NetWeaver Reference Manual

A compendium of NetWeaver and NetWeaver related terms, concepts, and functions

Bruce J. Miller, and Michael C. Saunders
Penn State University and,
The Heron Group LLC.

September 2002, updated November 2006

© 2002-2006 Bruce J. Miller and Michael C. Saunders
# Table of Contents

- NetWeaver Software License ................................................................. 6
- Registration Warning .............................................................................. 7
- Introduction ............................................................................................... 8
- The Help System ......................................................................................... 11
- Building Knowledge Bases with NetWeaver ............................................. 12
- Working with NetWeaver Objects ............................................................. 14
- Fuzzy Logic Nodes .................................................................................... 16
- Dependency Networks ............................................................................... 18
- Data Links ................................................................................................ 24
- Data Link Influence Calculations .............................................................. 31
- Evaluation Groups ..................................................................................... 33
- Network Editing ......................................................................................... 36
- Topic Referencing ..................................................................................... 38
- Documenting a Topic ................................................................................ 41
- Navigating a knowledge base ................................................................. 44
- Connecting to databases .......................................................................... 45
- Topic Evaluation ....................................................................................... 47
- Basic File Management .......................................................................... 48
- Application Main Menu .......................................................................... 49
- Application Button Bar .......................................................................... 51
- Data Link Monitor .................................................................................... 52
Data Link Editor ................................................................. 53
Crisp Argument Definition ............................................. 54
Fuzzy Argument Definition ............................................ 55
Goal Group Definition ...................................................... 56
Batch Topic Creation ......................................................... 57
Batch Documenting of Topics .......................................... 57
Batch Renaming ............................................................... 57
Topic Exporter ................................................................. 57
Dependency Overview ...................................................... 58
Automated HTML Documentation ..................................... 58
Knowledge Base Scrubber ............................................... 58
Topic Selector ................................................................. 59
Influences Viewer ............................................................ 59
Knowledge Base Database Browser .................................. 60
Knowledge Base Database Tables ................................... 61
Table Creation Wizard ..................................................... 62

NODE REFERENCE
Node reference conventions ........................................... 63
Abs - Absolute Value ....................................................... 64
Acos - Arc Cosine ............................................................ 65
Addition .............................................................................. 66
AND .................................................................................... 67
Fuzzy math in AND nodes ............................................... 69
Asin - Arc Sine ................................................................. 70
Atan - Arc Tangent ................................................................. 71
Ave - Average ........................................................................... 72
Ave-nz - Average-non zero .......................................................... 73
Calculated Data Link ................................................................. 74
Ceil - Ceiling ............................................................................. 75
Cos - Cosine .............................................................................. 76
Cosh - Hyperbolic Cosine .......................................................... 77
Crisp Argument ............................................................................ 78
Data Link Proxy ........................................................................... 80
Deg2Rad - Degrees to Radians ...................................................... 82
Division ....................................................................................... 83
e .................................................................................................... 84
Equation ....................................................................................... 85
ERROR .......................................................................................... 86
Exp - Exponential ....................................................................... 87
False ............................................................................................ 88
Fuzzy Curve ................................................................................. 89
Goal .............................................................................................. 90
Goal Proxy .................................................................................. 91
Goal Group .................................................................................. 92
Goal Group Proxy ....................................................................... 95
If ................................................................................................. 96
k - Constant ................................................................................ 100
Ln - Natural Logarithm ............................................................... 101
This is a legal agreement between you, the end user, and Rules of Thumb, inc.

By acquiring and using this software, you agree to become legally bound by the following terms and conditions. PLEASE READ THEM CAREFULLY.

DO NOT INSTALL THIS SOFTWARE IF YOU DO NOT AGREE TO THEM. INSTALLING THIS SOFTWARE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS.

COPYRIGHT NOTICE - Copyright ©2002 The Pennsylvania State University and Rules of Thumb. All rights reserved. This publication may not, in whole or in part, be reproduced, transmitted, transcribed or translated into any language or computer language in any form or by any means, without the prior written permission of the publisher, Rules of Thumb.

LICENSE - In return for your acceptance of the product, signified by 1) Registration of the product or 2) Retaining the product for more than thirty (30) days after delivery, Rules of Thumb grants a non-exclusive license for a single user of the product. You may use one copy of the software on a single computer. You may not sublicense, assign or transfer the license of the program. Any attempt to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

RESTRICTIONS - Compiled code of the computer program(s) described herein may not be copied, transmitted or transcribed from the accompanying media (if any), except by the original purchaser for his own convenience. The original and any back-up copies of the licensed software and its documentation shall be used ONLY ON ONE COMPUTER AT A TIME. When used on a multi-user or network system, you must purchase a separate copy or run-time license for each user or station on the system that will use the software.

WARRANTY - The authors have made all reasonable attempts to prepare this publication and its accompanying programs in an error free manner. Rules of Thumb warrants that the software contained in this package will work substantially in agreement with the documentation provided, for a period of ninety (90) days from the date of receipt. Rules of Thumb does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free. You, as the user, accept full responsibility of determining the suitability of this software for a particular application. Rules of Thumb further warrants that the media (if provided) on which the software is supplied are free from defects in materials or workmanship under normal use and service for a period of ninety (90) days after purchase. Rules of Thumb will replace defective media within this time period if you return the defective media along with a copy of your purchase invoice. If you are dissatisfied with the product for any reason, you may return this product in resalable condition within thirty (30) days of the purchase date, for a full refund of the purchase price (not including shipping charges). A copy of your invoice must be included along with a signed statement attesting to the fact that you have destroyed all copies of the software that you may have made either in the process of testing this product or otherwise. Rules of Thumb's entire liability shall be, at Rules of Thumb's option, either (a) return the price paid, or (b) repair or replacement of the product that does not meet this limited warranty and which is returned to Rules of Thumb with a copy of your receipt.

DISCLAIMER - The above are the only warranties of any kind either expressed or implied, including, but not limited to the implied warranties of merchantability and fitness for a particular purpose, that are made by Rules of Thumb, on this product. Neither Rules of Thumb, nor any of its employees, agents or authors involved in the production of this product shall be liable for any direct, indirect, consequential, or incidental damages (including damages from loss of business profits, business interruption, loss of business information and the like) arising out of the use or the inability to use this product. No oral or written information or advice given by Rules of Thumb, its authors, employees, agents, dealers or distributors shall
create a warranty or increase the scope of this warranty. Your exclusive remedy hereunder shall be return of the purchase price, or replacement of faulty or damaged diskette(s) at Rules of Thumb's sole option.

GENERAL - You acknowledge that you have read this agreement, understand it and agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement of the agreement between you and Rules of Thumb which supersedes any proposal or prior agreement, oral or written, and any other communication between us relating to the subject matter of this agreement.

Registration Warning

The Registration Warning randomly appears if NetWeaver has not been registered. It gives directions for obtaining a registration key. To continue the user must either register the software or at least acknowledge agreement that the software is being used only for evaluation purposes.
Introduction

NetWeaver is a knowledge base development system created for the Microsoft Windows platforms that provides a graphical environment in which to construct and evaluate knowledge bases. The software provides graphic tools for constructing executable dependency networks that permit both forward and backward chaining reasoning and renders knowledge dependency networks with a fully executable graphic representation so that networks appear just like they would on a white board. Because the inference engine is integral, networks can be evaluated in real-time with nodes changing color to indicate their changing "trueness" levels. This ability to peer into the logical workings of a knowledge network greatly optimizes the knowledge engineering process by (1) providing the ability to run and evaluate freshly elicited knowledge in the presence of the domain expert, (2) enabling the knowledge engineer to trace the logic structure from data to conclusions and (3) allowing the knowledge engineer to quickly identify and edit errors and inconsistencies in the logic.

Knowledge base systems come in a variety of forms, but the dominant type currently in use are rule-based systems. Knowledge representation in NetWeaver, in contrast, is based on object-oriented fuzzy-logic networks which 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 very 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

The NetWeaver inference engine was developed at Penn State University by Michael C. Saunders and Bruce J. Miller. Initial development of the system began in 1987, but the system has been steadily evolving ever since that time. It was originally developed for the Macintosh (tm Apple Computer) but has since been ported to Windows and Unix. It is written in highly portable C++. The Windows NetWeaver development system (the visual interface) was developed at Rules of Thumb, Inc. to provide efficient interactive access to the NetWeaver inference engine on the Windows platforms. It is written in Object Pascal.

What is a knowledge base?

In recent times, the phrase "knowledge base" has come into popular usage and is now generally taken to mean a body of knowledge about some problem domain. As originally used, however, the term meant something much more precise. In particular, in the original sense in which the term was conceived, a knowledge base is a formal logical representation of the entities of interest in some problem domain and their relations to one another. That is, a knowledge base is a body of knowledge that has been organized within a formal syntactic and semantic framework allowing formal inferencing about the problem at hand. We use the term, knowledge base (or its abbreviation: kb), throughout the NetWeaver documentation in this more formal sense.

Components of a knowledge base system

A knowledge-based system consists of a knowledge base, inference engine and user interface. The knowledge base is used by the inference engine for data interpretation. The inference engine represents a program library that interprets external data according to the semantics built in the knowledge base by the
knowledge engineer. The user interface is an intermediary program between the user and the inference engine. With the help of the interface the user constructs knowledge bases and directs the data processing by the inference engine.

During construction of a knowledge base, for example, the inference engine constructs the knowledge base in response to mouse and keyboard operations directed by the user in 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 the conduct of a knowledge-base evaluation of data.

**Fuzzy Logic**

Although the term, fuzzy logic, has a distinctly esoteric ring to it, the concept is actually fairly simple. Fuzzy logic provides a metric for expressing the degree to which an observation on 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, let's consider a simple example.

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 to be a person that 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, 11 months, and 30 days old, they are not an adult, but if the person is one or more days older they are an adult. This characterization of adulthood is sufficient if the concept of adult is limited to a simplistic legal one. However, if by adulthood we instead are really interested in expressing something more complex such as an individual’s emotional maturity, then the simple bivalent rule for determining adulthood is no longer adequate. Most people would agree that a 5-year-old has no, or best at minimal, adult qualities. In a 13-year-old, however, we might begin to see at least some early signs of adult characteristics. Many 18-year-olds demonstrate many adult qualities (they act very "grown up"). Conversing, most people can think of at least a few 25-year-olds they have met in their life who could not be called particularly emotionally mature. So, as a first step toward improving the characterization of adulthood, one might construct a simple (fuzzy) curve that translates age into degree of membership in the set "adult".

![Fuzzy Logic Graph](image)

We indicated that fuzzy logic allowed a more parsimonious knowledge representation than that which is possible with rule-based systems. The reason is simple. A single fuzzy curve is sufficient to express the full spectrum of adulthood. In contrast, rule-based systems are inherently bivalent, meaning that a rule is either true or false. To more precisely characterize adulthood 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 our 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 our rule base could require as many as 25 additional rules. In contrast, in a fuzzy logic-based representation of this more complex situation, we only need one more fuzzy curve and perhaps a new network object to jointly evaluate the two fuzzy curves. So, in our 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 explodes approximately exponentially, whereas the number of fuzzy curves and related objects needed to describe the same problem in an object-oriented fuzzy logic representation only increases approximately linearly.
The Help System

NetWeaver has several layers of online help.

Hints

Hints are displayed in the lower left corner of the main window. The hints are updated automatically as the cursor moves over different controls or objects. This feature can be enhanced by toggling on the Fly-over Hints menu item under the Help Menu. When it is on, the hints "pop-up" in a little window near the object the hint is describing as the cursor pauses over it.

Cue Cards

Context sensitive help can be turned on by selecting the Cue Cards menu item from the Help menu. Cue Cards are a part of NetWeaver's hyper text based help system. The cue cards update as you navigate through NetWeaver so that the help topic showing is the one related to the topmost window.

Navigating the Help System

NetWeaver has an extensive hyper text based help system that can be navigated in a number of different ways. The help window has two main parts: a navigator window and a topic window.

The help navigator window can be hidden or made visible by clicking the tabs button at the top left of the help topic window. The help navigator window has three tabs:

- **Contents.** Select the Contents menu item from the Help menu or, if the help system is already open, you can click on the "Contents" tab in the help navigator window. The help system will open with an outline of the help system contents. Clicking on a topic in the outline will open that topic in the help window.
- **Index.** Select the Index menu item from the Help menu or use the "Index" tab in the help navigator. Either scroll to the index selection or start typing the topic of interest in the edit box to make the list scroll automatically.
- **Search.** Select the Search menu item from the Help menu or use the "Search" tab in the help navigator. Enter the text to find in the edit box and click on the "Find" button. This a free form text search, so the system will look for the entered text, in whole or part, in all of the help files. Once a match is made the help window will open to the occurrence. Repeated pressing of the "Find" button (now the "Find Again" button) will seek the next occurrence(s) until all files have been searched.

The topic window displays the hyper text based help topic. The tabs button toggles the help navigator window between visible and not. The rewind button re-opens the initial help topic. The previous and next buttons move one topic forward or back in the history list of viewed help topics. The print button sends the current help topic to the printer.
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 simply be easier to solve with logic. The second case arises very frequently because, particularly when dealing with ecosystems, there is an almost unlimited number of relations potentially of 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. However, there is often a wealth of human experience in these same institutions that can be drawn upon to develop useful, more qualitative, knowledge based models to guide decision making.

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 engine used provides partial evaluations of system states and processes based on available information, and provides useful information about the influence of missing data which can be used to improve an assessment.

NetWeaver Knowledge Bases

Uses of dependency networks include:

- Evaluate the truth value of assertions about system states and processes given existing data.
- Identify data requirements for an analysis.
- 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 very wide variety of topics. This is particularly valuable in the context of ecological assessments 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 inter-relations that can be represented in a knowledge base is only limited by a computer’s dynamic memory.

Evolving Knowledge Bases

As we discussed in the overview, NetWeaver is a rigorously object-based knowledge base development system. One of the more practical implications of object-based knowledge bases is that it is very easy to start with simple knowledge representations and gradually evolve them into large complex systems 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:
"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" (Gall, J. 1986. Systematics: how systems really work and how they fail. The General Systematics Press, Ann Arbor, MI).

NetWeaver is designed to build a knowledge base in an incremental and evolutionary fashion. The best possible advices that we can give to the novice user is to avoid designing large complex systems at the outset, and instead, start with a simple knowledge base built from a small number of dependency networks, and gradually evolve this simple representation into a more complex representation of the problem.

It is almost always 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 sections Creating network objects and Documenting network objects). 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 can easily be added later in development without upsetting overall knowledge base structure.

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

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

As a knowledge base structure evolves, you will probably find occasion to use existing antecedents (both dependency networks and data links). Multiple occurrences of dependency networks and data links in a single knowledge base, are not a problem because NetWeaver objects are reusable. In fact, the presence of a dependency network, in particular, in two or more other networks within the same knowledge base is an important mechanism by which networks are interrelated.
Working with NetWeaver Objects

Introduction to NetWeaver objects

Here we briefly summarize the collection of objects that make up a NetWeaver knowledge base. There are two basic categories: user- and pre-defined 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 an object that 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 either may evaluate a datum against a simple or fuzzy argument, or they may 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 and performs some transformation on the data.

Pre-defined NetWeaver objects

Logic nodes such as AND and OR are objects that are 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 that are used to create mathematical expressions for use in calculated data links and comparison nodes.

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

Creating objects

Evaluation groups can only be created from the Application button bar. Data links and networks either may be created from the Application button bar or they may be created from 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 itself is created. In the second case, a reference to the object is also created and attached to the node that is 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 the appropriate menu item on the Topic menu of the Application menu bar.

**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. When an object is created in this way, you are effectively creating an object itself 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 network object). After any pertinent documentation is entered in the Documentation window and the window is closed, NetWeaver automatically displays another window in which to provide specifications for the new object.
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 no data links antecedent to a network use fuzzy arguments, then the operation of these nodes conforms quite closely to their usage in conventional logic. The only real difference in this context between these nodes, as used in NetWeaver, and standard logic is that true = 1, 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. In order 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 a 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:

\[ \text{AND}(t) = \text{min}(t) + (\text{average}(t) - \text{min}(t)) \times (\text{min}(t)+1)/2 \]

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 might later be found to be completely false in which case the fuzzy AND should evaluate to completely false, regardless of with 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 that 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 data link rather than constructing another argument. For example, if a data link for percent slope has already 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 1):

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.

1. XOR evaluates to -1 (completely false) if all immediate antecedents are completely true or completely false.

2. 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 fully true.

<table>
<thead>
<tr>
<th>Data link truth values</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1.0</td>
<td>-1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>1.0</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

**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 node (reading from left to right) whose absolute truth value is at least equal to its weight. Since weight defaults to 1, the SOR typically uses the first topic or node that is fully true or fully false. If no immediate antecedent of a SOR node satisfies this condition, then SOR remains undetermined.
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 a system’s states and process, relevant to the description of the network being modeled with the network structure. A single dependency network is hierarchical in structure. However, NetWeaver knowledge bases generally are networks of such hierarchies. In general, a network is composed of other networks whose logical relations to the parent network are defined by relational nodes.

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. NetWeaver knowledge bases rely on propositional logic and every NetWeaver network supports or rejects 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.

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. The truth value is undetermined in the following cases:

1. Some of the data needed to fully evaluate the node or dependency network has not yet been supplied when an evaluation of a network is being performed.
2. The data are missing and cannot be supplied.
3. One or more of the 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 a network that another network depends upon for its truth value. The term antecedent (the adjective is 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 in order 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 upon 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**

![Network Window Image]

**Usage**

Use this window to define, edit, or simply view the logical dependencies of a specific dependency network or calculated data link. The structure of network dependencies that are built 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. If you want to defer defining the logical structure of a network, just close the Network window.

One of NetWeaver’s very powerful features is that many windows are dynamically linked to one another. When an evaluation group or network is being evaluated in the Evaluation window, for example, the states of network objects displayed in any Network windows that are open are dynamically updated concurrent with the evaluation. Object references are color coded as bright red if the truth value of the referenced object is completely false, bright green if the truth value is completely true, and take on 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 a network that already exists. To open the Network window from the Knowledge base window:

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

1. 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 the later section, Fuzzy logic nodes), 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.

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

2. 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. 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, 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 on use of the Data link reference window, see the later chapter, Data links.

Calculated data links are similar to networks in that a Network window is used to graphically construct its functional structure. Later, we refer to this particular Network window 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 only enabled 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:

- **Object Spy** - turns the object spy on and off. The object spy is a popup window that displays internal information about the node the cursor is over. This is a very handy device during knowledge engineering sessions to view the inner workings of a dependency network.
- **Zoom In** - changes the scale of the network to make it appear larger.
- **Zoom Out** - changes the scale of the network to make it appear smaller.
- **Shift left** - shifts the selected object to the left.
- **Shift right** - shifts the selected object to the right.
- **Undo** – undo the last edit action.
- **Delete** – delete the object from the network, but does not copy it.
- **Cut** - removes the object from the network and copies it for subsequent Paste action (compare to Delete, previous).
- **Copy** - copies the selected object and any of its antecedents.
- **Paste** - pastes the previously copied object and any of its antecedents to the currently selected network object (compare with Paste children only, next).
- **Paste children only** - is similar to Paste (see above), except that only the antecedents (and not their parent, the originally selected node) are pasted to the selected object.
- **View Influences** - opens a document that lists the topics which contribute to the value of the network and their respective influence values.
- **Document** - opens the node documentation editor (the network reference window).
- **Capture Image** - saves an image of the network to disk.
- **Data link** - creates and attaches a new data link or attaches an existing data link to the currently selected network object. Valid objects to attach to are fuzzy logic nodes, comparison nodes, switches, fuzzy curve nodes, and mathematical operators.
- **Goal** - creates and attaches a new dependency network or attaches an existing network to the currently selected network object. Valid objects to attach to are fuzzy logic nodes, comparison nodes, switches, fuzzy curve nodes, and mathematical operators.
- **Group** - not currently implemented. But if it were... it would attach a reference to a goal group.
- **OR** - attaches a fuzzy OR node to the currently selected network object. Valid objects to attach to 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 to attach to 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 to attach to 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 to attach to 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 to attach to 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 to attach to 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 to attach to 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 to attach to 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 to attach to are function nodes, comparison nodes, switches, and fuzzy curve nodes.

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

• Go to Connections - if the network is used in only one other network it opens that network. Otherwise, a list of networks to open drops down.

Viewing logical dependencies of a network

Networks may extend many levels deep. However, the network window only displays 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 reference window

Usage

You may never need to use the network reference window, but it primarily is provided to enter a comment and explanation that is specific to the referenced network concerning its use as an antecedent in the current network window. Often, no other commentary, other than 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. We generally recommend that users stick with the weight of 1.0, which is the default weight assigned in the Documentation window when you create a network object. However, there may be specific circumstances in which it is desirable to alter the default weight.

Access
To open the Network reference window, double click, or 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.
Data Links

Data input to a knowledge base is accomplished through data links which, in important respects, are simply elementary dependency networks. As with dependency networks, the primary state variable 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 (typically, a calculated data link). Also 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 while evaluating 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. Finally, data also can be applied to data links through data base linkages.

Data Influence

Data that are input to a knowledge base influence the evaluation of networks in the sense that data determine the truth values of any dependency networks that depend on that data. In NetWeaver, influence is specifically defined to mean influence of missing data. The semantics of the formal logical model that underlies 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 interrogation mode (see Evaluating a network object). Other software applications that use the NetWeaver inference engine can also make use of information about data influence to guide analyses in various ways.

Types of Data Links

Simple data link

Either a simple data link can read a data value that it passes to another data link without evaluation, or a simple data link can compute a truth value by evaluating the data value against an argument list. A simple data link whose data value is used directly usually passes its data value to a 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 data value is tested against. If the argument is "<= 10", for example, and the data input to the data link satisfies this condition, then the truth value of the data link evaluates to 1 (completely true). Data input may also 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 hung from it that connect two or more data links or networks. The functional representation of a calculated data link is built graphically in the Calculated data link window, similar to
the way that dependencies of networks are built. Because of this, like 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 data value is tested against. If the argument is "\leq 10", for example, and the data input satisfies this condition, then the truth value of the data link evaluates to 1 (completely true). Data input may also be evaluated against a fuzzy argument.

**Data Link Window**

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

**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. Each reference to a data link could therefore (but not necessarily) use a different argument. Arguments that have been 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.

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.

**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. The selected items are removed from the Arguments list box.

Arguments that have been created for a data link can be deleted providing that 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 (e.g., 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 only be 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** of the **Data link window** 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 to select among at evaluation is optional. If no choices have 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**. Furthermore, a specification of choices in the **Choices folder** does not preclude a user from entering some other value in the **edit line** of the **Choices combination list box**.

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, let us 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 both displayed in the **list box** portion of the **Choices combination list box** in the **Influence pane** of the **Evaluation window**.

### Calculated Data Link Window

The **Calculated data link window** is a special case of the **Network window** in which buttons for mathematical operators (for operating on function node) are enabled on the **Network window button bar**.

#### 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 one of the mathematical operators from the pop-up menu to hang from the calculated data link.
3. Choose another math operator or an operand (e.g., a simple data link, another calculated data link, etc.) to hang from the math operator added at step 2.
4. Repeat steps 1-3 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 directly hung from the calculated data link itself.
2. Only a math operator can be directly hung from a calculated data link.
3. The number of operands that can be hung from a math operator depends on the operator. Refer to mathematical operators for specifications.

### 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 a 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. On selecting a specific data link, NetWeaver automatically displays the data link reference window.

Use the dialog to choose a specific argument or set of arguments from the Available arguments list box, and add them to the Arguments list box. Those arguments appearing in the Arguments list box are those used to actually evaluate the truth value of the data link. Arguments may be either simple arguments or fuzzy arguments. New arguments for the data link object can also be created in this window.

NetWeaver automatically opens the Data link reference window when a data link reference is added to the currently open Network window. To edit arguments used by a referenced data link subsequently:

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.

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 for use in evaluating the reference, select the argument in the Available arguments list box and press the Add button to add the selection to the list of arguments in the Arguments list box.

To remove an argument that is used to evaluate the referenced data link, 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, respectively, later in this chapter.

Data Link Arguments

A data link can either use data directly, in which case the data value input to the data link is propagated upward in the network, usually to a calculated data link, or the data link may compare the data value that is 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 is used directly, the data value input to the data link 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 in order 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).

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Simple argument window</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>If two expressions are specified, they are logically ANDed together. That is, the two expressions must both be true for a data value input to the simple data link in order for the truth value of the simple data link to evaluate to 1 (completely true).</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td></td>
</tr>
<tr>
<td>==</td>
<td>Equal to</td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td></td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td></td>
</tr>
<tr>
<td>is in</td>
<td>Data is a substring of the string argument. The data link evaluates to true if the argument is &quot;treehouse&quot; and the data input is &quot;house&quot;.</td>
<td></td>
</tr>
<tr>
<td>contains</td>
<td>Data includes the argument as a subset. The data link evaluates to true if the argument is &quot;house&quot; and the data input is &quot;treehouse&quot;.</td>
<td>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.</td>
</tr>
</tbody>
</table>

The Simple argument window 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.

Fuzzy argument

A fuzzy argument defines a fuzzy surface function with up to four points that determines a data value’s degree of membership in a fuzzy set. If a data link has a fuzzy argument as opposed to the simple argument discussed in the previous section, then the data value read by the data link is compared to the fuzzy surface function to compute the data value’s membership in the fuzzy set defined by the fuzzy surface.

Fuzzy argument window

To specify a fuzzy argument function surface for a data link in the Fuzzy argument window:
1. Enter a descriptive name in the Descriptor edit box for the fuzzy surface that the points will define.

1. In the Operator 1 group box, enter a value for x 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.

2. Repeat the procedure in step 2 for the Operator 2 group box to define a second point on the fuzzy surface.

3. 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 surface. In the example, 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 function surfaces that offers some advantages over the use of a fuzzy argument as implemented in a simple or calculated data link. In particular, the fuzzy curve is not limited to four points. Also, a fuzzy curve allows the function surface to be defined dynamically; that is, by data input at the time of network evaluation.

The fuzzy argument dialog window 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**

Because data links are really just elementary networks, they can be evaluated. However, the Data link evaluation window is much simpler than the Evaluation window for networks and evaluation groups. The top Comment pane of the data link evaluation window displays the comment associated with the data link, if any (see the earlier section, Documentation window). The bottom 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.

   1. Press either the Replace (default) or Append radio button.
   2. Press the Accept button to evaluate the data link.

To open the Data link evaluation window from the Knowledge base window:

1. Click on a data link in either the Cals or Data folder to highlight it.

1. 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 that is 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.
Data Link Influence Calculations

(Please note that the following description, while being algorithmically correct, does not necessarily represent the actual order of execution and computational efficiencies employed in NetWeaver. Rather it is described in this fashion to facilitate understanding of the principles used.)

Target node is the goal (dependency network) or goal group (group of dependency networks) of interest of which we are testing the data link against to determine the data link's influence upon the target node.

In practice all nodes are reset so that they are all dirty and then the target node is evaluated. In the process of evaluation, evaluate messages are sent down through the networks undirtiring nodes along the way. Only nodes that are being used are undirtied. Others that have not been reached stay dirty.

For each place a given data link is used, a get impact message is sent and the results are summed for the data link. Basically, each path (branch) between a data link and target node is evaluated for its impact and then the impacts for each branch are added together to give a composite impact for the data link.

Each branch can have an impact of 0-1. Each node along the path has an impact factor of 0-1 based on the node type and current value with a number closer to 1 representing more impact and closer to zero representing less impact. The branch's impact is then the product of the impact factors of all the nodes along the path.

The node impact factor is a product of two other factors. The first factor deals with the local topology. The topology factor is calculated as the node's fraction of weight of the total sibling weight. In other words, the topology factor is the node's weight divided by the sum of the weights of all the nodes directly connected to the node above it. The topology factor is one if the node is the only child of its parent (it is the only node hanging from the one above) and something less than one otherwise. The more competition, the less the topology factor.

The second impact factor deals with the node's capability to be changed (influenced) by the data link (actually the data link's branch to the node). For example, an already true OR node cannot be changed by considering additional information. The same is true for an already false AND node. The factor then is a function of the current value and dirtiness of the node.

A node that is dirty has an influencability factor of zero. Being dirty indicates that the node is not being used in the current evaluation, in effect stating that this branch has no influence. Forced and Ignored nodes are treated similarly (they also have a factor of zero influence).

For non-dirty nodes, we are currently using a simple linear function that is one when the current node value indicates it can be maximally changed (a value of false for an OR node and true for an AND node) and goes to zero as the node's value indicates it cannot be changed (a value of true for an OR node and false for an AND node). Currently, only OR and AND nodes have this sort of influencability factor, the others are given a factor of one as a default. We probably will want to change this as we figure out how.

In summary:

\[
\text{Data Link Influence} = \text{Branch}_1 \text{ Inf} + \text{Branch}_2 \text{ Inf} + \ldots + \text{Branch}_n \text{ Inf}
\]

\[
\text{Branch}_i \text{ Influence} = \text{Node}_1 \text{ Inf} \times \text{Node}_2 \text{ Inf} \times \ldots \times \text{Node}_n \text{ Inf}
\]
Node_i Influence = Local_Topology_Factor * Local_Influencability_Factor

Local_Topology_Factor = Node_Weight/Total_Cohort_Weight

Local_Influencability_Factor = f(dirtiness, trueness_level)

Because along a branch all factors are 0-1 and are multiplied, the resultant influence for a branch can range only 0-1. Therefore, the influence of a data link can range 0-number_of_branchs.

Influence is dependent on current states and topology. Current states change with data applied. "More" topology tends toward smaller influences.
Evaluation Groups

Multiple networks may be defined as an evaluation group to simultaneously view the evaluation of several dependency 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 that represent a variety of components of the overall problem being modeled. The set of dependency networks that comprise a single knowledge base are conceptually related at least to the extent that they represent different aspects of the same problem, but some of these dependency networks may represent 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 either may be discrete, or may 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.

Group editor

Use the Group Editor 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.

Evaluation window

As its name suggests, the Evaluation window is used to conduct an evaluation of an evaluation group or an individual network.
The upper pane of the Evaluation window displays the network or networks being evaluated and the result of their evaluation. A bar graph for each network displays the truth value for that network 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 windows, the content of these windows is dynamically updated during an evaluation. By opening Network windows during an evaluation, it is thus 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. 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 of data input that is expected for data links.

Influence pane

The bottom pane of the Evaluation window is the Influence pane. This pane contains three list boxes.

Influence list box

The leftmost list box of the Influence pane displays data links or networks that have influence on evaluation of the networks by virtue of not yet having been evaluated. In manual evaluation mode (see the later section, Manual evaluation versus step mode), the Influence list box is labeled "Influential antecedents," and contains a list of all data links and networks that influence the networks displayed in the Evaluation pane and that have not been evaluated. In step mode, the Influence list box is labeled "Influential data links," because only data links are displayed in this 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 is evaluated, any network or data link evaluates to fully true or fully false, that network or data link is highlighted in bold in this list box.

Choices combination list box

The rightmost list box of the Influence pane is the Choices combination list box which is composed of both an edit box and a list box. There also are buttons (see the following section, 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 provide a variety of options for controlling how antecedent data links and networks are used in an evaluation.

- **More Info** - if the choices are linked to hyper-text info a selection window will open.
- **Accept** - accept 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.
- **Clear** - clears any data already applied to the selected data link.
- **Ignore** - ignores the data link or network that is currently selected in the Influence list box or Other antecedents list box in evaluation of the network. This is a powerful option that should be used with caution. Choosing to ignore data links or networks amounts to deleting the item from the network for the purposes of the current evaluation (they are not deleted from the knowledge base, however). On the other hand, there may well be occasions when such 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 being run in interrogation mode, NetWeaver queries the user to input values to data links, with the order of query determined by data link influence (most influential first). Pressing the Skip button (which is only functional in interrogation mode), informs NetWeaver that you want to defer providing a data value for the data link until later in the evaluation. If a data link is skipped, it is displayed in the Influence list box with a strike-through. A skipped item can be manually highlighted in the Influence list box at any later time during an evaluation by clicking on the item.

**Manual evaluation versus interrogation**

Evaluation of a network or evaluation group can be performed in one of two modes:

1. Data links and networks can be manually selected 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

1. In interrogation mode, NetWeaver will prompt the user to enter a value for the currently most influential data link.

To put NetWeaver in interrogation mode, press **Interrogation** in the lower right corner of the evaluation window. To halt interrogation mode at any time and revert to manual mode, select the same button now called "Stop Interrogation". In general, an evaluation can be freely switched between manual and interrogation mode. Moreover, while in interrogation 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 reverts back to interrogation mode automatically.
Network Editing

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 only be deleted from the knowledge base from the Knowledge base window. Deleting the graphic representation of an object from a Network window (see Deleting references, later) 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.

Furthermore, to avoid creating havoc in a knowledge base, NetWeaver does not allow an object to be deleted from the knowledge base if the object is referenced in one or more dependency networks. In order to remove an object from the knowledge base, all references to it must first 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.

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

<table>
<thead>
<tr>
<th>Object</th>
<th>Editable attributes</th>
<th>Applicable window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation group</td>
<td>Networks in group</td>
<td>Evaluation group window</td>
</tr>
<tr>
<td>Dependency network</td>
<td>Dependency hierarchy</td>
<td>Network window</td>
</tr>
<tr>
<td>Data link</td>
<td>Available arguments</td>
<td>Data link window</td>
</tr>
<tr>
<td></td>
<td>Simple argument definition</td>
<td>Simple argument window</td>
</tr>
<tr>
<td></td>
<td>Fuzzy argument definition</td>
<td>Fuzzy argument window</td>
</tr>
<tr>
<td>Calculated data link</td>
<td>Functional hierarchy</td>
<td>Calculated data link window</td>
</tr>
</tbody>
</table>
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 we call the *Calculated data link window* to distinguish it from the standard *Network window* only for purposes of clarity in later discussion). 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.
**Topic Referencing**

Data links and dependency networks are re-usable 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. A reference to the object 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*) 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 must first 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*. The attributes displayed in this window are specific to the particular reference and are distinct from the attributes of the same name that may have been entered for the network object itself 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 our 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 also has its own *Comment, Explanation, and Weight attributes* that can be edited in the *Data link reference window*. These attributes are specific to the particular reference and are distinct from the 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 that are specific to a specific data link reference is the same as that for network references (see the 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. Similar to a network object, a data link object has global scope in the knowledge base. However, specific references to a data link can, and commonly do, use different arguments. It is, therefore, quite possible for a data link reference to evaluate to true in one instance in which it is used, and false in other instances 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 evaluated once for each instance in which the data link is referenced.
Documenting a Topic

NetWeaver displays the Documentation window as soon as a menu item is chosen or a button is pressed to create a new topic. Other than entering a name for the new object in the name field of the Documentation window, documentation is strictly optional but also extremely valuable. Documenting a network can be streamlined somewhat 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

Documentation attributes of an object

Name

When naming a dependency network or data link, it is important to keep in mind that their basic function is to test a proposition about the topic of interest (data links that use data directly are an exception). The name given to a dependency network should therefore be formulated so as to state a proposition concerning the topic that is being represented by the network. If, for example, the topic was concerned with elk habitat, the network name might be "adequate elk habitat."
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. NetWeaver will auto-generate a suitable alias when you click on the "Create Alias" button. For a column name to be dBase compliant it has to meet the following requirements:

- a maximum of 10 characters
- no spaces
- all uppercase
- first character is alphabetic
- alphanumeric or underscore for other characters

**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 (|) (e.g., mybmp|yourbmp|ourbmp).

**Comment**

The comment provides a brief description of the 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 for data links during network evaluation as a guide to the type of input that is expected.

NetWeaver also lets you set values for networks when a network is evaluated, so also it is a good idea to comment a dependency network.

**Explanation**

The explanation field is for providing more extensive documentation on an object. Here, one might explain the rationale for why this object is represented in the network, and perhaps something about the structure of underlying dependencies.

**Domain source**

This field documents who developed this portion of network structure. As an example of streamlining documentation, if three people have designed a large network hierarchy together 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 what those assumptions are. 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 by the user.

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, we recommend use of the default weight. The one exception to this advice is in the context of antecedents to a fuzzy SOR node.
Navigating a knowledge base

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

- The Outline tab 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 tab by clicking on the icons in front of the names to expand or collapse the outline branch.
- 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 then 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 Open from the pop-up menu.
Connecting to databases

NetWeaver can connect to databases through either the knowledge base database browser within NetWeaver or through GeoNetWeaver. The database browser links to a set of database tables that: coordinate naming conversions between the knowledge base and the database, store input data, and store output results and statistics. The tables (for now) must be in dBase format (*.dbf). GeoNetWeaver links to data through ESRI ArcView shapefiles (*.shp). This discussion will deal only with the database browser built into NetWeaver.

NetWeaver supports multiple concurrent instances of database. Each linkage is defined by a set of database tables. Each table has a unique role in the database link. Not all are required. For insight on the structure of the tables use the “Create New Tables” wizard available in the database browser window to create a sample or working set of tables.

The **Alias** table is used to coordinate the columns in the other tables (except the stats table) with the topics in the knowledge base. It is a list of topic names and the names of the table columns which contain their data. If the topic names or aliases already match the column names then this table is not needed.

The **Input** table contains the data to be applied to the data links. It is required. It can contain extra columns. If the same column names exist in the output table their contents will be transferred when the tables are processed.

The **Flag** table is used to indicate which data links are to be ignored, row by row. This table should have a 1:1 correspondence to the input table. Blanks indicate NOT to ignore the data link whereas an "I" in the cell indicates the data link should be ignored for the row. This table is generally used only in special circumstances where a management decision has been made to disregard portions of the knowledge base. A better alternative is to use either the Persistent Ignore or Conditional Ignore attributes in the knowledge base. This table is not required.

The **Output** table stores the results from processing. It is required only if you are going to process the database. Data can be loaded row by row for review without it.

The **Influences** table stores the row by row influences of missing data. It is not required.

The **Ranks** table stores the row by row influence rank of missing data. It is not required.

The **Stats** table stores a summary of influence statistics. It is not required.
### Table Summary:

<table>
<thead>
<tr>
<th></th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dbf Table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aliases</td>
<td>optional</td>
<td>Coordinates columns in other tables with topic names. In most tables (all except aliases and stats) each column represents a topic. To be associated with a topic the column name must match a topic name, topic alias, or an entry in this table.</td>
</tr>
<tr>
<td>Inputs</td>
<td>yes</td>
<td>Contains input data.</td>
</tr>
<tr>
<td>Flags</td>
<td>optional</td>
<td>Used to flag topics to be ignored. Use with caution!</td>
</tr>
<tr>
<td>Outputs</td>
<td>yes*</td>
<td>Stores the results (topic values) after processing. * if processing</td>
</tr>
<tr>
<td>Influences</td>
<td>optional</td>
<td>Stores the influences of missing data.</td>
</tr>
<tr>
<td>Ranks</td>
<td>optional</td>
<td>Stores the influence ranks of missing data.</td>
</tr>
<tr>
<td>Stats</td>
<td>optional</td>
<td>Stores overall influence rank statistics of missing data.</td>
</tr>
</tbody>
</table>
**Topic Evaluation**

**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*. 
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. Either action opens a new Knowledge base window.

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.

Save a knowledge base

To save the current 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 click the "go-away" box in the upper right corner of the knowledge base window. 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.
File

New - Creates a new knowledge base.
Open - Opens an existing knowledge base.
Merge - Merges an existing knowledge base into the current one.
Get Info - Opens the information dialog for the knowledge base.
Close - Closes the topmost window.
Save - Saves the current knowledge base to disk. Any previous version with the same name will be overwritten.
SaveAs - Saves the current knowledge base to disk, but prompts for a new file name. Use this option when you don't want to overwrite the previous version.
Revert to Saved - Discards knowledge base changes and reloads from file. Use this option to "rewind" to the last saved version of the knowledge base.
Document - Documents the knowledge base in hyper text markup language. Use this option to automatically build a "web site" of the knowledge base which documents all topics, creates images of dependency networks, and hyper-links each topics to its antecedent and dependent topics.
Print - Prints the contents of the topmost window.
Exit - Exits NetWeaver

Edit

Revert - Undo the last edit.
Cut - Copy and remove the selection.
Copy - Copy the selection.
Paste - Paste the last copied selection.
Delete - Remove the selection.
Preferences - Opens the preferences dialog for customizing the configuration of NetWeaver.

Data

Clear All - Removes all data currently associated with any data links in the knowledge base.
Link to Db Tables - Opens a window where database tables can be associated with the knowledge base for the purpose of batch processing data and analyzing the results.
Create Db Tables - Creates empty database tables to be associated with the knowledge base. Any required aliases for topics will be created automatically so that the table column names can be linked to topics.

Topics

New Note - Create a new note topic. Note topics are used to document general knowledge base information that does not need to be linked in a dependency network such as authorship, revision information, or FAQs (frequently asked questions).
New Group - Create a new goal group topic.
New Goal - Create a new goal topic (a dependency network).

New Calculated Data Link - Create a new calculated data link (an equation network).

New Simple Data Link - Create a new simple data link.

Batch Create - Opens utility for creating multiple topics at one time.

Batch Rename - Opens utility for editing the names of all topics at one time in one place.

Batch Document - Opens utility for editing the documentation attributes for all topics in tabular form.

Scrub Knowledge Base - Opens a wizard for purging the knowledge base of used topics and for general housecleaning.

Dependency Overview - Opens a graphical dependency browser.

Export Topics - Opens utility for extracting the documentation from topics into a text file.

Find Topic - Open the find utility to locate references to topics within a knowledge base.

---

Window

Arrange Icons - "Cleans up" any icons for minimized by rearranging them along the bottom of the screen.

Minimize All - Minimizes all open windows and creates icons for them along the bottom of the screen.

Window List - A dynamic list of currently open windows. Selecting a window from this list brings the window to the front, un-minimizing it if necessary.

---

Help

Contents - Opens the help system to the table of contents outline.

Index - Opens the help system to its index.

Search - Opens the help system in find mode.

Cue Cards - Toggles the automatic context sensitive mode of the help system. If this menu item is checked, the help system will display information for the currently active window and change contents as you change context by moving from one window type to another.

Fly-over Hints - Toggles the fly-over hints. If this menu item is checked, the hint for screen objects will pop up as the cursor dawdles over the screen object.

Contact Us - Opens a utility to send comments to the developers over the Internet.

About NetWeaver - Opens the NetWeaver version info window.
Application Button Bar

New. Create a new knowledge base.
Open. Open an existing knowledge base.
Save. Save the current knowledge base to disk. If more than one knowledge base is open, then the one with the topmost window will be saved.

Wizard. Open the knowledge base starter wizard tool.
Get Info. Open the knowledge base info window.
Find. Open the find utility to locate references to topics within a knowledge base.
Clear All Data. Removes any data associated with the topmost knowledge base.

Create a new note topic.
Create a new goal group topic.
Create a new goal topic (a dependency network).
Create a new calculated data link (an equation network).
Create a new simple data link.
Create a new database link.

Help. Open the help system.
The data link monitor is a window which can be used to assign data, clear data, and toggling the ignore flag for the data link.

The top of the window displays the data link's comment.

New data values are applied to the data link by typing in the drop down box or by selecting a choice from its drop down menu in the "Choices" group. If "Replace" is selected any existing data is erased before the new data is applied. If "Append" is selected the new data or choice is added to the list of data already in the data link. The checkmark button applies the new value. The revert button cancels the new value. The question mark button launches the hyper link for the choice.

Currently applied values are displayed in the "Values" list box. Pressing the "Clear" button removes all data from the data link. Clicking the "Ignore" check box toggles the ignore flag for the data link.
Data Link Editor

Use the data link editor to create and edit arguments and to manage the data link's choice list.

To edit an argument double click on it or select it and press the "Edit" button. An argument can only be removed if it is not referenced in a dependency network. The "Remove" button will enable if the selected argument is removable. New arguments can be created by pressing the "New Arg..." or "New Fuzzy Arg..." buttons. Pressing the "Args from Choices" button uses the choice list to build similar arguments.

Choices can be edited by typing in the list box. Pressing the Choices from Args" button parses all the arguments and creates choices from values within the arguments.

Defaults are values that are used when no others have been applied.
Crisp Argument Definition

The crisp argument definition window builds and edits crisp arguments. The argument converts its datalink’s value into a Boolean result when it is used within a dependency network. The argument can have one or two operations. If the argument has two operations, they are “anded” together. Each operation consists of an operand and an operator. Operations are set up by clicking on the desired operator and typing in an operand.

The following window is read as “the result is true when the data link is greater than zero AND less than ten.”
The purpose of Fuzzy Argument Definition tool is to build a fuzzy curve to compare to the data in its data link and give a fuzzy result ranging from false to true. Up to four points can be assigned to define the curve. Each point has two values: the "x" value is typed into the edit box, the "y" value is "true", "undetermined", or "false", each represented by a radio button.

The editor for each fuzzy curve point is activated by clicking on a radio button for that point. The point can be removed by clicking on its "eraser" button.

The descriptor is a name for the argument.
Goal Group Definition

There are two columns of goals topics: The one on the left is a list of goals currently assigned to the group while the list on the right is a list of all other goal topics in the knowledge base. To move a goal from one list to the other either double click on it or press the "Add" or "Remove" buttons. Pressing the "New Goal" button creates a new goal topic and automatically adds it to the goal group.

To assign a default goal to the group select one of the goals in the group and press the "Make Default" button.

Group processing is dependent on whether the goal topics are discrete (they do not necessarily influence each other) or they represent codependent positions along a continuum of possible values. Use "Discrete" unless the goals are used to categorize the levels of one outcome (i.e., high, medium, low).

More...
**Batch Topic Creation**

The Batch Topic Creation tool lets you create multiple new topics in one process. Type or paste in the names of the topics to be created in the list box. Optionally a text file can be loaded by pressing the "Load List..." button. Select a topic type (only one type of topic can be created at a time). Press the "Process" button to create the new topics. The names are validated against names already in use before they are created. If a name is already in use, a node will not be created and a message will be issued.

**Batch Documenting of Topics**

The Batch Document Topic Form presents a list of all topics in the knowledge base and their documenting attributes in tabular form. Make changes by typing (pasting into) the appropriate cell. Changes are saved into the knowledge base as they are made. Topic names and aliases cannot be edited here, instead use the Batch Topic Name Editor.

**Batch Renaming**

The Batch Topic Renaming Form is a list of all knowledge base topics and their aliases. Edit the names and aliases by typing in the appropriate cell. Changes are not saved until the "Commit" button for the topic is pressed. Changes are validated to avoid name collision before they are saved into the knowledge base. Pressing the "Revert" button for the topic erases changes for the topic and redisplays its current name and alias.

**Topic Exporter**

The Topic Exporter is used to build a text file of information about selected topics from the knowledge base. It consists of four groups: available topics, options, export topics, and the export file.

To use:

- Select topic type(s) to display in the Available Topics list from the Available topics options list. This will load the Available Topics list with topics.
- Select which topic(s) to use from the Available Topics list. Multiple selection is allowed by holding down either the shift or control keys.
- Select the type(s) of relations to use. This will populate the Export Topics list.
- If you only want to see the end points of the dependency networks (typically elemental data or goal groups) check "No intermediates" in the Filter options. This will strip out all intermediate dependency networks.
- Select the type(s) of information to include. This will build the Export File.
- If you want to change the way the data in the export file is delimited, select the Delimiting method.
- Review the Export File. If you want to change any options, go ahead. The export file will be updated accordingly.
- Save the Export File by pressing the "Save Export File..." button.
## Dependency Overview

The Dependency Overview illustrates the connections between objects in the knowledge base. Clicking on a topic button highlights the connections for that topic, both its dependents and antecedents. Double clicking the topic button opens the window for that topic.

The "+" magnifying glass button zooms in the view.

The "-" magnifying glass button zooms out the view.

The horizontal lines button puts the view of the knowledge base in top down order (dependents are toward the top, antecedents to the bottom).

The vertical lines button puts the view of the knowledge base in left right order (dependents are toward the left, antecedents to the right).

## Automated HTML Documentation

The Automated HTML Documentation tool writes hyper-linked HTML files for the entire knowledge base. Select a directory into which to write the files by clicking on the "Target Directory:" button. To create a new directory, while the directory selection dialog is open, type in the new directory name.

Choose whether to include images of the dependency networks by checking the "Embed" checkbox. Choose either GIF or JPEG file type for the images.

Checking "Force file names to lowercase" causes all filenames and their references to be written in lowercase. Otherwise file names are mixed case.

Press the "Generate HTML Files" button to start the process. Progress status will be displayed near the top of the window.

## Knowledge Base Scrubber

The Knowledge Base Scrubber is a wizard-like tool for systematically removing unwanted or unused objects from the knowledge base. Scrubbing the knowledge base is a step by step iterative process. At any time the process can be terminated without causing a problem. However, any changes made will have already been committed to the knowledge base. The only recourse to rollback the changes is to revert to a previously saved version from disk. **In other words: use this tool with caution.**

To use the scrubber, follow the directions on the left side of the window at each step.
**Topic Selector**

The topic selector is a list of topics. Clicking on the topic attaches that topic to the network at the currently selected node. To create a new topic, click on the "New..." line. If a topic cannot be used because of circularity it will be grayed out.

**Advanced feature:** the topic selector has a popup menu (right mouse click) with two options which can be toggled:

- **stay on top** if checked the window does not close after selecting a topic
- **suppress dialog** if checked no dialog for argument assignment is opened after selecting a topic

**Influences Viewer**

Topics which influence the current topic are listed in influence rank order along with a numerical influence value.

The buttons, left to right, are: copy the list to the clipboard, save the list to disk, and refresh the list. The "Elemental data only" checkbox, when checked, limits the list to only simple data links.
The knowledge base database browser is used to manage linkages to databases.

Assign or create tables to link to the knowledge base in the **Setup** tab. Press the "Link" button to connect the tables to the knowledge base. Use the "Create New Tables..." wizard to make appropriately configured new tables. Table linkages can be changed only when the knowledge base is not linked to tables. More info on the tables...

As tables are selected, tabs for each are added to the window for editing. Once the link is established, a **Run** tab is available. From this tab you can save the data in the kb to the current row ("Save Row") or to a new row ("Append Row" button), load data into the kb (dropdown listbox), or process all the rows in the database ("Batch Process" button).
### Knowledge Base Database Tables

<table>
<thead>
<tr>
<th>Dbf Table</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>optional</td>
<td>Coordinates columns in other tables with topic names. In most tables (all except aliases and stats) each column represents a topic. To be associated with a topic the column name must match a topic name, topic alias, or an entry in this table.</td>
</tr>
<tr>
<td>Inputs</td>
<td>yes</td>
<td>Contains input data.</td>
</tr>
<tr>
<td>Flags</td>
<td>optional</td>
<td>Used to flag topics to be ignored. Use with caution!</td>
</tr>
<tr>
<td>Outputs</td>
<td>yes</td>
<td>Stores the results (topic values) after processing.</td>
</tr>
<tr>
<td>Influences</td>
<td>optional</td>
<td>Stores the influences of missing data.</td>
</tr>
<tr>
<td>Ranks</td>
<td>optional</td>
<td>Stores the influence ranks of missing data.</td>
</tr>
<tr>
<td>Stats</td>
<td>optional</td>
<td>Stores overall influence rank statistics of missing data.</td>
</tr>
</tbody>
</table>
Table Creation Wizard

The table creation wizard is used to build database tables that can be linked to the knowledge base.

Select the types of tables to build. Input and output are required for linking. All others are optional.

Topics of interest are those topics you want to view as outputs in the output table. Typically these are goals but can be calculated data links or even elemental data links. The wizard makes sure the appropriate antecedents are included in the input table.

Populate the Topics of Interest list by double clicking an available topic, pressing one of the topic type buttons, or by selecting available topics and pressing "Antecedents" or "Dependents" buttons.

Remove a topic from the Topics of Interest list by double clicking the topic or by selecting it and pressing the "Remove" button.

Press the "Create Tables..." button to create the tables. A series of dialogs will request file names for new tables.
NODE REFERENCE

Node reference conventions

The node references section describes in detail the functioning of each of the node varieties using the following organization and conventions:

**Visual Representation**

A picture of the node, how it is represented on screen.

**Function**

A description of the node, how it is supposed to function, how it is used, etc.

**Class Hierarchy**

A listing of the family line of the node.

**Constraints**

The minimum and maximum number of children, valid input ranges, and any other limiting factors.

**Node Calculus**

A verbal description and mathematical model, where appropriate, of how the node handles the evaluate message. The following are used throughout:

- \( i \) the \( i^{th} \) child node
- \( n \) the number of children
- value the node's evaluated result
- value, the \( i^{th} \) child node's evaluated result
- boss the boss node for the proxy

**Example**

A diagram and chart of sample cases and their results.

**Related Nodes**

abs acos Addition AND asin atan ave ave-nz Calculated Data Link ceil cos cosh Crisp Arguments Data Link Proxy Deg2Rad Division e equation ERROR exp false floor Fuzzy Arguments Fuzzy Curve Goal Goal Proxy Goal Group Goal Proxy if k ln log max min mod multiplication NOT Note OR Pi pow Rad2Deg Simple Data Link sin sinh SOR sqrt subtraction/negation switch tan tanh true undetermined XOR x-y pair
**Abs - Absolute Value**

**Description**

The abs node calculates the absolute value of any node attached to it.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints

**Node Calculus**

The abs node returns the absolute value of its child node.

\[ \text{value} = |\text{value}_1| \]

**Related Nodes**

ave ave-nz ceil floor max min
Acos - Arc Cosine

Description

The acos node returns the arc cosine of the value of its child node.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1
Maximum number of child nodes: 1
Valid input range: -1. to +1.
Output range: 0 to Pi (radians)

Node Calculus

The acos node's value is the arc cosine, in radians, of the value of its child node. If the input value is out of range then the acos node returns an ERROR value.

For valid value₁:

\[ \text{value} = \cos^{-1}(\text{value}_1) \]

For invalid value₁:

\[ \text{value} = \text{ERROR} \]

Related Nodes

asin atan cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tan tanh
**Addition**

**Description**

The add node returns the sum of the values of the nodes attached to it.

**Class Hierarchy**

DNode:DCalcNode:DAddNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: no limit

Valid input range: no constraints

**Node Calculus**

The add node's value is the sum of its child nodes' values.

\[
\text{value} = value_1 + value_2 + \ldots + value_n
\]

**Related Nodes**

Division multiplication subtraction/negation
Description

The AND node is a Boolean AND based on the fuzzy MIN function modified to compensate for missing data. In its purest use, it evaluates to true when all its child nodes are true and false when any child node is false.

Class Hierarchy

DNode:DAndNode

Constraints

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

Valid input range: -1.00 to +1.00 (false to true)

Node Calculus

See following section for a discussion of the calculus of AND nodes.
Example

AND Node Example Case Table

<table>
<thead>
<tr>
<th>Weight -&gt;</th>
<th>AND</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>100% True</td>
<td>100% True</td>
<td>100% True</td>
<td>100% True</td>
<td>100% True</td>
<td></td>
</tr>
<tr>
<td>100% False</td>
<td>100% False</td>
<td>(anything)</td>
<td>(anything)</td>
<td>(anything)</td>
<td></td>
</tr>
<tr>
<td>25% True</td>
<td>100% True</td>
<td>100% True</td>
<td>undetermined</td>
<td>100% True</td>
<td></td>
</tr>
<tr>
<td>25% True</td>
<td>undetermined</td>
<td>undetermined</td>
<td>100% True</td>
<td>undetermined</td>
<td></td>
</tr>
<tr>
<td>50% True</td>
<td>50% True</td>
<td>50% True</td>
<td>50% True</td>
<td>50% True</td>
<td></td>
</tr>
<tr>
<td>100% False</td>
<td>50% True</td>
<td>50% True</td>
<td>100% False</td>
<td>50% True</td>
<td></td>
</tr>
<tr>
<td>8.3% True</td>
<td>50% True</td>
<td>50% True</td>
<td>undetermined</td>
<td>undetermined</td>
<td></td>
</tr>
</tbody>
</table>

Related Nodes

NOT OR SOR XOR
Fuzzy math in AND nodes

The following is an overview of the fuzzy math used to determine the value at an AND node.

\[
\text{AND value} = \minScore + \frac{(\text{weightedAve} - \minScore)}{\text{distance FALSE to TRUE}}
\]

- Traditional fuzzy value, as a starting point
- Distance between bulk value of the child nodes and the chosen node value, i.e., how “different” it is from the rest. Ranges from 0 - ∞.
- \(\minScore\)'s distance from FALSE

"Pulling strength" of chosen node, ranging from 0 - 1, with 0 giving it full influence.

(the strength increases as it leaves the undetermined region and approaches FALSE)
Asin - Arc Sine

Description

The asin node returns the arc sine of its child node.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1
Maximum number of child nodes: 1
Valid input range: -1. to +1.
Output range: -Pi/2 to +Pi/2 (radians)

Node Calculus

The asin node's value is the arc sine, in radians, of the value of its child node. If the input value is out of range then the asin node returns an ERROR value.

For valid value₁:

\[ \text{value} = \sin^{-1}(\text{value₁}) \]

For invalid value₁:

\[ \text{value} = \text{ERROR} \]

Related Nodes

atan cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tan tanh
Atan - Arc Tangent

**Description**

The atan node returns the arc tangent of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints

Output range: -Pi/2 to +Pi/2 (radians)

**Node Calculus**

The atan node's value is the arc tangent, in radians, of the value of its child node.

\[ \text{value} = \tan^{-1}(\text{value}_1) \]

**Related Nodes**

acos asin cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tan tanh
**Ave - Average**

**Description**

The ave node returns the average value of the values of its child nodes.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: no limit

Valid input range: no constraints

**Node Calculus**

The ave node's value is calculated by dividing the sum of the values of its child nodes by their count.

\[
\text{value} = \frac{(\text{value}_1 + \text{value}_2 + \ldots + \text{value}_n)}{n}
\]

**Related Nodes**

abs ave-nz ceil floor max min mod
Ave-nz - Average-non zero

Description

The ave-nz node returns the average value of its child nodes whose value is not zero.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1
Maximum number of child nodes: no limit
Valid input range: no constraints

Node Calculus

The ave-nz node's value is calculated by dividing the sum of the values of its child nodes by the count of child nodes that are not zero.

\[
\text{value} = \frac{\text{value}_1 + \text{value}_2 + \ldots + \text{value}_n}{n}
\]

where \( n \) = number of child nodes whose values \( \neq \) zero

Related Nodes

abs ave ceil floor max min mod
**Calculated Data Link**

**Description**

The calculated data link holds an argument list and a single equation node that is evaluated to return a value to the data link. This value can then be used in other calculations or compared to a list of arguments to derive a trueness value for logical evaluation in a network. See "Data Link Proxy".

The argument list for the data link is a collection of all the arguments used to test the data link. The data link's proxies reference these arguments rather than each proxy having its own unique arguments. This method provides economy of objects and allow global editing of the arguments.

**Class Hierarchy**

DNode:DBossNode:DVarNode:DCalculatedVarNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The calculated data link simply returns the value of its attached equation node which in turn returns the results of its child node (if any).

\[ \text{value} = \text{value}_1 \]

**Related Nodes**

- Crisp Arguments
- Data Link Proxy
- equation
- Fuzzy Arguments
- Fuzzy Curve
- if
- Simple Data Link
- switch
- x-y pair
Ceil - Ceiling

**Description**

The ceil node returns the ceiling of the value of its child node. In other words, it rounds up the child's value to the smallest integer not less than it.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1
Maximum number of child nodes: 1
Valid input range: no constraints

**Node Calculus**

The ceil node's value is calculated by finding the smallest integer not less than the value of its child node.

**Example**

ceil (34.2323) = 35
ceil (34.0000) = 34
ceil (-123.45) = -123

**Related Nodes**

abs ave ave-nz floor max min mod
**Cos - Cosine**

**Description**

The cos node returns the cosine of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints, but must be in radians

**Node Calculus**

The cos node returns the cosine of the angle (in radians) represented by its child node's value.

\[ \text{value} = \cos(\text{value}) \]

**Related Nodes**

acos asin atan cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tan tanh
Cosh - Hyperbolic Cosine

Description

The cosh node returns the hyperbolic cosine of the value of its child node.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints

Node Calculus

The cosh calculates the hyperbolic cosine of the value of the child node using the following formula. If the results overflow, ERROR is returned.

\[
\text{value} = \frac{e^x + e^{-x}}{n}
\]

where \( x = \text{value1} \)

Related Nodes

acos asin atan cos Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tan tanh
Crisp Argument

Description

The crisp argument node is used to derive a trueness level from a data link by comparing its expression to the data link's value. It returns true, false, or undetermined.

Class Hierarchy

DNode:DArgNode:DCrispArgNode

Constraints

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Node Calculus

The crisp argument node's value is true, false, or undetermined. The argument is built up of an operator and an operand or two operators and two operands. When two operators and operands are used both operator/operand pairs must be satisfied for true to be returned. In other words their results are logically "anded" together.

Operands can be either numeric or textual. Arguments are read as:

"The argument is true when the value of the data link is [operator] [operand]"

or

"The argument is true when the value of the data link is both [operator] [operand] AND also [operator] [operand] ".

Operators are limited to the following lists. Operators in the arithmetic category work both on numeric values and on strings. If the operand contains anything other than digits, commas, periods, and signs it is considered to be a string. Operators in the string oriented category always consider the operand to be a string, even if it could otherwise be considered numeric.

Arithmetic Operators:

- < "less than"
String Oriented Operators:

- **is in** A value of the data link is in the operand. Case sensitive. A value of the data link must be a sub-string of the operand. "red" is in "redden", but not in "read" or "Redden"; "12" is in "33124", but not in "33142". "red" is in "red".
- **contains** A value of the data link contains the operand. Case sensitive. The operand must be a sub-string of a value of the data link. "5YR2/2" contains "5YR", but "10YR4/3" and "5yr2/2" do not. "red" contains "red".
- **is like** A value of the data link is like the operand. *Not currently implemented within NetWeaver.*
- **sounds like** A value of the data link sounds like the operand. *Not currently implemented within NetWeaver.*

**Related Nodes**

- Calculated Data Link
- Data Link Proxy
- Fuzzy Arguments
- if Simple Data Link
- switch
Data Link Proxy

Description

The Data Link Proxy provides dependency network access to a data link. The value of the data link can either be used directly in calculations or it can be compared to a list of arguments to provide a trueness level.

The data link proxy can be flagged as "required data" so that when its boss data link has no values it can communicate to its network that it is not valid.

Class Hierarchy

DNode:DProxyNode:DVarHolderNode

Constraints

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Node Calculus

If the compare directly option is on, then the data link proxy node returns the value of the data link.

If the compare to an argument list is on, then it returns the maximum trueness level of its arguments, in the same fashion as an OR node. See "Crisp Arguments" and "Fuzzy Arguments" for information on how arguments are evaluated.

if comparing directly:

\[
\text{value} = \text{value}_{\text{boss}}
\]

if comparing to arguments:

\[
\text{value} = \max(\text{value}_{\text{argument}_1}, \text{value}_{\text{argument}_2}, \ldots, \text{value}_{\text{argument}_n})
\]

Related Nodes
Calculated Data Link Crisp Arguments Fuzzy Arguments if Simple Data Link switch
Deg2Rad - Degrees to Radians

Description

The deg2rad node is used to convert an angular measure in degrees to one in radians. The NetWeaver trig functions require values in radians.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1
Maximum number of child nodes: 1
Valid input range: no constraints

Node Calculus

The deg2rad node returns the degree to radian conversion of the value of its child node.

\[ \text{value} = \text{value}_1 \times \frac{\pi}{180} \]

Related Nodes

acos asin atan cos cosh Pi Rad2Deg sin sinh tan tanh
Division

Description

The division node returns the quotient of the values of its two child nodes. Sometimes referred to as the "guzzintah" node.

Class Hierarchy

DNode:DCalcNode:DDivNode

Constraints

Minimum number of child nodes: 2

Maximum number of child nodes: 2

Valid input range: no constraints on first node, second node must be non-zero

Node Calculus

The division node's value is the value of its first node divided by the value of its second node. If the value of the second node is zero, then the node returns ERROR.

\[
\text{value} = \frac{\text{value}_1}{\text{value}_2}
\]

Related Nodes

Addition Division multiplication subtraction/negation
Description

The e node is a constant node that always evaluates to e (2.718281828).

Class Hierarchy

DNode:DCalcNode:DConstantNode:DENode

Constraints

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Node Calculus

The e node's value is 2.718281828.

value = 2.718281828

Related Nodes

ERROR exp false k ln log Pi pow sqrt true undetermined
**Equation**

**Description**

The Equation node is the node that attaches a network to a calculated data link. It passes the value of its child.

**Class Hierarchy**

DNode: DCalcNode: DEquationNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints

**Node Calculus**

The Equation node returns the value of its child node.

\[ \text{value} = \text{value}_1 \]

**Related Nodes**

[Calculated Data Link]
**Description**

The ERROR node is a constant node that always evaluates to ERROR. It is typically used to simulate an error in a calculated data link or to invalidate a calculated data link under some given conditions.

**Class Hierarchy**

DNode:DCalcNode:DConstantNode:DErrorNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The ERROR node always returns ERROR. ERROR is an internal flag value used to indicate networks or sections of networks that are not valid. When a node receives an ERROR value from one of its child nodes it also will pass ERROR unless the path of the child node is inconsequential, such as an unused path in a switch, if, or SOR node.

\[
\text{value} = \text{ERROR}
\]

**Related Nodes**

| e | false | k | Pi | true | undetermined |
Exp - Exponential

Description
The exp node returns the exponential $e$ to the power of the value of its child node.

Class Hierarchy
DNode:DCalcNode:DFuncNode

Constraints
Minimum number of child nodes: 1
Maximum number of child nodes: 1
Valid input range: no constraints, but results may under or overflow

Node Calculus
The exp node's value is calculated by raising $e$ to the power of the value of the child node.

$$\text{value} = e^x$$

where $x$ is the value of the child node

Related Nodes
$e$ $\ln$ $\log$ $\text{pow}$ $\sqrt{}$
The False node is a constant node that always evaluates to False (-1.00).

**Class Hierarchy**
DNode:DCalcNode:DConstantNode:DFalseNode

**Constraints**
Minimum number of child nodes: n.a.
Maximum number of child nodes: n.a.

**Node Calculus**
The False node always returns False.

\[ \text{value} = -1.00 \]

**Related Nodes**
e ERROR k Pi true undetermined
Fuzzy Curve

Function

A Fuzzy Curve node is used in conjunction with **XY pair** nodes to define a dynamic fuzzy curve.

Class Hierarchy

DNode:DProxyNode:DFuzzyCurveNode

Constraints

The minimum and maximum number of children, valid input ranges, and any other limiting factors.

Node Calculus

The value passed by the Fuzzy Curve node is the y value at the x position on the fuzzy curve defined by its XY pair nodes. The x value is provided by the boss node (either a data link or goal) for the Fuzzy Curve node. On evaluation the Fuzzy Curve node orders the XY pairs and selects the pair whose x values bracket the x value from the boss node. Then a straight line interpolation is performed to calculate the y value from the x and y values of the two XY pair nodes. If the x value is not bracketed by XY pairs then the y value is the same as the XY pair nearest the x value.

Example

A diagram and chart of sample cases and their results.

Related Nodes

Calculated Data Link  Crisp Arguments  Data Link Proxy  Fuzzy Arguments  if Simple Data Link  x-y pair
Goal

Description

The goal node contains two dependency networks: an affirmation network and a negation network. The affirmation network proves the goal true. The negation network is provided as an optional method for defining the cases that prove the goal false under the assumption that it is sometimes much easier to disprove something than it is to prove it.

Class Hierarchy

DNode:DBossNode:DGoalNode

Constraints

Minimum number of child nodes: n.a.
Maximum number of child nodes: n.a.
Valid input range: -1.00 to +1.00 (false to true)

Node Calculus

The goal node evaluates both its dependency networks. The value of the goal is whatever is returned from the affirmation network unless the negation network is more true than the absolute value of the affirmation network. Remember a high trueness level in the negation network indicates the goal is false.

if value\text{neg net} \geq | value\text{aff net} | 

\begin{align*}
\text{value} &= - \text{value}_{\text{neg net}} \\
\text{else} & \quad \text{value} = - \text{value}_{\text{aff net}}
\end{align*}

Related Nodes

Calculated Data Link Goal Proxy Goal Group Simple Data Link
Goal Proxy

Description

The Goal Proxy passes the value of a goal node.

Class Hierarchy

DNode:DProxyNode:DGoalHolderNode

Constraints

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Node Calculus

The goal proxy returns the value of its boss goal.

\[ \text{value} = \text{value}_{\text{boss}} \]

Related Nodes

Data Link Proxy Goal Goal Group Proxy
The Goal Group node organizes a group of goal nodes to either select the best goal of the group or to use the cumulative effect of the goals to build a higher level characteristic. These uses are qualitative and quantitative, respectively.

Qualitative groups are composed of discrete goals. Discrete goals are ones that are unique to themselves and not necessarily related in a symbiotic fashion to each other. Discrete goal groups (qualitative groups) can be used to either rank or select a best goal by evaluating all the goals in the group and ranking them according to their relative trueness. In this type of group the goals may not necessarily be competing against each other, rather each goal could stand on its own as a decision.

An example of a qualitative goal group with competing goals might be an insect key, a goal group of insects where the purpose is to determine which one of the many insects in the group is the one of interest. Each insect would be represented by a goal that would describe the mix of characteristics that indicate the presence of the insect. Once enough information has been input, say body shape and size, coloration, etc., only one insect (goal) should be true and, if we built things well, the others should be false. By comparison, we could build a knowledge based system to select aerobic sports. We might include as goals: biking, running, and skiing and base our decision on the available time, the weather, and equipment condition. It is conceivable that all goals in this group could be true or even all goals could be false.

Quantitative groups are composed of goals that represent positions along a continuum of possible values. (The position held by a goal is determined from the weight associated with the goal.) Goals that are positions on a continuum represent specific positions relative to each other goal in the goal group and are used to return a continuous value. As an example, the group "Dangerous" might include the goals "Danger.low", "Danger.moderate", and "Danger.high". The group would be used to select the best point along the danger continuum from low to high.

Quantitative goal groups are used to determine the point on the continuum that best fits the given data using the fuzzy logic technique of finding the centroid of the goal group. The centroid is calculated using the weights associated with the goals as their center points along the x-axis and the trueness levels of the goals as their heights. All the goals in the group contribute to the centroid calculation.
Goal groups are also handy for grouping intermediate goal results for debugging or evaluating.

**Class Hierarchy**

DNode:DBossNode:DGoalGroupNode

**Constraints**

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

**Node Calculus**

The goal group's value is either the index number of the most true goal in the group (discrete goals) or is the centroid of the values of the goals (goals as points on a continuum).

**For discrete goals:**

\[ \text{value} = i, \ \text{where} \ i = \text{index of max}(\text{value}_1, \text{value}_2, \ldots, \text{value}_n) \]

**For goals as points on a continuum:**

The centroid is calculated by dividing the sum of moments by the sum of magnitudes. The moments are the magnitudes multiplied by the weights (weights represent the position of the goal along the continuum).

\[ \text{value} = \frac{(\text{value}_1 + 1)\text{weight}_1 + (\text{value}_2 + 1)\text{weight}_2 + \ldots + (\text{value}_n + 1)\text{weight}_n}{(\text{value}_1 + 1) + (\text{value}_2 + 1) + \ldots + (\text{value}_n + 1)} \]

**Example**

An example of a quantitative goal group could be a risk rating system where we want to determine the level of risk associated with a given scenario. The group could have goals for definable levels of risk: Risk.vhigh, Risk.high, Risk.mod, Risk.low, Risk.vlow. The goal group would evaluate each goal and then combine the results to give a single value that represents the center of mass of the area under the curve defined by the individual
goals. In the example below a moderate risk (Risk.mod) has been determined.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Weight</th>
<th>Value</th>
<th>Magnitude</th>
<th>Moment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk.vLow</td>
<td>1.00</td>
<td>100% False</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Risk.low</td>
<td>2.00</td>
<td>75% True</td>
<td>1.750</td>
<td>3.500</td>
</tr>
<tr>
<td>Risk.mod</td>
<td>3.00</td>
<td>100% True</td>
<td>2.000</td>
<td>6.000</td>
</tr>
<tr>
<td>Risk.high</td>
<td>4.00</td>
<td>Undetermined</td>
<td>1.000</td>
<td>4.000</td>
</tr>
<tr>
<td>Risk.vhigh</td>
<td>5.00</td>
<td>45% False</td>
<td>0.550</td>
<td>2.750</td>
</tr>
</tbody>
</table>

Sums: 5.300  16.250

Centroid: 3.066

Related Nodes

Goal Goal Group Proxy
Goal Group Proxy

Description

The Goal Group Proxy passes the value of its goal group.

Class Hierarchy

DNode:DProxyNode:DGroupHolderNode

Constraints

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Node Calculus

The goal group proxy returns the value of its goal group.

\[ \text{value} = \text{value}_{\text{boss}} \]

Related Nodes

Data Link Proxy Goal Proxy Goal Group Goal Group Proxy
**Description**

The if node selects a value or path based on the value of its first child node. The if node mode of operation is dependent on how many child nodes it has. It returns 0 or 1 if it has only one child node; 0 or the value of the second child node for 2 nodes; the value of the second or third child node for 3 nodes; the value of its second, third, or fourth child nodes if it has 4 child nodes.

**Class Hierarchy**

DNode:DCalcNode:DIfNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 4

Valid input range: no constraints

**Node Calculus**

The first (leftmost) child node is the control node. The if node uses its value and its weight to decide on what to return. The specific result returned is respective to the number of attached child nodes. All variations of the if node return ERROR when the control node's value is ERROR. In general terms the if node functions as follows:

<table>
<thead>
<tr>
<th>Value1</th>
<th>True</th>
<th>Undetermined</th>
<th>False</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 child if node</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>ERROR</td>
</tr>
<tr>
<td>2 child if node</td>
<td>Value2</td>
<td>0</td>
<td>0</td>
<td>ERROR</td>
</tr>
<tr>
<td>3 child if node</td>
<td>Value2</td>
<td>Value3</td>
<td>Value3</td>
<td>ERROR</td>
</tr>
<tr>
<td>4 child if node</td>
<td>Value2</td>
<td>Value3</td>
<td>Value4</td>
<td>ERROR</td>
</tr>
</tbody>
</table>

where:
True is "sufficiently true", i.e. \( \text{value}_1 \geq \text{weight}_1 \)

Undetermined is "not sufficiently true or false", i.e. \( \text{weight}_1 > \text{value}_1 > -\text{weight}_1 \)

False is "sufficiently false", i.e. \( \text{value}_1 \leq -\text{weight}_1 \)

**Example**

<table>
<thead>
<tr>
<th>If Node - 1 Child</th>
<th>A Value</th>
<th>A Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>75% True</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>50% True</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>undetermined</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>25% False</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>6.23</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>25% True</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>0.1</td>
<td>any</td>
</tr>
<tr>
<td>ERROR</td>
<td>ERROR</td>
<td>any</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>If Node - 2 Child</th>
<th>A Value</th>
<th>A Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>75% True</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>50% True</td>
<td>0.5</td>
</tr>
</tbody>
</table>
If Node - 3 Child

<table>
<thead>
<tr>
<th>if</th>
<th>A Value</th>
<th>A Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>75% True</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>50% True</td>
<td>0.5</td>
</tr>
<tr>
<td>C</td>
<td>undetermined</td>
<td>0.5</td>
</tr>
<tr>
<td>C</td>
<td>25% False</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>6.23</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>25% True</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>-0.1</td>
<td>any</td>
</tr>
<tr>
<td>ERROR</td>
<td>ERROR</td>
<td>any</td>
</tr>
<tr>
<td>Related Nodes</td>
<td>Calculated Data Link Crisp Arguments Data Link Proxy Fuzzy Arguments Fuzzy Curve Simple Data Link switch x-y pair</td>
<td></td>
</tr>
</tbody>
</table>
**k - Constant**

**Description**

The k node is a constant node used to define arbitrary constant values.

**Class Hierarchy**

DNode:DCalcNode:DConstantNode:DKNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

Valid input range: anything that will evaluate to a numeric answer

**Node Calculus**

The k node returns the numeric equivalent of the text of its name.

\[
\text{value} = \text{numeric(name)}
\]

**Example**

- \( \text{56} \) = 56.0
- \( \text{6 ft} \) = 6.0
- \( \text{6 ft 4 in} \) = 6.0
- \( \text{Bob} \) = 0.0
- \( \text{45.385} \) = 45.3

**Related Nodes**

e ERROR false Pi true undetermined
**Ln - Natural Logarithm**

**Description**

The ln node returns the natural logarithm of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: must be a positive number

**Node Calculus**

The ln node returns the natural logarithm of the value of the child node.

\[ \text{value} = \ln(x) \]

where \( x = \text{value}_1 \)

**Related Nodes**

e exp log pow sqrt
Log - Base 10 Logarithm

**Description**

The log node returns the base 10 logarithm of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: must be a positive number

**Node Calculus**

The log node returns the base 10 logarithm of the value of the child node.

\[
\text{value} = \log_{10}(x)
\]

where \( x = \text{value}_1 \)

**Related Nodes**

\( e \, \text{exp} \, \text{ln} \, \text{pow} \, \text{sqrt} \)
Max - Maximum

Description
The max node returns the maximum value of the values of its child nodes.

Class Hierarchy
DNode:DCalcNode:DFuncNode

Constraints
Minimum number of child nodes: 1
Maximum number of child nodes: no limit
Valid input range: no constraints

Node Calculus
The max node returns the maximum value of the child nodes.

\[ \text{value} = \max(\text{value}_1, \text{value}_2, \ldots, \text{value}_n) \]

Related Nodes
abs ave ave-nz ceil floor min mod
Description

The min node returns the minimum value of the values of its child nodes.

Class Hierarchy

DNode:DCalcNode:DFuncNode

Constraints

Minimum number of child nodes: 1

Maximum number of child nodes: no limit

Valid input range: no constraints

Node Calculus

The min node returns the minimum value of the child nodes.

\[
\text{value} = \min(\text{value}_1, \text{value}_2, \ldots, \text{value}_n)
\]

Related Nodes

abs ave ave-nz ceil floor max mod
**Mod - Modulo**

**Description**

The mod node calculates the remainder after division of two numbers.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: value1 > value2 > 0

**Node Calculus**

The mod node calculates the remainder of the division of the value of the two child nodes.

\[ \text{value} = x \mod y \]

where \( x = \text{value}_1, y = \text{value}_2 \)

**Related Nodes**

abs    ave    ave-nz    ceil    floor    max    min
Multiplication

**Description**

The Multiplication node calculates the product of all its child nodes.

**Class Hierarchy**

DNode:DCalcNode:DMultNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: no limit

Valid input range: no constraints

**Node Calculus**

The Multiplication node returns the product of the values of its child nodes.

\[ \text{value} = \text{value}_1 \times \text{value}_2 \times \ldots \times \text{value}_n \]

**Related Nodes**

Addition Division subtraction/negation
Description

The NOT node is a Boolean NOT. It flips the logical value of its child node from true to false or false to true.

Class Hierarchy

DNode:DNotNode

Constraints

Minimum number of child nodes: 0

Maximum number of child nodes: 1

Valid input range: -1.00 to +1.00 (false to true)

Node Calculus

The NOT node's value is calculated by flipping the sign of its child node.

\[ \text{value} = -\text{value}_1 \]

Related Nodes

AND OR SOR XOR
**Description**

The Note node stores text for project documentation that is not attached to any node. Typical uses are change log, credits, future plans, notes to users, etc.

**Class Hierarchy**

DNode:DBossNode:DNoteNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The Note node only stores text. It has no evaluation function.
Simple Data Link

Description

The Simple Data Link node is a holder of data. It has a list of values that can be compared to arguments in a data link proxy node or its value can be used directly in calculations. It also has a list of defaults, a list of choices, and a list of arguments. The data link can be set so that it is required to have a value or that its influence be ignored.

The values in the default list are used only when the data link has no current values in its value list. If the data link has no values in its value list and no default list, then it will return undetermined when it is queried.

The choice list is used by NetWeaver to dynamically construct menus for the selection of possible values in goal group interface window. Arguments can be converted to choices by using the "Args->Choices" menu item in the Edit menu when the data link is highlighted in the project window's object selector. The choices can similarly be converted into arguments via the "Choices->Args" menu item. The choice list has no function in the evaluation of networks.

The argument list for the data link is a collection of all the arguments used to test the data link. The data link's proxies reference these arguments rather than each proxy having its own unique arguments. This method provides economy of objects and allows global editing of the arguments.

If the data link or its proxy has been set to be required the data link will report that it is bogus when it has no values in its value list (not to be confused with the choices list). This feature allows a system to prevent making evaluations when designated key data is missing.

If the data link has been set to be ignored, it, in effect, prunes its attachments to any networks. This feature allows "what if?" scenarios to be evaluated where a decision needs to be evaluated but the data is not able to be found (or, say, not economically able to be found) and the user chooses to eliminate the data link from influencing the evaluation.

Class Hierarchy

DNode:DBossNode:DVarNode:DSimpleVarNode

Constraints
Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The Simple Data Link provides data to be used by its proxy nodes. The evaluation is performed in the arguments via the proxy node. Its role is that of data repository.

If the data link is in ignore mode, the calling proxy's parent node performs its evaluation as if the proxy for the data link was not attached.

**Related Nodes**

- Calculated Data Link
- Crisp Arguments
- Data Link Proxy
- Fuzzy Arguments
- Fuzzy Curve
- switch
- x-y pair
**Sin - Sine**

**Description**

The sin node returns the sine of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints, must be in radians

**Node Calculus**

The sin node returns the sine of the angle (in radians) represented by its child node's value.

\[ \text{value} = \sin(\text{value}_1) \]

**Related Nodes**

acos asin atan cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sinh sqrt tan tanh
**Sinh - Hyperbolic Sine**

**Description**

The sinh node returns the hyperbolic sine of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints, but result may overflow

**Node Calculus**

The sinh calculates the hyperbolic sine of the value of the child node using the following formula. If the results overflow, ERROR is returned.

\[
\text{value} = \frac{e^x - e^{-x}}{2}
\]

where \(x = \text{value}_1\)

**Related Nodes**

acos asin atan cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sqrt tan tanh
Description

The Sequential OR (SOR) node selects the first path of adequate quality. It is used to pick among competing evaluation methods where the methods are arranged in descending order of preference. Often the first choice method of determining some property is not available but another, less desirable method is available. The SOR node provides a way of switching between these competing methods based on how well each method is currently functioning. This quality of the path is determined by the magnitude of the child node's value compared to the weight of the child node.

Class Hierarchy

DNode:DSorNode

Constraints

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

Valid input range: -1.00 to +1.00 (false to true)

Node Calculus

The SOR node's value is the value of its first child node whose absolute value is greater than or equal to its weight. If no child node meets these criteria, then the SOR node returns undetermined.

\[
\text{if } i \text{ is found} \\
\quad \text{value} = \text{value}_i \\
\text{else} \\
\quad \text{value} = 0.00
\]

where \( i \) is the first \( i \) where \( |\text{value}_i| \geq \text{weight}_i \) for \( i = 1 \) through \( n \)

Example
In the above example, the values from the chosen paths are highlighted for each case. The value of the A node is returned by the SOR node only when it is fully true or fully false because its weight is 1.00. The third case shows the C branch being selected because neither the A or B nodes have sufficient value compared to their respective weights. The last case shows that none of the nodes have sufficient quality to be used so the SOR node returns undetermined.

**Related Nodes**

**AND NOT OR XOR**
**Sqrt - Square Root**

**Description**

The sqrt node returns the square root of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: must be a positive number

**Node Calculus**

The sqrt node returns the square root of the value of its child node.

\[ \text{value} = \sqrt{\text{value}_1} \]

**Related Nodes**

\[ e \exp \ln \log \text{pow} \]
Subtraction / Negation

Description
The Subtraction / Negation node either takes the difference of the values of its two child nodes or returns the negation of its single child node.

Class Hierarchy
DNode:DCalcNode:DSubNode

Constraints
Minimum number of child nodes: 1
Maximum number of child nodes: 2
Valid input range: no constraints

Node Calculus
The Subtraction / Negation node's function is determined by the number of child nodes it has. If it has only one child node, it returns the negation of the value of the child node. If it has two child nodes it returns the difference between the values of the two child nodes.

if n = 1
    value = - value_1
if n = 2
    value = value_1 - value_2

Related Nodes
Addition Division multiplication
**Description**

The Switch node is used to select a path based on the comparison of its boss data link to a list of arguments. The path chosen is the one that corresponds to the first true argument in the list. If there are more child nodes than arguments the first surplus child node becomes the default.

**Class Hierarchy**

DNode:DSwitchNode

**Constraints**

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

Valid input range: no constraints

**Node Calculus**

The Switch node's value is the value of the chosen child node. The chosen child node is the one that corresponds to the first true argument in the switch node. If none of the arguments are true and there are more child nodes than arguments, the n + 1 child node is chosen as a default node. If the boss data link is not determined (has no value) then undetermined is passed.

if the data link has data
  
  value = value$_i$

else
  
  value = 0.00

where i is the sequence number of the first argument that is true or the number of arguments + 1 if no arguments are true.

**Example**
In the above example, the value of the B branch is returned by the switch node if the Color is red because "= red" is the first true argument and its position in the argument list is 2. So the 2nd (left to right) child node (B) is selected. B is also chosen when the Color includes both red and green because the red argument comes before the green argument and only one path can be chosen. The D path is chosen as a default when Color has a value (in this case yellow) and there is no match to an argument. No path is chosen (undetermined is returned) when Color has no value.

**Related Nodes**

[Calculated Data Link][Crisp Arguments][Data Link Proxy][Fuzzy Arguments][Fuzzy Curve if Simple Data Link][x-y pair]
**Tan - Tangent**

**Description**

The tan node returns the tangent of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints, but the value must be in radians

**Node Calculus**

The tan node returns the tangent of the angle (in radians) represented by its child node's value.

\[ \text{value} = \tan(\text{value}_1) \]

**Related Nodes**

acos asin atan cos cosh Deg2Rad e exp ln log Pi pow Rad2Deg sin sinh sqrt tanh
Tanh - Hyperbolic Tangent

**Description**

The tanh node returns the hyperbolic tangent of the value of its child node.

**Class Hierarchy**

DNode:DCalcNode:DFuncNode

**Constraints**

Minimum number of child nodes: 1

Maximum number of child nodes: 1

Valid input range: no constraints

**Node Calculus**

The tanh calculates the hyperbolic tangent of the value of the child node using the following formula.

\[ \text{value} = \frac{\sinh(x)}{\cosh(x)} \]

where \( x = \text{value}_1 \)

**Related Nodes**

acos asin atan cos cosh Deg2Rad e exp In log Pi pow Rad2Deg sin sinh sqrt tan
**True**

**Description**

The True node is a constant node that always evaluates to True (+1.00).

**Class Hierarchy**

DNode:DCalcNode:DConstantNode:DTrueNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The True node always returns True.

\[ \text{value} = +1.00 \]

**Related Nodes**

- e
- ERROR
- false
- k
- Pi
- undetermined
**Undetermined**

**Description**

The Undetermined node is a constant node that always evaluates to Undetermined (0.00).

**Class Hierarchy**

DNode:DCalcNode:DConstantNode:DUndeterminedNode

**Constraints**

Minimum number of child nodes: n.a.

Maximum number of child nodes: n.a.

**Node Calculus**

The Undetermined node always returns Undetermined.

\[
\text{value} = 0.00
\]

**Related Nodes**

- e
- ERROR
- false
- k
- Pi
- true
**Union**

**Description**

The Union node is used to aggregate values without the prejudice inflicted by using the AND node or the liberality of the OR node. It performs a weighted average of its child nodes.

**Class Hierarchy**

DNode:DUnionNode

**Constraints**

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

**Node Calculus**

The Union node calculates the weighted average of its child nodes:

\[
\text{value} = \frac{\text{value1} \times \text{weight1} + \text{value2} \times \text{weight2} + \ldots + \text{valuen} \times \text{weightn}}{\text{weight1} + \text{weight2} + \ldots + \text{weightn}}
\]
**XOR - Exclusive OR**

**Description**

The XOR node is an exclusive OR. It is true only when only one child node is true and the rest are false. The XOR uses a fuzzy algorithm to determine how true it is when there is missing data or levels of trueness. The return value is the "logical distance" between the two most true nodes. The minimum distance possible is 0, indicating that both nodes have the same value and therefore the XOR would be false. The maximum distance occurs when one node is fully true and others are fully false with the XOR resulting in true.

**Class Hierarchy**

DNode:DXorNode

**Constraints**

Minimum number of child nodes: 0

Maximum number of child nodes: no limit

Valid input range: -1.00 to +1.00 (false to true)

**Node Calculus**

The XOR node's value is calculated by offsetting the difference between its two most true child nodes.

\[
value = -1.00 + value_{\text{max}1} - value_{\text{max}2}
\]

where:

\[\text{max1} = \text{the most true child node}\]

\[\text{max2} = \text{the second most true child node}\]

**Example**
**Example XOR Node Case Table**

<table>
<thead>
<tr>
<th></th>
<th>XOR</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weight -&gt;</strong></td>
<td>-</td>
<td>n.a.</td>
<td>n.a.</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100% True</td>
<td>100% True</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
</tr>
<tr>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
</tr>
<tr>
<td>100% False</td>
<td>100% True</td>
<td>100% True</td>
<td>(anything)</td>
<td>(anything)</td>
<td></td>
</tr>
<tr>
<td>100% False</td>
<td>undetermined</td>
<td>undetermined</td>
<td>undetermined</td>
<td>undetermined</td>
<td></td>
</tr>
<tr>
<td>50% True</td>
<td>50% True</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
<td>100% False</td>
</tr>
<tr>
<td>undetermined</td>
<td>50% True</td>
<td>50% False</td>
<td>50% False</td>
<td>50% False</td>
<td>50% False</td>
</tr>
<tr>
<td>50% False</td>
<td>50% True</td>
<td>undetermined</td>
<td>undetermined</td>
<td>undetermined</td>
<td></td>
</tr>
</tbody>
</table>

**Related Nodes**

AND NOT OR SOR
XY pair

Function

XY pair nodes are used in conjunction with a fuzzy curve node to dynamically define transition points on a fuzzy curve. As many xy pair nodes can be used as needed to define the curve. Positions left to right of the XY pair nodes under a fuzzy curve node are irrelevant because the fuzzy curve node dynamically orders them when evaluation occurs.

The first (left most) child node defines the x position while the other (right most) child node defines the level of trueness at that x position. The children of an XY pair can be any type of nodes.

Class Hierarchy

DNode:DCoordinatePairNode

Constraints

Minimum number of child nodes: 2
Maximum number of child nodes: 2
Valid input range: no constraints

Node Calculus

See Fuzzy Curve node.

Example

A diagram and chart of sample cases and their results.

Related Nodes

Calculated Data Link Fuzzy Arguments Fuzzy Curve if Simple Data Link switch x-y pair