.

Adding new dependency networks to an evaluation group—If the knowledge base does not contain any dependency networks when the evaluation group is created, then there will be no networks to select in the *Available networks list box*. The *New button* can be pressed, however, to create new networks while in the group window, which is equivalent to pressing the *New button* on the *Application button bar*.

Removing dependency networks from an evaluation group—To remove one or more dependency networks from an evaluation group, select the items in the *Networks in group list box* and press the *Remove button*. The items are removed from the *Networks in group list box* and added to the *Available networks list box*.

Evaluation Window

Usage—The *Evaluation window* (fig. 22) is used to evaluate a group or an individual network.

Access—

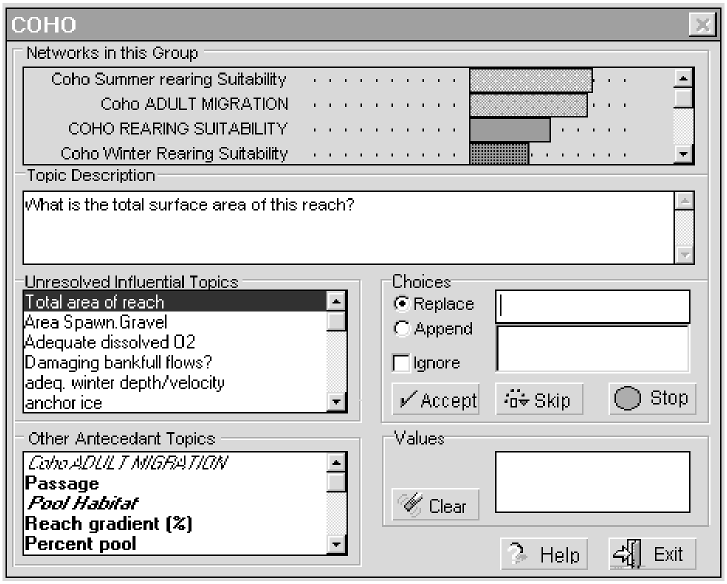
1. Click on the evaluation group or dependency network to be evaluated to highlight it in the *Knowledge base window*.

2. Choose *Evaluate object* on the *Evaluate menu* of the *Application menu bar* or press the *Eval button* on the *Knowledge base window menu bar*.

In the case of evaluation groups, the *Evaluation window* also can be opened by double clicking on the group name in the *Knowledge base window*.

To close the window, press the *Exit button* in the lower right corner of the window.

Evaluation pane—The upper pane of the *Evaluation window* displays the network


COHO

Networks in this Group

Coho Summer rearing Suitability

Coho ADULT MIGRATION

COHO REARING SUITABILITY

Coho Winter Rearing Suitability

Topic Description

What is the total surface area of this reach?

Unresolved Influential Topics

Total area of reach
Area Spawn.Gravel
Adequate dissolved O2
Damaging bankfull flows?
adeq. winter depth/velocity
anchor ice

Choices

Replace [✓]

Append

Ignore

Other Antecedant Topics

Coho ADULT MIGRATION
Passage
Pool Habitat
Reach gradient (%)
Percent pool

Values

Figure 22—The Evaluation window is used to view the evaluation of an evaluation group (a collection of networks) or individual networks.

or networks being evaluated and the result of their evaluation (fig. 22). A bar graph for each network displays its truth value as antecedent networks and data links are evaluated. Bars extending to the right of the bar graph midpoint become progressively brighter green, indicating increasingly positive truth values. Bars extending to the left of the bar graph midpoint become progressively brighter red, indicating increasingly negative truth values.

The *Network window* for any network displayed in the *Evaluation pane* can be opened by double clicking on the network name in the *Evaluation pane*. As discussed in the earlier section, "Network Window," the content of this window is dynamically updated during an evaluation. By opening *Network windows* during an evaluation, it is possible to observe the details of the changing evaluated states under any network.

Information pane—The *Information pane* is the central pane of the *Evaluation window* (fig. 22). It displays any comments associated with the currently selected network object in the *Influence pane* (see next section). The comment displayed in this pane is the comment attribute that was entered for the object in the *Documentation window*. Comments are particularly useful as a guide to the user for the type and units of data input for data links.

***Influence pane*—**The bottom pane of the *Evaluation window* is the *Influence pane* (fig. 22). This pane contains three *list boxes*.

Influence list box—The leftmost *list box* of the *Influence pane* displays unevaluated data links or networks that still have influence on networks being evaluated. In manual evaluation mode (see “Manual evaluation versus step mode,” below), the *Influence list box* is labeled “Influential antecedents,” and contains a list of all remaining data links and networks that still influence the networks displayed in the *Evaluation pane*. In step mode, the *Influence list box* is labeled “Influential data links,” because only unevaluated data links are displayed in the *list box* in step mode.

Other antecedents list box—The central *list box* of the *Influence pane* is labeled “Other antecedents.” In manual evaluation mode, the *Other antecedents list box* is usually empty at first. Data links and networks are added to this *list box* as they are evaluated. In step mode, the *Other antecedents list box* initially contains all unevaluated networks. If, as data are evaluated, any network or data link evaluates to fully true or fully false, that network or data link is highlighted in bold in the *list box*.

Choices combination list box—The rightmost *list box* of the *Influence pane* is the *Choices combination list box*, which is composed of an *edit box* and a *list box*. There also are *buttons* (see below, “Choices buttons”) associated with the *combination list box* that operate in conjunction with selections made in the combination list box.

If a data link is highlighted in the *Influence list box* or the *Other antecedents list box*, and choices for data input were defined for the data link, then the list of choices is displayed in the *Choices combination list box*. Either enter a value on the *edit line* or click to highlight a choice in the *list box*, if any are available.

If a network is highlighted in the *Influence list box*, or the *Other antecedents list box*, three predefined choices (True, False, and Undetermined) are displayed in the *list box*. The *edit line* is hidden when the selected antecedent is a network. Select one of the three choices from the *list box*.

***Choices buttons*—**The *Choices buttons* associated with the *Choices combination list box* (fig. 22) provide a variety of options for controlling how antecedent data links and networks are used in an evaluation.

- *Accept*: Accepts the current value of the data link or network that has been entered on the *edit line* or highlighted in the *list box* of the *Choices combination list box*. If a value was entered previously, it is replaced by the new value.

- *Append*: If a data link has no previously assigned data value, the function of *Append* is equivalent to *Accept*. If a data link does have a previously assigned value, then the data link now has two values associated with it. *Append* does not apply to networks.
- *Revert*: Restores the last value entered for the data link or network.
- *Ignore*: Ignores the currently selected data link or network in the *Influence list box* or *Other antecedents list box* during evaluation of the network. This is a powerful option that should be used with caution. Choosing to ignore data links or networks deletes the item from the network in the current evaluation (it is not deleted from the knowledge base, however). On the other hand, there may be occasions when such an action is justified. If a data link or network is ignored, it is displayed in the *Other antecedents list box* with a strike-through.
- *Skip*: When an evaluation is run in step mode, NetWeaver queries the user to input values to data links, with the order of query determined by descending data link influence. Pressing the *Skip button* (only functional in step mode), informs NetWeaver that you want to defer providing a data value for the data link. If a data link is skipped, it is displayed in the *Influence list box* with a strike-through. A skipped item can be highlighted manually in the *Influence list box* at any later time during an evaluation by clicking on the item.

Manual evaluation versus step mode—Evaluation of a network or evaluation group can be performed in one of two modes:

1. Data links and networks can be selected manually one at a time, by the user from the scrolling *Influence list box* in the *Influence pane* (bottom pane in the evaluation window) labeled “Influential antecedents,” or
2. In step mode, NetWeaver will prompt the user to enter a value for the most influential data link.

To put NetWeaver in step mode, press the *Step button* in the lower right corner of the *Evaluation window*. To halt step mode at any time and revert to manual mode, select the *Stop button*. In general, an evaluation can be switched freely between manual and step mode. Moreover, while in step mode, the order of query can be overridden by selecting a data link from the *Influence list box*. After processing such a user-selected input, NetWeaver automatically reverts to step mode.

Data Links

Data input to a knowledge base is accomplished through data links that, in important respects, are simply elementary dependency networks. As with dependency networks, the primary attribute of a data link is its truth value, although data links also may not evaluate a datum, but simply pass the value on to another NetWeaver object, such as a calculated data link. Analogous to a dependency network, data links broadcast their changing states to their dependent networks. Data links may be either simple or calculated.

Getting Data Into a Knowledge Base

When a data link is encountered during evaluation of a dependency network in the NetWeaver development environment, NetWeaver queries the user for the relevant data via the *Evaluation window*. Data values also can be entered in the *Data link evaluation window* or applied to data links through data base linkages when the NetWeaver engine is running under the control of an external application, such as the EMDS system.

Data Influence

Data input to a knowledge base influences the evaluation of networks in the sense that data determine the truth values of any dependent networks. In NetWeaver, influence is specifically defined to mean influence of missing data. The semantics of the formal logical model underlying NetWeaver make it possible to compute a measure of relative influence for unsatisfied data requirements that have been defined for a given knowledge base. Note that influence of missing data is a dynamic variable. In general, it depends on what data are already available and the values input for those data.

When evaluating a network, NetWeaver uses information about the influence of missing data to direct the order of data requests when a knowledge base is being evaluated in step mode (see “Evaluating a Network Object,” above). Other software applications (for example, EMDS) that use the NetWeaver inference engine also can make use of information about data influence to guide analyses in various ways.

Types of Data Links

Simple data link—A simple data link can read a data value that it passes to another data link without evaluation or it can compute a truth value by evaluating the datum against an argument list. A simple data link whose data value is used directly usually passes its data value to a calculated data link.

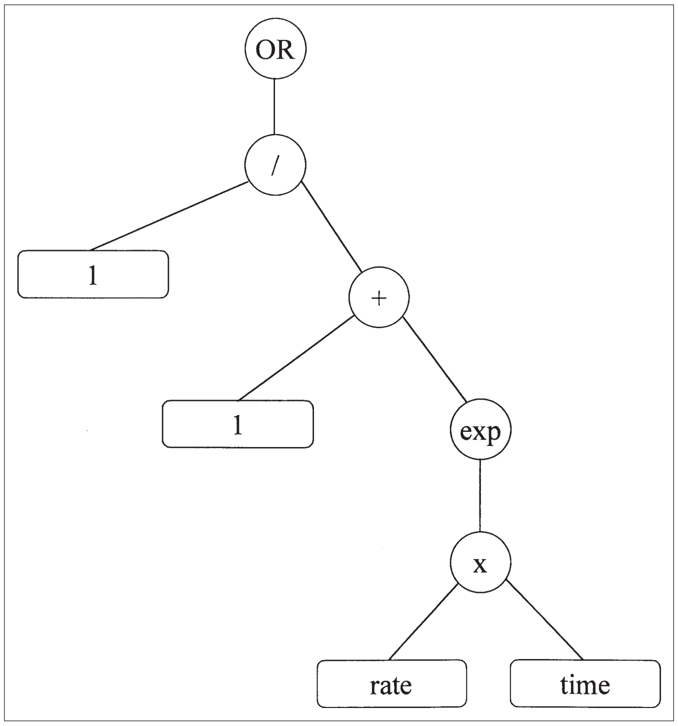


Figure 23—A calculated data link that evaluates the expression 1/(1 + exp(-rt)). The value of t comes from a simple directly link whose value is directly used by the calculated data link.

When data input to a simple data link is not used directly, it is compared to an argument to compute a truth value for the data link. A simple argument specifies a condition that the datum is tested against. If the argument is “<= 10”, for example, and the value input to the data link satisfies this condition, then the truth value of the data link evaluates to 1 (completely true). Data input also may be evaluated against a fuzzy argument, as discussed below.

Calculated data link—A calculated data link transforms input data. It can hold either a truth value or a data value that it passes directly to another calculated data link without evaluation. In either case, a calculated data link has one or more function nodes that connect two or more data links or networks (fig. 23). The functional representation of a calculated data link is built graphically in the *Calculated data link window* (see the description of this window later in this chapter), in a similar way to dependency networks. As with networks, the functional representation of a calculated data link is displayed in its own graphic window.

When data input to a calculated data link is not used directly, it is compared to an argument to compute a truth value for the data link. A simple argument specifies a condition that the datum is tested against. If the argument is “<= 10,” for example, and the value satisfies this condition, then the truth value of the data link evaluates to 1 (completely true). Data input also may be evaluated against a fuzzy argument.

Example: Figures 24 and 25 illustrate a simple example of a calculated data link. In figure 24, a calculated data link named “road density” has been created. Note that “road density” has an argument, “ <= 0.1.” The data value input to “road density” is not used directly. Instead, the result of calculations input to “road density” is compared to the argument to compute a truth value for “road density.” The functional structure of the calculated data link that actually calculates the data input to “road density” is illustrated in figure 25. A function node (/) hangs from the calculated data link and connects the two data links, “miles of road” and “land area.” Note the orientation: dividend on the left, and divisor on the right. The function reads from left to right.

Data Link Window

Usage—Use this *dialog window* (fig. 26) to define available arguments for a simple data link or calculated data link when the data link is created, or to subsequently edit the list of arguments in the data link by adding and removing arguments.

A data link is not actually used in the knowledge base until it is referenced in some network in the knowledge base. A reference to a data link is added to a network by pressing the *Data link button* on the *Network window button bar* and choosing an existing data link from the *pop-up menu*. On selecting a data link, NetWeaver automatically displays the *Data link reference window* (see the description of this window later in this chapter).

Access—When a simple data link or calculated data link is created, the *Data link window* is automatically displayed after the documentation window for the new data link is closed. To edit the data link, in the *Knowledge base window*:

1. Select the data link in the *Data folder* or *Calc folder*.
2. Choose *Object* on the *Edit menu* of the *Application menu bar*.

To edit the data link, in a *Network window*:

1. Right click on the graphic representation of the referenced data link.
2. Choose *Local edit* from the *pop-up menu*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

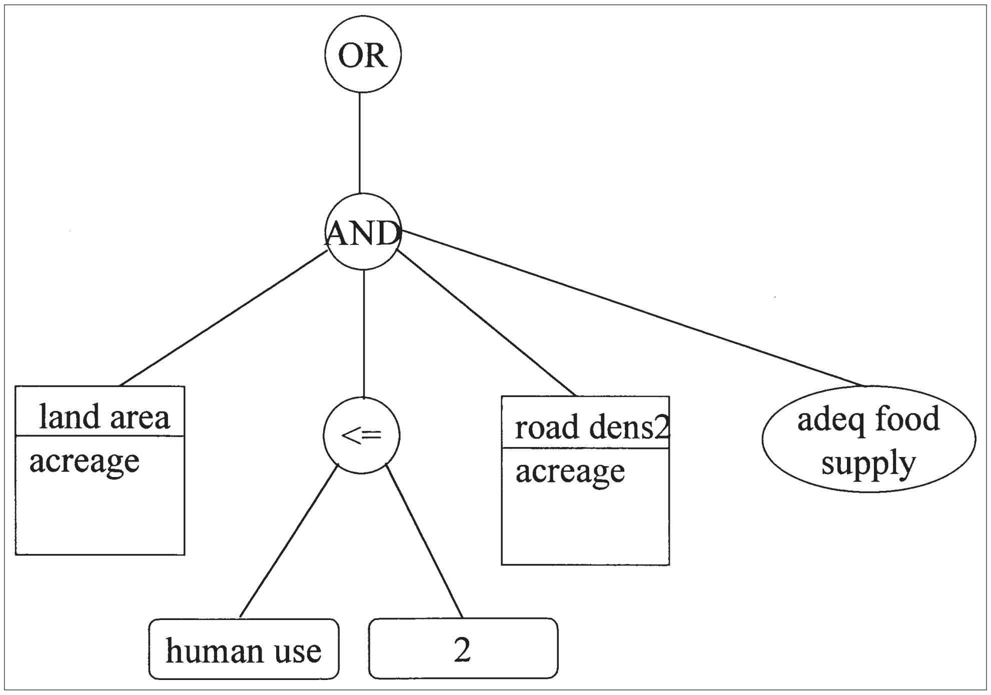


Figure 24—Road density is a calculated data link. Data input to the data link is compared to an argument (<= 0.1).

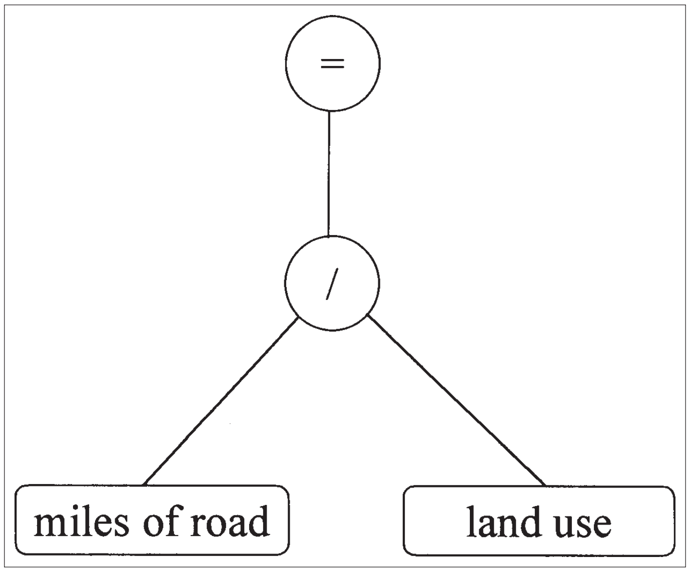
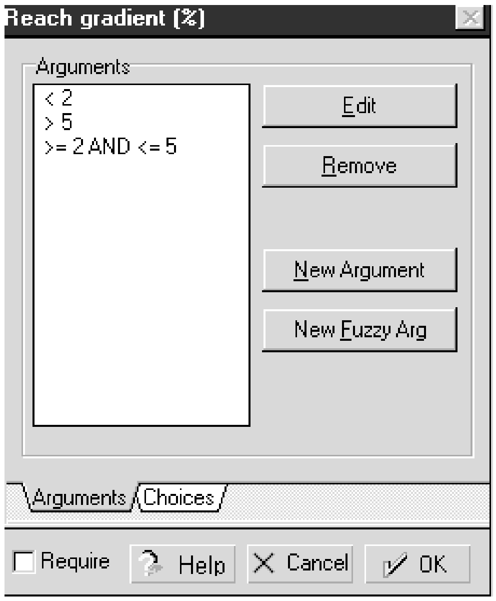


Figure 25—The functional structure of the calculated data link used to compute an input to road density in figure 27.



```
< 2
> 5
>= 2 AND <= 5
```

Figure 26—The Data link window is used to initially enter, or subsequently edit, the arguments of a data link. The Simple argument and Fuzzy argument windows are opened from this window.

Adding new arguments to a data link—One or more arguments may be defined for a data link. A single data link object may have multiple alternative arguments because arguments are not actually used until the data link is referenced somewhere in the knowledge base. Each reference to a data link potentially could use a different argument. Arguments created for a data link object are displayed in the *Available arguments list box* of the *Data link reference window* (see the description of this window later in this chapter).

To create a new simple argument, press the *New Argument button* (fig. 26).

To create a new fuzzy argument, press the *New Fuzzy Argument button*.

Editing arguments—Click on an argument to highlight it in the *Argument list box*, then press the *Edit button* (fig. 26).

Removing arguments—To remove one or more arguments from a simple data link, select the items in the *Arguments list box* and press the *Remove button* (fig. 26). The selected items are removed from the *Arguments list box*.

Arguments that have been created for a data link can be deleted if there are no references to the data link in the knowledge base that use the argument. If an attempt is made to remove an argument from a data link and the argument is actually used in a reference to the data link, NetWeaver displays a *dialog window* advising that it cannot remove the argument because it is in use.

Defining choices—Choices are a predefined set of discrete alternatives (for example, yes/no or poor/fair/good) from which an answer can be selected when a data link is being evaluated. The primary reason for defining choices is that there may be only a limited number of discrete choices for legitimate data values for a data link, and the knowledge base designer wants to prompt the user to choose from this list rather than entering some other value.

Press the *Choices tab* in the *Data link window* to edit or view the list of choices. Choices defined in the *Choices list box* are displayed in the list box portion of the *Choices combination list box* in the *Influence pane* of the *Evaluation window* when the data link is evaluated.

Providing choices is optional. If choices have not been provided for a data link, the user can simply enter a value on the *edit line* of the *Choices combination list box* in the *Influence pane* of the *Evaluation window*. A specification of choices in the *Choices folder* does not preclude a user from entering some other value.

Any discrete arguments that have been defined for a data link can be imported to the choice list by pressing the *Import from args button* in the *Choices folder*. Additional choices can be typed directly in the *Choices folder* to specify conditions other than those that make a data link evaluate to true. As an example, suppose there is a data link named “habitat” and an argument “=good” has been defined for it (the data link will evaluate to true if the user selects the choice “good”). Pressing the *Import from args button* will put the argument on the choice list. In this example, also suppose that there is no need to define the argument “=bad” for the habitat data link because the knowledge base designer never needs to test for this condition explicitly. The choice “bad” can be typed directly into the choices list so that, when the habitat data link is evaluated, the two choices, bad and good, are displayed in the *list box* portion of the *Choices combination list box* in the *Influence pane* of the *Evaluation window*.

Calculated Data Link Window

Usage—The *Calculated data link window* is a special case of the *Network window* in which *buttons* for mathematical operators are enabled on the *Network window button bar*.

Access—Double click on the calculated data link in either the *Calcs folder* of the *Knowledge base window* or its graphic representation in a *Network window*.

Defining the structure of a calculated data link—To define the structure of a mathematical expression that provides input to a calculated data link:

1. Press the *Function button* on the *Network window button bar*.

2. Choose a mathematical operator from the *pop-up menu* to attach to the calculated data link.
3. Choose another math operator or an operand (for example, a simple data link, another calculated data link, etc.) to attach to the math operator added at step 2.
4. Repeat steps 1 through 3 until all lowest level math operators in the expression have an appropriate number of operands.

There are a few restrictions on how calculated data links are constructed:

1. One, and only one, math operator can be attached to the calculated data link itself.
2. Only a math operator can be directly attached to a calculated data link.
3. The number of operands that can be attached to a math operator depends on the operator (see “Function Nodes” in “Relations Among Objects,” below, for specifications).

For an example of how to construct a moderately complex expression for a calculated data link, see figure 23 in which the data link calculates the value of the logistic function given input from a simple data link, t.

Data Link Reference Window

A data link is not actually used in the knowledge base until it is referenced in some network in the knowledge base. A reference to an existing data link is added to a network by pressing the *Data link button* on the *Network window button bar* and selecting an existing data link from the *pop-up menu*. After selecting a specific data link, NetWeaver automatically displays the data link reference window.

Usage—Use this *dialog window* (fig. 27) to choose a specific argument or set of arguments from the *Available arguments list box* and add them to the *Arguments list box*. The arguments appearing in the *Arguments list box* are used to actually evaluate the truth value of the data link. Arguments may be simple arguments or fuzzy arguments. New arguments for the data link object also can be created in this window.

Access—NetWeaver displays the *Data link reference window* when a data link reference is added to a *Network window*. To edit arguments used by a referenced data link:

1. Right click on the graphic representation of the referenced data link in the *Network window*.

2. Choose *Local edit* from the *pop-up menu*.

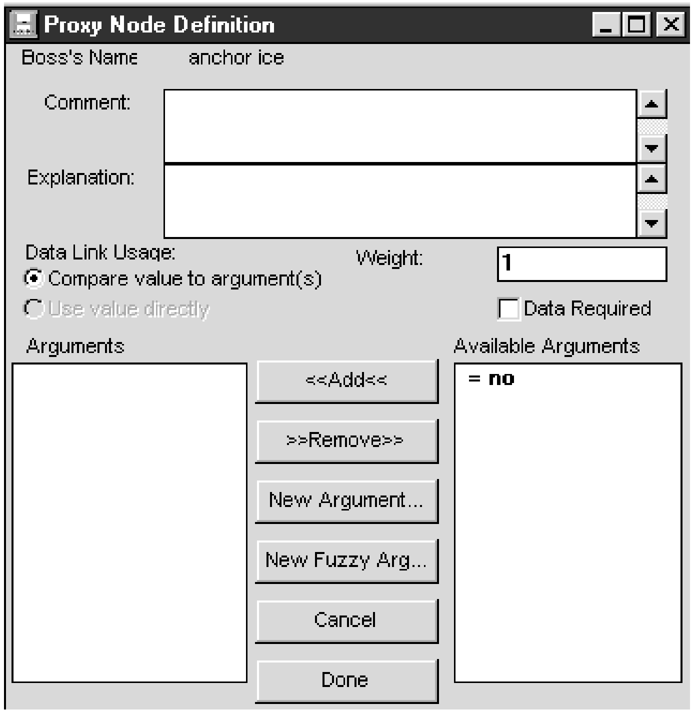


Figure 27—Data link reference window.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Adding and removing arguments—To add an existing argument to the referenced data link, select the argument in the *Available arguments list box* and press the *Add button* to add the selection to the list in the *Arguments list box*.

To remove an argument, select the argument in the *Arguments list box* and press the *Remove button* to move the selected item back into the *Available arguments list box*.

Creating new arguments—It also is possible to create new simple arguments and new fuzzy arguments from the *Data link reference window* by pressing the appropriate *New button*. For further details, see descriptions of the *Simple argument window* and *Fuzzy argument window* later in this chapter.

Table 2—Logical and string operators used to construct a simple argument

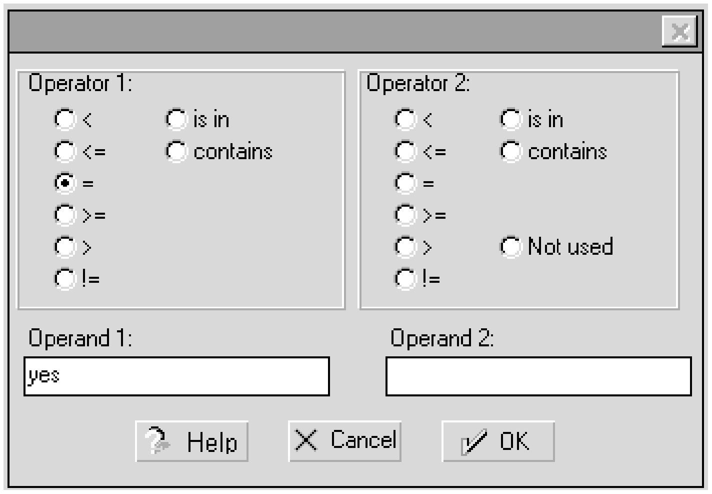
Symbol	Meaning
<	Less than
<=	Less than or equal to
==	Equal to
>=	Greater than or equal to
>	Greater than
!=	Not equal to
is in	Data are a substring of the string argument. The data link evaluates to true if the argument is “treehouse” and the data input is “house.”
contains	Data include the argument as a subset. The data link evaluates to true if the argument is “house” and the data input is “treehouse.”

Data Link Arguments

A data link either can use data directly, in which case the value input to the data link is propagated upward in the network, usually to a calculated data link, or it may compare the value input to an argument. The argument of a data link may be either a simple argument or a fuzzy argument.

Using data directly—When a data link is first created, you have the option to either compare the data value to an argument or to use the data directly. When data are used directly, the value input is propagated upward in the network to a calculated data link.

Simple argument—A simple argument is constructed from an operator (table 2) and an operand, and specifies a condition that data input to a simple data link or calculated data link must satisfy for the data link to evaluate to true. If the condition is met, then the truth value of the data link is 1 (completely true); otherwise, the truth value of the data link is -1 (completely false).


Operator 1: < | <= | [✓ =] | >= | > | != | is in | contains

Operator 2: < | <= | = | >= | > | != | is in | contains | Not used

Operand 1: yes

Operand 2:

Figure 28—Simple argument window.

Simple argument window—

Usage—To specify an argument for a data link in the *Simple argument window* (fig. 28):

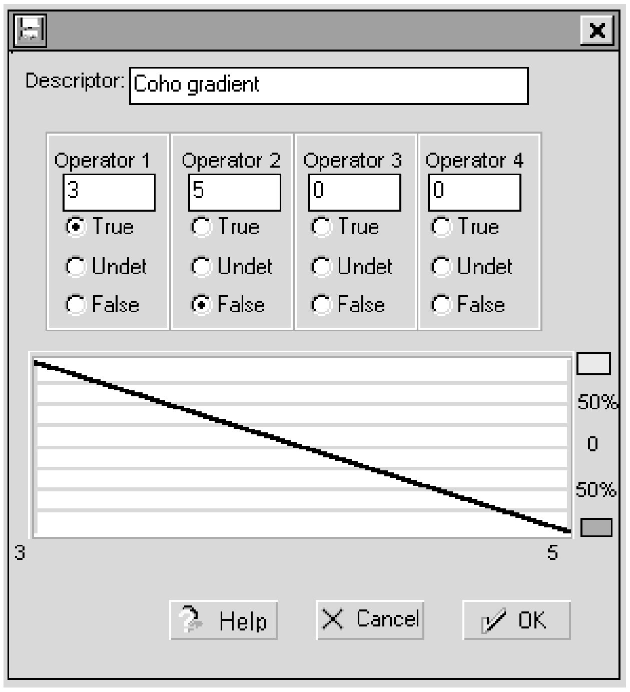
1. Select an operator from the *Operator 1 group box*.
2. Enter an operand on the *edit line*.
3. Optionally, select another operator from the *Operator 2 group box* and enter an operand for it.

If two expressions are specified, they are logically ANDed together; that is, the two expressions both must be true for the truth value of the simple data link to evaluate to 1 (completely true).

Choose the *OK button* to complete the specification of the argument and close the *dialog window*. Choose the *Cancel button* to abandon any edits that have been made to the argument specification.

Access—The *Simple argument window* (fig. 28) is accessed from the *Data link window* or the *Data link reference window* by pressing the *New Argument button*, or by selecting an existing simple argument and pressing the *Edit button*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.


Descriptor: Coho gradient

Operator 1	Operator 2	Operator 3	Operator 4
3	5	0	0
[✓ True]	True	True	True
Undet	Undet	Undet	Undet
False	[✓ False]	False	False

Figure 29—Fuzzy argument window.

Fuzzy argument—A fuzzy argument defines a fuzzy membership function with up to four points that describe a data value's degree of membership in a fuzzy set. If a data link has a fuzzy argument instead of a simple argument, as discussed in the previous section, then the data value is compared to the fuzzy membership function to compute its membership in the fuzzy set defined by the fuzzy function.

Fuzzy argument window—

Usage—To specify a fuzzy membership function for a data link in the *Fuzzy argument window* (fig. 29):

1. Enter a phrase in the *Descriptor edit box* that describes the proposition evaluated by the function.
2. In the *Operator 1 group box*, enter an x value in the *edit box* and press one of the three *radio buttons* (true, false, or undetermined) to define the truth value (y) that corresponds to x.

3. Repeat the procedure in step 2 for the *Operator 2 group box* to define a second point on the function.
4. Optionally, continue defining up to two additional xy-pairs in the *group boxes* for operators 3 and 4.

A minimum of two points must be specified to define a fuzzy membership function. In the example (fig. 29), the data link evaluates to 1 (completely true) if the value entered for reach gradient is less than or equal to 3. The data link evaluates to -1 if the value entered for reach gradient is greater than or equal to 5. For values of reach gradient between 3 and 5, the data link evaluates to some value between -1 and 1, thereby indicating some degree of partial membership in the Coho gradient fuzzy set.

The fuzzy curve (see the later section, “Fuzzy Curve Nodes”) is an alternative way of defining fuzzy membership functions that offers some advantages over the use of a fuzzy argument. In particular, a fuzzy curve is not limited to four points, and it allows the function to be defined dynamically; that is, by data input at the time of network evaluation.

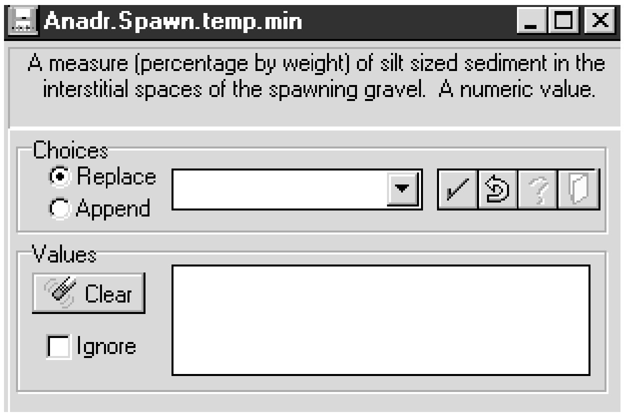
Access—The fuzzy argument *dialog window* (fig. 29) is accessed from the *Data link window* or the *Data link reference window* by pressing the *New Fuzzy Argument button*, or by selecting an existing fuzzy argument and pressing the *Edit button*.

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Data Link Evaluation Window

***Usage*—**Because data links are really just elementary networks, they can be evaluated; however, the *Data link evaluation window* (fig. 30) is much simpler than the *Evaluation window* for networks and evaluation groups. The upper *Comment pane* of the *Data link evaluation window* displays the comment associated with the data link (see the earlier section, “Documentation window”). The lower *Values pane displays* the current value or set of values assigned to the data link if it has been evaluated. Press the *Clear button* to reset the data link to an unevaluated (undetermined) state. Select the *Ignore checkbox* to ignore the data link. The central *Choices pane* is used to append a new data value or replace the current value with a new one:

1. Enter a data value on the *edit line*.
2. Press either the *Replace* (default) or *Append radio button*.
3. Press the *Accept button* to evaluate the data link.


Anadr.Spawn.temp.min

A measure (percentage by weight) of silt sized sediment in the interstitial spaces of the spawning gravel. A numeric value.

Figure 30—Data link evaluation window.

Access—To open the Data link *evaluation window* from the *Knowledge base window*:

1. Click on a data link in either the *Calcs* or *Data folder* to highlight it.
2. Either choose *Evaluate object* on the *Evaluate menu* of the *Application menu bar* or press the *Eval button* on the *Knowledge base window button bar*.

To open the *Data link evaluation window* for a data link displayed in a *Network window*:

- Either right click on the graphic representation of the data link reference and choose *Evaluate* from the *pop-up menu*.
- Or click on the graphic representation of the data link reference to highlight it and press the *Eval button* on the *Network window button bar.*

Press the *OK button* to close the *dialog window*. Press the *Cancel button* to exit the *dialog window* and abandon any editing changes that have been made.

Relations Among Objects

Fuzzy Logic Nodes

Fuzzy Node Overview

NetWeaver uses OR, AND, NOT, and XOR and SOR logic nodes to define the logical dependency of a network on antecedent networks and data links. If a network does not use any fuzzy arguments, then the operation of these nodes conforms quite closely to their usage in conventional logic. The only real difference between these nodes, as used in NetWeaver, and standard logic is that true = 1 and false = -1 in NetWeaver.

NetWeaver allows simple or calculated data links to take fuzzy arguments to determine a data value's membership in a fuzzy set. For fuzzy set membership to be propagated through a knowledge base, the definitions of the conventional logical operators OR, AND, NOT, and XOR have been extended to handle measures of fuzzy set membership. The SOR node object is unique to NetWeaver.

Fuzzy OR Node

An OR node evaluates to the maximum truth value associated with a set of antecedents connected to the OR node. For example, if A, B, and C are immediate antecedents of an OR node, and they have truth values of 0.8, 0, and -1.0, respectively, then OR evaluates to 0.8.

Fuzzy AND Node

An expression containing the conventional logical AND is true if all conditions joined by AND are true, and false otherwise. In NetWeaver, when the antecedents of a fuzzy AND node are either all fully true or at least one is fully false, the node behaves like the conventional AND. More generally, the fuzzy AND is calculated as:

AND(t) = min(t) + [average(t) - min(t)]*[min(t)+1]/2 ,

in which AND(t) is the truth value of the AND node, min(t) is the minimum truth value of the AND node's antecedents, and average(t) is a weighted average of the truth values of the AND node's antecedents. The equation for calculating the value of a fuzzy AND is designed to produce a conservative estimate of truth in the presence of missing or partial negative evidence. For example, if data links A and B have truth values of 1 (completely true) and 0 (undetermined), respectively, then AND(A, B) evaluates to 0.25. In other words, there is a penalty for missing information that prevents the evaluation of the AND from being overly optimistic. The justification for this conservative formulation is intuitively appealing. When there is no evidence concerning the truth of B, as in our example, the conservative value of 0.25 is preferable to the simple average of truth values for A and B (0.5) because B later might be found to be completely false, in which case the fuzzy AND should evaluate to completely false, regardless of whether the evaluation is based on crisp or fuzzy logic.

Fuzzy NOT Node

The NOT node is a unary operator (can have only one immediate antecedent) that negates the truth value of its antecedent. Networks are constructed to evaluate whether some condition is true, but at times it also can be useful to test whether something is not true. Negating data links is generally not necessary because it is always possible to construct an argument that directly evaluates the negative condition. On the other hand, the fuzzy NOT is a convenient way of inverting the truth value of a data link rather than constructing another argument. For example, if a data link for percentage of slope already has been created that evaluates data input against the argument "<=10," and a new reference to the data link now needs to evaluate the alternative condition ">10," then the data link reference can be attached to the fuzzy NOT node.

Fuzzy XOR Node

The function of XOR is to evaluate the degree to which one, and only one, of its immediate antecedents is true. Three rules specify the behavior of XOR (table 3):

1. XOR evaluates to 1 (completely true) if one, and only one, immediate antecedent is completely true, and all other immediate antecedents are completely false.

Table 3—Evaluation of a fuzzy XOR node

Data link truth values			
A	B	C	XOR
1.0	-1.0	-1.0	1.0
1.0	0.5	0.2	0.5
0.5	0.5	0.2	0.0
1.0	1.0	0.2	-1.0
1.0	1.0	1.0	-1.0
1.0	-1.0	-1.0	-1.0

2. XOR evaluates to -1 (completely false) if all immediate antecedents are completely true or completely false.
3. In general, the value returned by XOR is a measure of the distance between the two most true antecedents. However, XOR is completely false if any two antecedents are completely true.

Fuzzy SOR Node

The sequential OR node (SOR) provides a mechanism for choosing among alternative networks and data sources as input to a knowledge base. The leftmost antecedent of the SOR node is the most preferred source, the second antecedent from the left is the next most preferred source, and so on.

A SOR node selects data values as input from the first antecedent or logic node (reading from left to right) with sufficient data. The weight associated with the antecedent's reference specifies the proportion of data that must be available in its antecedents for the antecedent to be selected as the preferred logic path. Because the weight of a reference defaults to 1, the SOR node typically uses the first network or logical node having all its data available. If no immediate antecedent of a SOR node has sufficient data, then the SOR node remains undetermined.

As an example, consider an SOR node with two antecedent networks, A and B (fig. 31) and six different cases (table 4). In case A, each network, A and B, has weight = 1.0, each network has all its data available (data = 1.0), and network A is selected because it is first in order of preference (fig. 31). In case B, networks A and B again each have weight = 1, but network B is selected because network A does not have sufficient data (data < weight). In case C, network A does not have all its data (data = 0.5), but it is still selected over network B because its availability of data exceeds its weight (data > weight). Case D is similar to case B. Case E is similar to case A. Finally, in case F, SOR remains undetermined because neither network has sufficient data to meet its weight criterion.

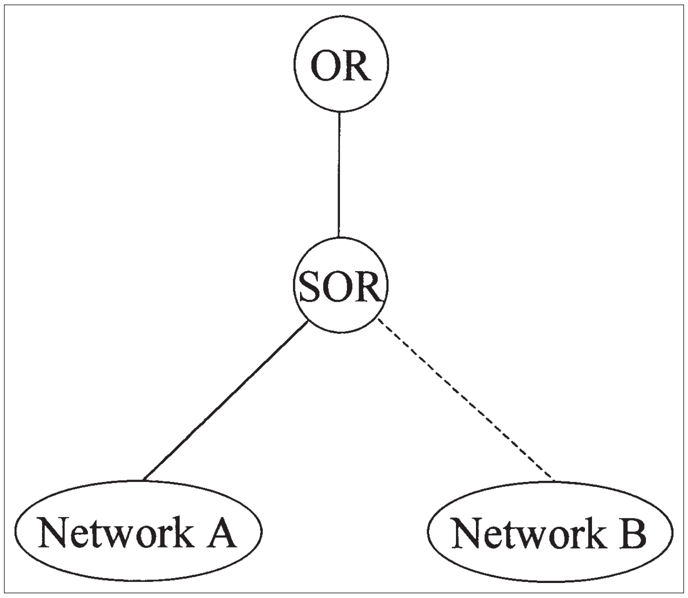


Figure 31—A SOR node with two alternative networks, A and B. See table 4 for examples of how SOR is evaluated under various conditions of network weights and data availability.

Table 4—Evaluation of a SOR node

	Network A			Network B			
Case	Weight	Data	Truth	Weight	Data	Truth	SOR
A	1.0	1.0	1.0	1.0	1.0	0.5	1.0
B	1.0	0.5	0.2	1.0	1.0	0.75	0.75
C	0.5	0.6	0.2	1.0	1.0	0.5	0.2
D	1.0	0.8	0.2	1.0	1.0	0.75	0.75
E	1.0	1.0	-1.0	1.0	1.0	1.0	-1.0
F	1.0	0.5	0.25	1.0	0.8	0.5	0.0

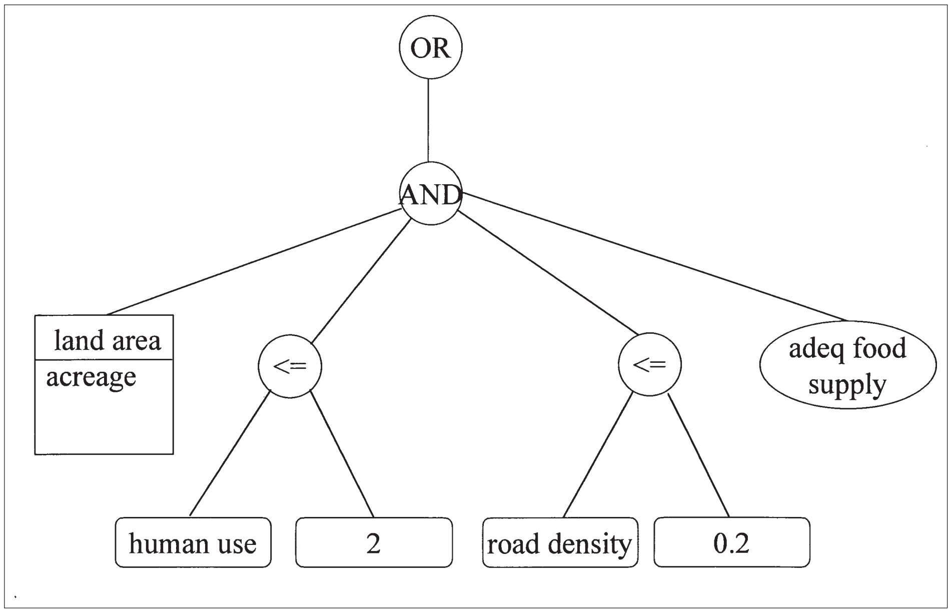


Figure 32—An alternative way to evaluate road density by comparing a data value to a constant. Compare to figures 27 and 28.

Other Types of Relational Nodes

Constant Nodes

The constant is a special case of the data link in which the data value is fixed. The data value of a constant is actually a part of the knowledge base. Typically, constants are used as an antecedent of a comparison node or a function node (see descriptions of these nodes below), or as an element of the mathematical expression of a calculated data link. To create a constant:

1. Click on a logical node in the *Network window* to highlight the node.
2. Press the *Constant button* on the *Network window button bar* and choose "*k*" from the *pop-up menu* to attach a constant to the highlighted node.
3. The constant is initially inserted into a network with a value of zero. To change the value, right click on the graphic representation of the constant and choose *Local edit* on the pop-up menu.
4. In the Local edit window for the constant, enter a value for the constant in the *Name/type edit box*.
5. Press the *OK button* to complete editing of the constant.

As step 4 implies, the name of a constant also is its value. Although the name is a character string, NetWeaver performs any necessary type conversion if the constant is used in an arithmetic expression. As an example (fig. 32), the calculated data link “road density” contains an actual data value (instead of a truth value), which is compared to the constant “0.2” to compute a truth value for the comparison node. For an alternative way of modeling road density input with a calculated data link that contains an argument, see figures 24 and 25.

Switch Nodes

A switch node selects among alternative networks and data links based on the value of an associated data link. Antecedents attached to a switch represent alternative network paths, one of which is selected based on the data value input to the data link that is associated with the switch. To construct a switch node in a *Network window*:

1. Click on a logical node in the *Network window* to highlight the node.
2. Press the *Switch button* on the *Network window button bar*.
3. From the *pop-up menu*, choose an existing data link or choose one of the *New* options on the *pop-up menu* to associate a specific data link with the switch.
4. If the data link is new, complete the *Documentation window* for the new data link.
5. After closing the *Documentation window*, if applicable, create new arguments or select existing arguments in the *Data link reference window* to apply to the switch node.
6. Press the *OK button* in the *Data link reference window* to close the window and complete the switch specification.

Like a simple data link, NetWeaver will query the user for a data value for the data link associated with a switch when the network is evaluated. The data value supplied to the switch is compared to the argument list of the data link. If the data value matches arguments 1, 2, 3, etc., then the corresponding pathway through the switch (read left to right) is evaluated. Figure 33 illustrates a switch that uses a data value obtained from a data link named “scenario switch.” Arguments defined for the data link are “=A,” “=B,” and “!=A AND !=B,” and these arguments are displayed in the switch node. The last argument is read “not equal to A and not equal to B,” and is equivalent to “anything other than A or B.” When this switch is evaluated, it will prompt the user to indicate which scenario applies. If the user selects choice “A,” then the network “scenario A” is evaluated. Networks B and C would not be evaluated in this example.

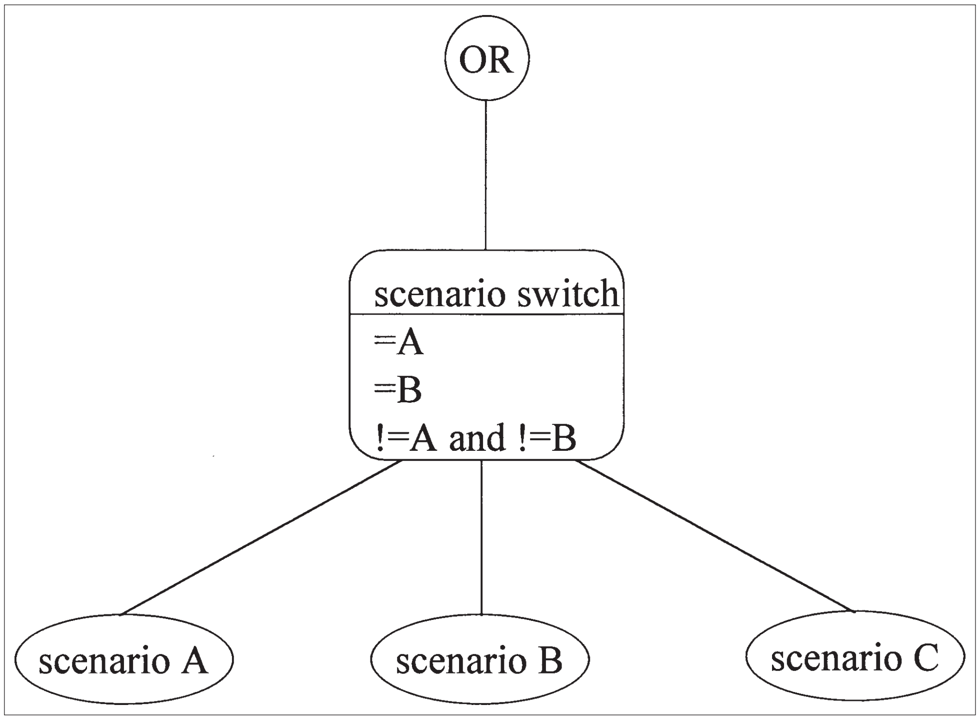


Figure 33—A switch to choose among three scenarios (A, B, and C).

If you intend that one of the networks must be evaluated, be sure that the argument list of the switch is exhaustive (covers all possible cases), as in the above example. If, instead of the argument “!= A AND !=B,” “=C” had been the third argument to “scenario switch,” and the user had elected to type “D” rather than choose from the list of available choices, then none of the scenarios under “scenario switch” would have been evaluated because the choice did not match any of the switch arguments. In some cases, this may be an appropriate result, but often it is not.

Fuzzy Curve Nodes

The fuzzy curve is a generalization of the fuzzy argument in a data link. Simple and calculated data links both allow the specification of a fuzzy argument which is used to determine a data value’s degree of membership in a fuzzy set. However, in the fuzzy argument specification for a data link, definition of the curve is limited to four points (that is, it defines a simple ramp function). The fuzzy curve node allows a more precise specification of the shape of the fuzzy surface because the number of points is unlimited for all practical purposes. Another important distinction is that the points of a fuzzy curve can be defined dynamically. To create a fuzzy curve in a Network window:

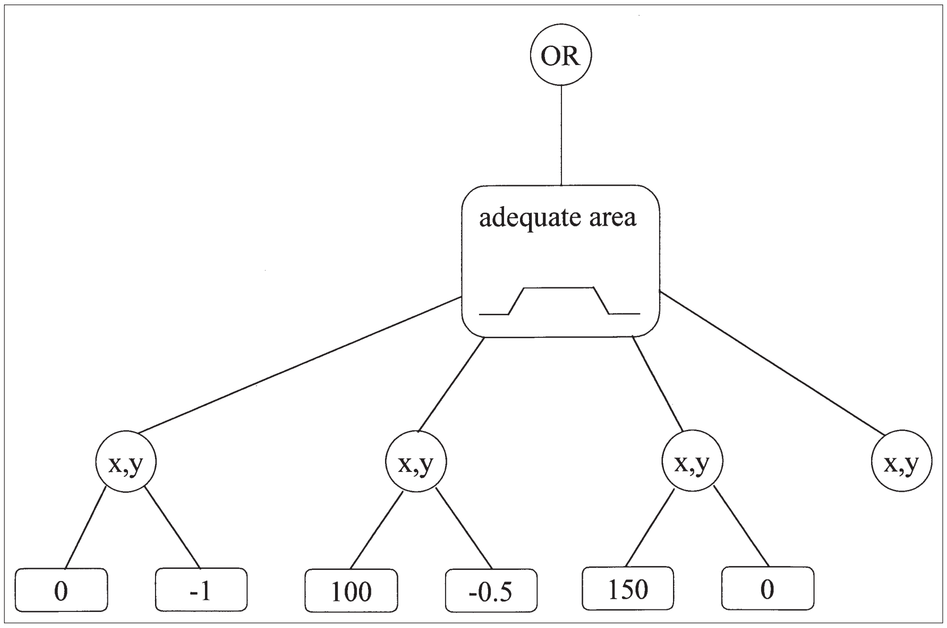


Figure 34—A fuzzy curve with a few xy pairs.

1. Click on a logical node in the *Network window* to highlight the node.
2. Press the *Fuzzy curve button* on the *Network window button bar* to attach the fuzzy curve to the highlighted node.
3. From the *pop-up menu*, choose an existing data link or network or choose one of the *New options* on the *pop-up menu* to associate a specific data link or network with the fuzzy curve.
4. Press the *Fuzzy curve button* on the *Network window button bar* again and choose *xy pair* to attach an xy pair node to the fuzzy curve. Repeat this step for as many xy pairs as required.
5. For each xy-pair node, click on the node to highlight it, and attach a new or existing data link, constant, or network that defines x and another that defines y.

In the two objects that define an xy pair, the left member is the x term, and the right member is the y term. The curve shape can be fixed by using constants to define each x and y (fig. 34). The curve shape can, however, be dynamically determined during an evaluation by reading input from data links or even using the truth values of networks.

Table 5—Logical comparators used in comparison nodes

Symbol	Meaning
<	Less than
<=	Less than or equal to
==	Equal to
>=	Greater than or equal to
>	Greater than
!=	Not equal to

Comparison Nodes

A comparison node is a binary operator that performs a logical comparison of the values of two antecedents. Objects involved in a comparison may be any pairwise combination of simple data links, calculated data links, constants, dependency networks, and function nodes. A standard set of logical comparators is available from which to select (table 5).

The truth values of two networks, A and B, are compared (fig. 35) to test for truth(A) less than (<) truth(B). If the comparison evaluates to true, the comparison node is set to true (1), otherwise it is set to false (-1). Note that the graphic representation (fig. 35) reads from left to right.

Function Nodes

A function node can be either a unary operator (for example, sin and cos) that operates on a single argument, a binary operator such as “/” that operates on two arguments, or an operator that can take one or more arguments such as “+” and “*” (table 6).

Function nodes apply only to calculated data links and comparison nodes, so the *Function button* on the *Network window button bar* is generally disabled unless a comparison node is highlighted in a *Network window* or unless another suitable function node is highlighted in a *Calculated data link window*.

In figure 24, a calculated data link named “road density” has been created. “Road density” has an argument, “ <= 0.1.” The data value input to “road density” is not used directly. Instead, the result of calculations input to “road density” is compared to the argument to compute a truth value for “road density.” The functional structure of the calculated data link actually calculates the data input to “road density” (fig. 25). A function node (/) is attached to the calculated data link and connects the two data links, “miles of road” and “land area.” Note the orientation: dividend on the left, and divisor on the right. The function reads from left to right.

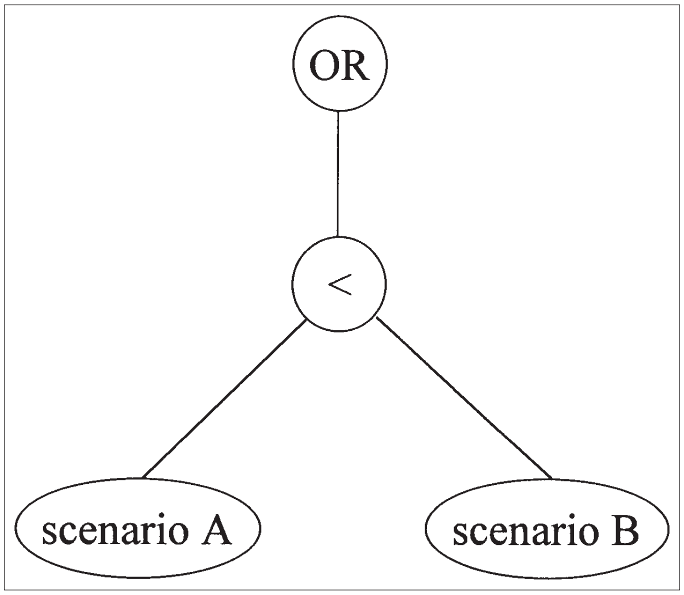


Figure 35—Use of a logical comparator to compare the truth value of two networks.

Table 6—Functional nodes available in NetWeaver

Symbol	Arguments	Operation
Arithmetic operators:		
+	1..n	addition
-	2	subtraction
*	1..n	multiplication
/	2	division
abs	1	absolute value
Trigonometric operators:		
cos	1	cosine(radians)
sin	1	sine(radians)
tan	1	tangent(radians)
cosh	1	hyperbolic cosine(radians)
sinh	1	hyperbolic sine(radians)
tanh	1	hyperbolic tangent(radians)
acos	1	arc cosine(radians)
asin	1	arc sine(radians)
atan	1	arc tangent(radians)

Table 6—Functional nodes available in NetWeaver (continued)

Symbol	Arguments	Operation
Deg2Rad	1	convert from degrees to radians
Rad2Deg	1	convert from radians to degrees
Log and power operators:		
ln	1	natural logarithm (base e)
log	1	logarithm (base 10)
exp	1	e raised to the power of the exp argument
pow	2	first argument raised to power of second
sqrt	1	square root
Integer operators:		
ceil	1	smallest integer greater than or equal to argument
floor	1	largest integer less than or equal to argument
mod	1	modulus (remainder from integer division)
Set operators:		
min	1..n	minimum
max	1..n	maximum
ave	1..n	average: *ave(0,0,0,4,3,5) = 2*
ave-nz	1..n	average of non-zero values: ave-nz(0,0,0,4,3,5) = 4

Literature Citations

Barr, A.; Cohen, P.R.; Feigenbaum, E.A. 1989. Artificial intelligence, volume IV. Reading, MA: Addison-Wesley Publishers. 699 p.

Gall, J. 1986. Systematics: how systems really work and how they fail. Ann Arbor, MI: The General Systematics Press. 356 p.

Jackson, P. 1990. Introduction to expert systems. Reading, MA: Addison-Wesley Publishers. 526 p.

Reynolds, K.; Cunningham, P.; Bednar, L. [and others]. 1996. A design framework for a knowledge-based information management system for watershed analysis in the Pacific Northwest U.S. AI Applications. 10: 9-22.

Saunders, M.C.; Rajotte, E.G.; Travis, J.W. 1989. The use and significance of expert systems in small fruit IPM. In: Bostanian, N.J.; Wilson, L.T.; Dennehy T.J., eds. Monitoring and integrated management of arthropod pests of small fruit crops. London: Intercept.

Saunders, M.C.; Coulson, R.N.; Folse, J. 1990. Applications of artificial intelligence in agriculture and natural resource management. In: Kent, A.; Williams, J., eds. Encyclopedia of computer science and technology. New York: Marcel Dekker Inc.: 1-14. Vol. 25, suppl. 10.

Schmoldt, D.L.; Rauscher, H.M. 1995. Building knowledge-based systems for natural resource management. New York: Chapman & Hall. 378 p.

Stillings, N.A.; Feinstein, M.H.; Garfield, J.L. [and others]. 1991. Cognitive science: an introduction. Cambridge, MA: MIT Press. 533 p.

Stone, N.D.; Coulson, R.N.; Frisbie, R.E.; Loh, D.K. 1986. Expert systems in entomology: three approaches to problem solving. Bulletin of the Entomological Society of America. 32: 161-66.

Waterman, D.A. 1986. A guide to expert systems. Reading, MA: Addison-Wesley Publishers. 419 p.

Zadeh, L.A. 1965. Fuzzy sets. Information and Control. 8: 338-353.

Reynolds, Keith M. 1999. NetWeaver for EMDS user guide (version 1.1): a knowledge base development system. Gen. Tech. Rep. PNW-GTR-471. Portland, OR: U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station. 75 p.

The guide describes use of the NetWeaver knowledge base development system. Knowledge representation in NetWeaver is based on object-oriented fuzzy-logic networks that offer several significant advantages over the more traditional rule-based representation. Compared to rule-based knowledge bases, NetWeaver knowledge bases are easier to build, test, and maintain because the underlying object-based representation makes them modular. The modularity of NetWeaver knowledge bases, in turn, allows designers to gradually evolve complex knowledge bases from simpler ones in small, simple steps. Modularity also allows interactive knowledge base debugging at any and all stages of knowledge base development, which expedites the development process. Finally, fuzzy logic provides a formal and complete calculus for knowledge representation that is less arbitrary than the confidence factor approach used in rule-based systems and much more parsimonious than bivalent rules .

Keywords: NetWeaver, knowledge base, fuzzy logic, decision support

Pacific Northwest Research Station
333 S.W. First Avenue
P.O. Box 3890
Portland, Oregon 97208-3890