Contents

NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver
NetWeaver

A User's Guide to NetWeaver™

Preliminary Version

Bruce J. Miller
Michael C. Saunders
Penn State University

NetWeaver version 7.6

Overview

NetWeaver™ is a knowledge base construction, maintenance, documentation, and debugging tool written at Penn State University to provide an efficient knowledge engineering environment for cross platform knowledge based systems development. NetWeaver™ is written in C++ and reads and writes platform-independent ASCII scripts that can be read by an embedded inference engine in stand-alone applications. NetWeaver™ renders knowledge dependency networks with a fully editable graphic representation so that the networks appear just like they would on the white board. All the typical Macintosh editing functions are available (i.e., Cut , Copy & Paste) and can be used on single nodes or whole sections of a network. Because the inference engine is built in, networks can be evaluated real-time with nodes changing color to indicate their changing trueness levels. This ability to peer into the logical workings of a knowledge network has really helped domain experts better understand what they "know". On some projects, NetWeaver™ alone is a suitable substitute for stand-alone knowledge based systems.

IcKEE™ (Iconic Knowledge Engineering Environment, Penn State University), also written in C++, performs as the inference engine within NetWeaver™. As an embedded tool, it is available to run under the Macintosh OS, HyperCard, MSDOS, Windows, UNIX and XWindows. IcKEE™ is completely object oriented and not just in the typical programming sense. Each logical operator (node: ANDs, ORs, NOTs, etc.) is handled as an object in memory which knows its connections, holds its own data, performs its own calculations, and communicates via messages to other, connected nodes. IcKEE™ evolved to handle not only standard Boolean connections but also mathematical calculation connections and fuzzy logic. Fuzzy logic is used to handle missing data, to evaluate between competing goals, and for the classical use of determining a variable's membership in a given class. Due to its modular nature, IcKEE™ can be (and has been) extended to embrace other methodologies as the needs arise. An interesting example is the Sequential OR (SOR) node. It was developed as a device to better handle alternative decision scenarios were there is a definite hierarchy of quality level associated with each possible data gathering method. In other words, the SOR node functions as a data route selector; it provides a method for selecting the best choice of paths within the scope of currently available data.

IcKEE™ reads knowledge bases stored in LISP-like ASCII scripts that describe the construction of the knowledge networks. These scripts are editable in a normal text processing environment or, preferably, with NetWeaver™. As the scripts are read, IcKEE™ constructs and links the objects in memory to represent the knowledge base. Knowledge base functions are extremely efficient as all objects are in memory, their routines are all compiled and no on-the-fly interpretation is required. Efficiency is further enhanced by firing only nodes when they have changed. When data changes, the associated variable passes a "dirty" message up through the network. When a dirty message reaches the top of a network, an "evaluate" message is propagated back down through the network. As each node receives the message, it checks itself to see if it is "dirty". If it is "dirty", it fires its evaluation routine and returns its new value to the requesting node once the evaluation routine is done. If the node is not dirty, the message stops propagating and the node returns its current value without performing its evaluation routine. Once the dirty nodes have been resolved adequately (as indicated by the receipt of evaluation results) goal results can be obtained.

NetWeaver™ was developed under cooperative agreements with the USDA Forest Service, Soil Conservation Service, and the USDI National Park Service. NetWeaver™ is currently only available on the Mac, but Windows and XWindows are possible in the near future. For more information contact the Principal Investigator, Michael C. Saunders, Department of Entomology, Penn State University, 501 Ag Science & Industries Bldg., University Park, PA 16802, 814/863-2979, MSAUNDERS@PSUPEN.PSU.EDU., or the author of the software, Bruce J. Miller, BJMILLER@PSUPEN.PSU.EDU.

1
NetWeaver/Ickee Objects

Objects

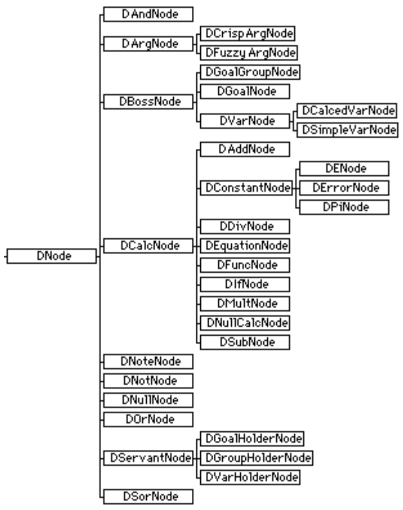
Both NetWeaver and Ickee are built entirely of objects. Ickee has node objects, list objects, the project object, etc. NetWeaver has the application object, document objects, node pane objects (the visual representation of a node), selector objects, etc. These objects encapsulate their own information and have inherited or overridden methods by which they perform their tasks.

The Project

NetWeaver (and Ickee) organizes a knowledge base as a collection of related node objects. The main controlling object is the project. There is only one instance of the project and it is the object through which an application communicates with Ickee. The project holds lists of the major objects and controls global operations such as resetting all of the data links, toggling global variables, and finding a major object by its name, etc.

Nodes

The object class that is the fundamental operational object for Ickee is the node class. Other object classes perform important roles (such as the list classes), but the node class is the one that is used to store and manipulate knowledge related information. The Ickee node class hierarchy:



Node objects have children and parents, which are lists of other nodes to which they pass or from which they receive messages. Nodes also store comments, explanation and weights for documentation and calculation enhancement. Nodes internally keep track of their current states and respond accordingly to messages they receive. Node subclasses have overridden methods to give them appropriate behaviors such as how they perform calculations or how they combine inputs from their children to give a new trueness value.

Three major object classes are used to encode the knowledge base: Data links, Goals, and Goal Groups. Data links manipulate data, goals encode information, and goal groups derive knowledge.

Data Links

Data links are used to hold, fetch, or modify raw data. Data links come in two flavors: simple and calculated. Simple data links fetch and hold data from various sources (databases, GIS map layers, flat files, direct input, environment variables, daemons, etc.) Calculated data links modify data through networks of calculation nodes chosen from a toolbox of logical, arithmetic, trigonometric, selection, summation, etc. nodes.

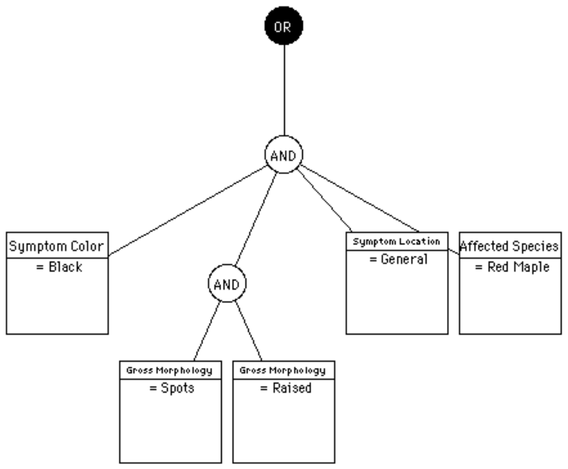
To provide a trueness level that can be used in a goal dependency network, the contents of a data link needs to be compared to arguments. Arguments can be decision level thresholds, lists of qualifiers, or fuzzy set membership functions. Ickee provides two types of arguments: the standard argument and the fuzzy argument. The standard argument compares the data link's value against a threshold function or other qualifier to return a true or false value (or undetermined if the data link has no value). The fuzzy argument compares the data link's value against a fuzzy set membership function that returns a level of trueness based on the degree of membership in the fuzzy set.

Goals

Goals quantify the results of dependency networks as levels of trueness. Each goal has two dependency networks, an affirmation network and a negation network. The affirmation network is used to prove the goal true. Because it is sometimes much easier to prove something false than it is to prove it true, the negation network can be used as a short cut method to disprove a goal.

Goal dependency networks are linkages of logical nodes (ORs, ANDs, etc.) and other major objects (other goals, data links, etc.) that evaluate, communicate, and transform trueness values to supply the goal node with an emerged trueness value.

The dependency network is built like an upside down tree. By convention it starts with an OR node at the top. Other nodes connect to (hang from) this OR node and become children of the OR node and the OR node becomes a parent of the new nodes. The following dependency network for Tar Spot can be read as "Tar Spot is true when the symptom color is 'black' and the gross morphology is 'spots' and 'raised' and the symptom location is 'general' and the affected species is 'red maple'.



When none of the four data links have known values (missing data), the network as a whole evaluates to undetermined. As data links become determined (have values) the network progressively evaluates itself to a new state. For example, if it becomes known that the symptom color is 'black', the symptom color node becomes true and its parent node (the upper AND node) becomes .25 true (the other nodes need to contribute to make it more true). However, if the symptom color was 'white', the symptom color node would become false and so would its parent AND node and the OR node at the top making the network fully false ("the disease is NOT Tar Spot").

There is no real limit to the number of nodes that can hang from another node and there is no real limit to the number of layers of nodes in a network. However, there is a point where the network becomes incomprehensible due to its complexity. Therefore, it is prudent to knowledge engineer the system in digestible chunks, i.e., to modularize it.

Goal Groups

Goal groups come in two flavors: qualitative and quantitative. Qualitative groups are composed of discrete goals while quantitative groups are composed of goals that represent positions along a continuum of possible values.

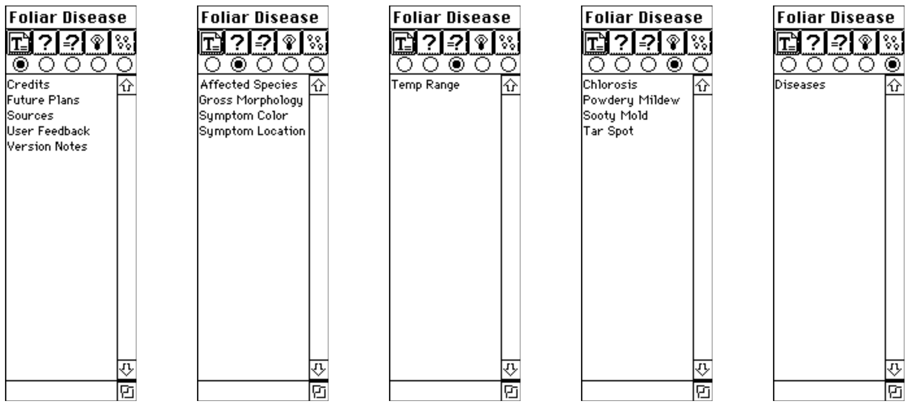
Discrete goal groups (qualitative groups) can be used to either rank or select a best goal by evaluating all the goals in the group and ranking them according to their relative trueness. In this type of group the goals may not necessarily be competing against each other, rather each goal could stand on its own as a decision. For example, we could build a knowledge based system to rank 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 could be true or even all goals could be false. However, we might want to build a system that tries to determine which foliar disease is infecting a tree. In this system each possible disease would be represented by a goal and, once enough information has been ascertained about the symptoms, only one disease (goal) should be true and the rest false. Although, with only partial data, many goals might be partially true (to be interpreted as "we're not sure which disease it is, but it is likely to be one of these:...").

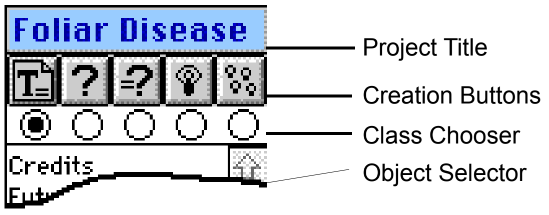
Quantitative goal groups are used to determine the point on the continuum that best fits the given data. That best point is calculated using the fuzzy logic technique of finding the centroid of the goal group. The centroid is calculated using the weights associated with the goals as their center points along the x-axis and the trueness levels of the goals as their heights. All of the goals in the group contribute to the centroid calculation. 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.high, risk.moderate, risk.low. 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.

1
The NetWeaver Windows

The Project Window

The project window is the first window to open. It is always open. It consists of a row of push buttons along the top, a row of five corresponding radio buttons below it and a scrolling object selector. The push buttons create new major objects. The radio buttons select the class of major object to show in the object selector. The object selector selects objects by single clicking on an object name and opens objects by double clicking. Shift double clicking opens the background editor for the object which allows access to otherwise hidden attributes of the object such as its name, weight, etc. Five views of the same project window appear below, each showing a different class of major object within the project.





Object Selector Actions

A single click on an object name selects that object for various operations.

A double click on an object name or selecting "Open" from the File menu opens the appropriate editor for the object:

Notes	Opens the note editor.
Simple Data Links	Opens the data source definition dialog.
Calculated Data Links	Opens the calculated data link network editor.
Goals	Opens the goal dependency network editor.
Goal Groups	Open the goal group interface window.

A shift double click on an object name or selecting "Get Info" from the File menu opens the appropriate enhancement dialog for the object:

Data Links	Opens the data link enhancement dialog.
Goals	Opens the goal definition dialog.
Goal Groups	Opens the goal group definition dialog.

Project Window Buttons

"Note". This button creates a note and opens a simple text editing environment for note taking and system documenting.

"Simple Data Link". This button creates a new simple data link and opens the data link enhancement dialog where the data link can be named and arguments created and edited.

"Calculated Data Link". This button creates a new calculated data link, opens the data link enhancement dialog where the data link can be named and arguments created and edited, and then opens the calculated data link network editor.

"Goal". This button creates a new goal, opens the goal definition dialog where the goal can be named and weighted, and then opens the goal network editor.

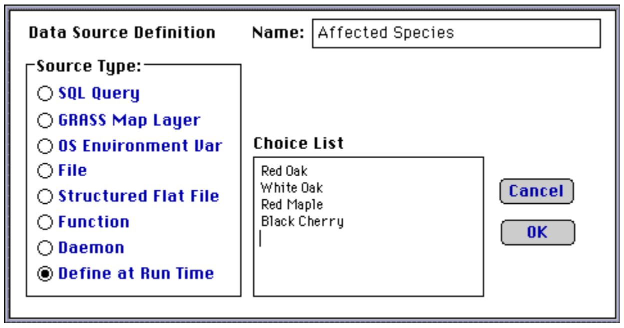
"Goal Group". This button creates a new goal group and opens the goal group definition dialog where the goal group can be named and defined.

◉ ○ ○ ○ ○ "Class Chooser". These radio buttons select between the different classes of major objects to be shown in the object selector. Each radio button corresponds to the class type created by the creation button above it.

Data Source Definition Dialog

The data source definition dialog is used to set the source of a simple data link and allows the inclusion of a list of choices. These choices show in a menu in the goal group interface window when the data link is selected. The Choice list can also be filled in by using the "Args->Choices" menu item in the Edit menu when the data link is highlighted in the object selector.

The source type is not used in NetWeaver because NetWeaver has its own interface for data input. The source type can also be overridden when inputting data to the Ickee inference engine.



Data Source Definition Name: Affected Species

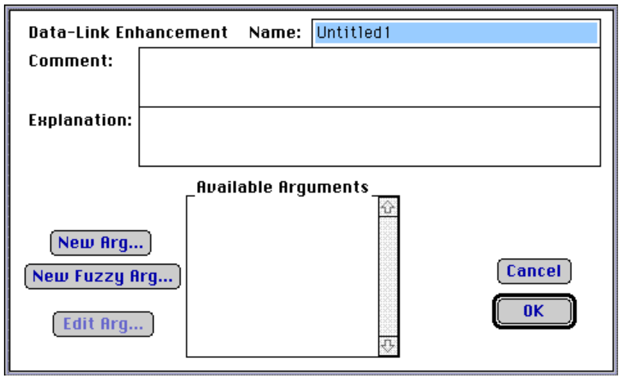
Source Type:
- ○ SQL Query
- ○ GRASS Map Layer
- ○ OS Environment Var
- ○ File
- ○ Structured Flat File
- ○ Function
- ○ Daemon
- [✓ Define at Run Time]

Choice List
Red Oak
White Oak
Red Maple
Black Cherry

Data Link Enhancement Dialog

The data link enhancement dialog is used to set the name, comment, and explanation for a data link and to provide global access to the data link's arguments. Arguments can be created and modified via this dialog. Because of their global availability, this dialog is the only place where arguments can be edited once they are created.

A data link owns a list of arguments that can be shared by argument lists for the data link throughout the project. This feature aids in the economy and maintainability of the project. However, note that the changes made to an argument are propagated throughout the entire project: every reference to that argument will reflect the change. This can be handy if a decision making threshold has changed and yet hazardous, if you only wanted to change one instance. If you don't want to change the argument globally, create a new argument and swap it out in the argument list where you want to use it.


Data-Link Enhancement Name: Untitled1
Comment:
Explanation:

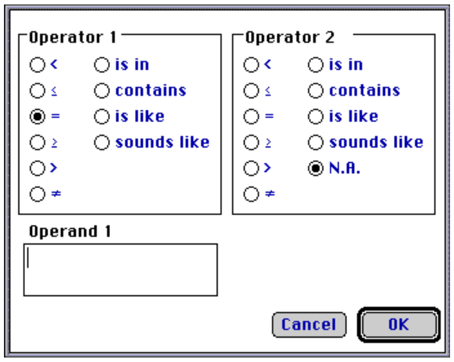
This button creates a new standard argument for the data link and opens the standard argument editor dialog.

This button creates a new fuzzy argument for the data link and opens the fuzzy argument editor dialog.

This button opens the appropriate argument editor dialog for the argument selected in the Available Arguments selector. Read the caveat above before using this feature.

Standard Argument Editor Dialog

The standard argument editor is used to construct and edit standard arguments (as opposed to fuzzy arguments). Arguments are used to convert a data link value into a trueness level. When an argument matches the data link it returns true, when it does not it returns false. The argument can be either one or two part. When an argument is two part it can be read as an "and" connecting the two parts (Temp °F >= 32 AND <= 80 is equivalent to 32 <= temp <= 80). Each part of the argument consists of an operator and an operand. Choose an operator by clicking the appropriate radio button. Fill in an operand in the text edit box. When any radio button except the N.A. button is selected in the Operator 2 group the argument automatically becomes two part and a text edit box appears for the second operand.



Operator 1: ( ) < ( ) ≤ [✓ =] ( ) ≥ ( ) > ( ) ≠ ( ) is in ( ) contains ( ) is like ( ) sounds like

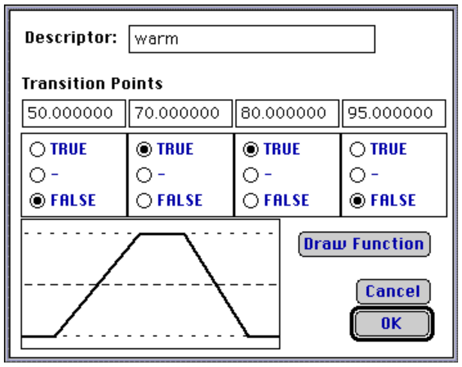
Operator 2: ( ) < ( ) ≤ ( ) = ( ) ≥ ( ) > ( ) ≠ ( ) is in ( ) contains ( ) is like ( ) sounds like [✓ N.A.]

Operand 1:

Cancel OK

Fuzzy Argument Editor Dialog

The fuzzy argument editor is used to a build fuzzy set membership function to be used in calculating a data link's trueness level. The fuzzy argument has a name box and four point definitions. The point definitions are the transition points in the fuzzy set membership function. The argument defined below is for 'warm'. It can be read as: "It is not warm at all below 50° but is starting to become warm. At 70° it is definitely warm. At 80° it is still warm but it is starting to become too hot to be considered warm. At 95° it is too hot to be considered warm any longer."



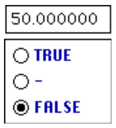
Descriptor: warm

Transition Points

50.000000	70.000000	80.000000	95.000000
( ) TRUE	[✓ TRUE]	[✓ TRUE]	( ) TRUE
( ) -	( ) -	( ) -	( ) -
[✓ FALSE]	( ) FALSE	( ) FALSE	[✓ FALSE]

Draw Function

Cancel OK



50.000000

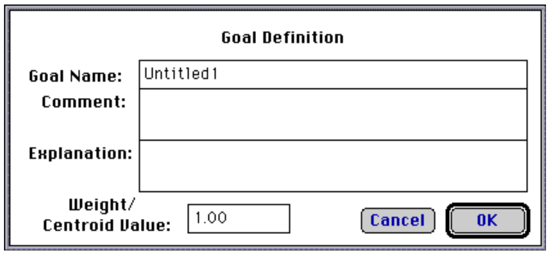
( ) TRUE ( ) - [✓ FALSE]

The point definitions consist of an edit box and a set of radio buttons. Put the transition point in the edit box and indicate whether the function should return true or false at this point. Once the function is defined, click the draw function button to verify.

This button uses the values in the edit boxes to draw a graph of the fuzzy function for visual concurrence.

Goal Definition Dialog

The goal definition dialog is used to set the name of the goal and to optionally apply comments and explanation. The weight of the goal can be set here to indicate the goal's position within the context of the goal group.

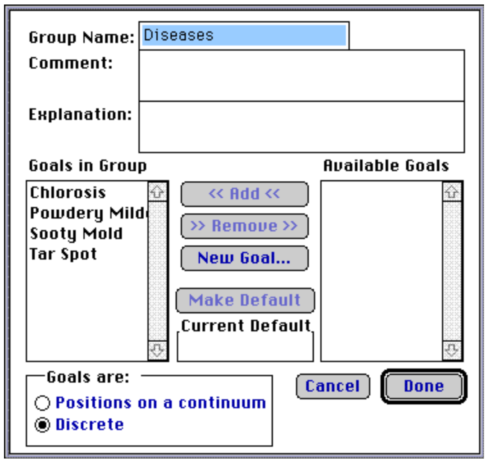


Goal Name: Untitled1
Comment:
Explanation:
Weight/
Centroid Value: 1.00

Goal Group Definition Dialog

The goal group definition dialog is used to set the name of the goal group, the type of group, and to apply comments and explanation and to select the goals to include in the group. A default goal can also be chosen.

The dialog has two object selectors. The selector on the left is a list of the goals already in the goal group. The other selector is a list of all of the rest of the goals in the project. The buttons in the column between the two selectors are used to move the goals from one side to the other and to set an optional default goal.



Group Name: Diseases
Comment:
Explanation:

Goals in Group
Chlorosis
Powdery Mild
Sooty Mold
Tar Spot

Available Goals

Current Default

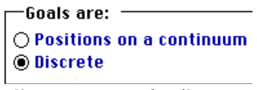
Goals are:
○ Positions on a continuum
● Discrete

This button adds the selected goal to the group list and removes it from the available goals list.

This button removes the selected goal from the group list and places it back into the available goals list.

This button creates a new goal and opens a goal definition dialog for it.

This button makes the selected goal the default goal for the group. The default goal is only chosen when no other valid choice can be made within the group. Having a default is optional.


Goals are:
○ Positions on a continuum
● Discrete

This group of radio buttons toggles the goal group between continuous goals and discrete goals (between quantitative and qualitative goals).

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. The position held by a goal is determined from the weight associated with the goal.

Discrete goals are ones that are unique to themselves and not necessarily related in a symbiotic fashion to each other. An example would be the diseases goal group in the sample dialog above.

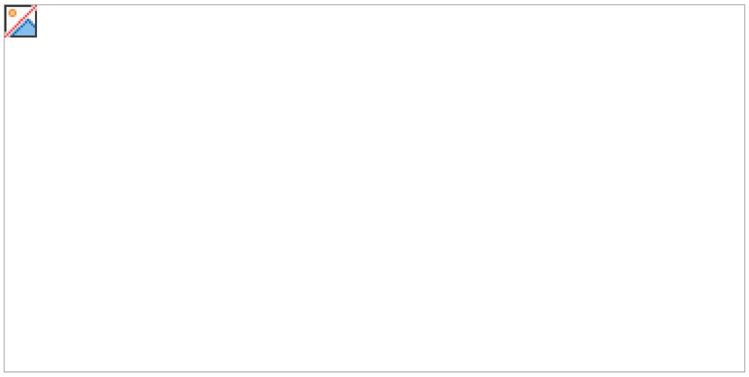
Network Editors

There are two network editors sharing a common interface and mode of operation: a calculated data link editor and a goal dependency editor.

Networks link the fundamental knowledge building blocks, the nodes, into conceptual representations of modular portions of the knowledge base. Nodes are where the real work is done, i.e. operations are performed. Each class of node has its own evaluation method that interprets the incoming data and passes on results appropriate to the function of that node class. Implementation of new functionality is simply a matter of creating a new node class (simple in terms of programming, currently there is no utility for ad hoc class creation).

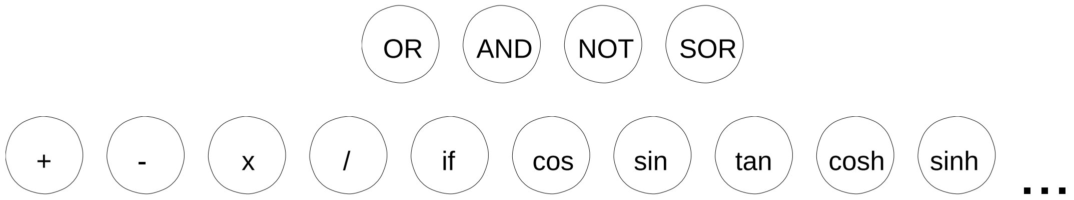
Major objects (goal groups, goals, and data links) are incorporated into the network through the use of proxy nodes. A proxy node's role is to represent its boss node's interests by communicating with it to pass values and messages back and forth between the network and the boss node.

The connections in a network represent paths of communications between nodes with messages being passed both directions, as needed. The two most conceptually important messages passed are the "dirty" message and the "evaluate" message. The "dirty" message is passed from the bottom up, from child to parent and from boss to proxy. Its purpose is to inform the appropriate portions of the network that something has changed so that the next time its status is requested it will perform whatever operation is necessary to "clean itself up". That operation is typically the evaluation operation which is prompted by the "evaluate" message. When the node receives the "evaluate" message from its parent or proxy, it checks to see if it is dirty (needs to be cleaned up) and if so, performs the evaluation routine of its class, otherwise it just passes back its current value that was obtained during a previous evaluation when it was dirty. The purpose of this monitoring of the node's internal status is for performance: A complex system would be brought to its knees if every node had to be evaluated every time anything changed. This method provides efficient husbanding of system resources while maximizing system performance.

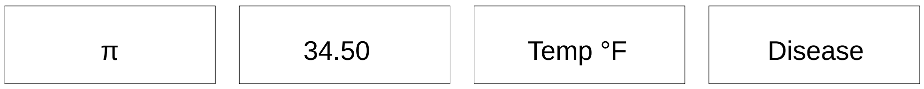


The graphic representation of a node is related to the functionality of the node:

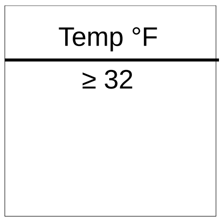
Nodes whose function is to convert data are represented by circles. Included are Boolean (OR, AND, NOT, & SOR) and calculation (+, - , *, ÷, trig, function, etc.) nodes:



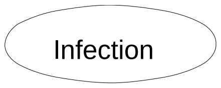
Nodes whose function is to directly transmit data are represented by rounded rectangles. Included are constant nodes, direct references to data links and goal group results:



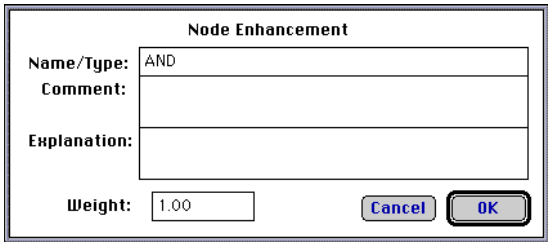
Nodes whose function is to compare a data link to a list of arguments (argument list nodes) are represented by a rectangle with the name of the data link at the top and the arguments listed below.



References to goal networks are represented by ovals:



The following dialog is available to enhance most nodes (all except data link proxy nodes). It is used to control comments, explanation, and weight of the node.


Node Enhancement

Name/Type: AND

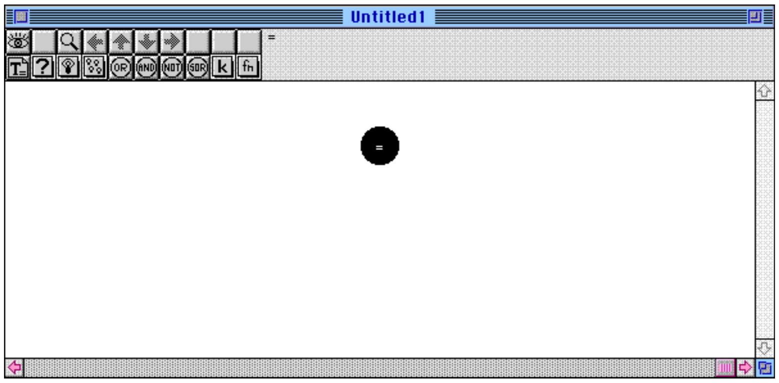
Comment:

Explanation:

Weight: 1.00

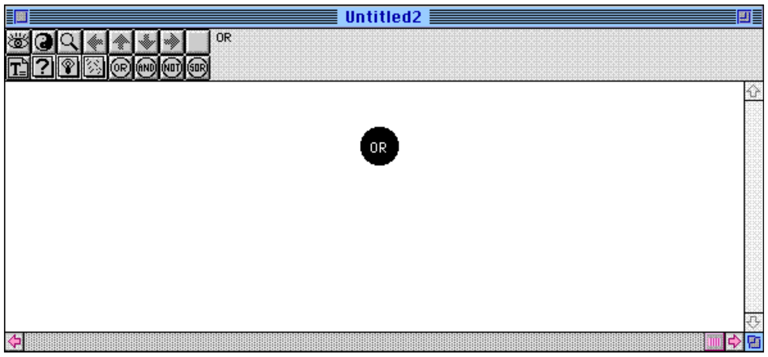
Calculated Data Link Network Editor

The calculated data link network editor is used to build the network that represents the calculations to be performed for determining the value of a calculated data link. By convention the topmost node is always an equals (=) node.

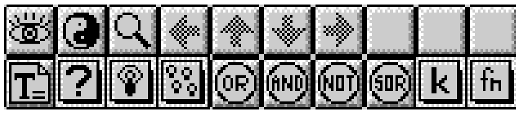


Goal Dependency Network Editor

The goal dependency network editor is used to build the goal's dependency networks. By convention the topmost node is always an OR node.



Network Editor Toolbar

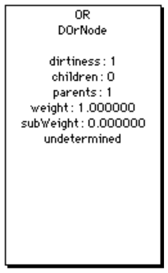


- Viewing/Manipulation Tools
- Object Creation Tools

The toolbar contains two rows of buttons and popup menus. The buttons in the upper row change views or move nodes. The buttons and menus in the lower row create new objects: text boxes and nodes.

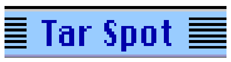
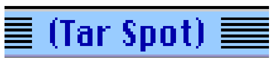
Viewing/Manipulation Tools

"Object Spy". Toggles the object spy function on and off. When the object spy function is on, holding the mouse button down on a node in a network or on a name in an object selector pops open a window that shows internal information about that node. The type of information shown in the window is respective of the node's class type.


OR
DOrNode

dirtiness: 1
children: 0
parents: 1
weight: 1.000000
subWeight: 0.000000
undetermined

"Yin-yang". This button toggles the displayed network between the affirmation network and the negation network of the goal. You can tell which network you are viewing by looking at the title bar at the top of the window. If the negation network is shown, the title is enclosed in parenthesis (accounting notation) while the affirmation network's title is not:

Affirmation network: 
Tar Spot
Negation network: 
(Tar Spot)

"Zoom." Zooms the view out or back in. The close in view is the normal view, but often it is handy to be able to zoom out to see the network in its entirety.

"Bump Left". Bump the currently selected node one position to the left. This button is only active for nodes that have at least one sibling node to left.

"Bump Right". Bump the currently selected node one position to the right. This button is only active for nodes that have at least one sibling node to right.

"Bump Up". Bump the currently selected node back up. This button is only active for nodes that have already been extended down using the Bump Down function.

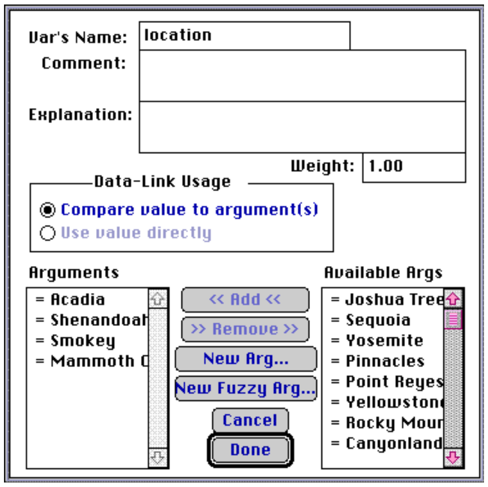
"Bump Down". Bump the currently selected node down. This action inserts a null node to visually move the node down one layer.

Object Creation Tools

"Text tag". Creates a text edit box for documenting the network, much like a PostIt® note. The cursor changes at various locations to indicate the current function of the mouse:

Text insertion. Click and type.
Stretch horizontal. Click and drag to resize.
Stretch vertical. Click and drag to resize.
Stretch random. Click and drag to resize.
Dispose. Click to get rid of the box.
Dragger. Click and drag to move the box to a new position.

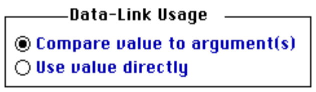
"Data Link Selector". This popup menu provides a list of all of the data links in the project. Only the data links that do not cause circularity are available. The first item in the menu creates a new data link.


Var's Name: location
Comment:
Explanation:
Weight: 1.00
Data-Link Usage
[✓ Compare value to argument(s)]
Use value directly

Arguments	Available Args
= Acadia	= Joshua Tree
= Shenandoah	= Sequoia
= Smokey	= Yosemite
= Mammoth C	= Pinnacles
	= Point Reyes
	= Yellowstone
	= Rocky Mour
	= Canyonland

When a data link is used within a goal dependency network, an argument list node is created to hold a list of arguments to compare against the data link. Once the data link is added to the node a dialog is provided to allow the selection of appropriate arguments for this proxy of the data link. It functions much like the data link enhancement dialog except it has an extra selector and buttons to transfer arguments. Also note that existing arguments can not be edited through this dialog.

When a data link is used within a calculated data link network, its value list can be converted into a floating point number and used directly or it can be compared to a list of arguments as in a goal dependency network.


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

"Goal Selector". This popup menu provides a list of all of the goals in the project. Only the goals that do not cause circularity are available. The first item in the menu creates a new goal. The selected goal is added as a child node to the currently selected node.

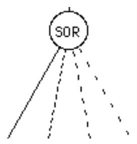
"Goal Group Selector". This popup menu provides a list of all of the goal groups in the project. Only the goal groups that do not cause circularity are available. The first item in the menu creates a new goal group. The selected goal group is added as a child node to the currently selected node.

"OR". Hangs a new OR node from the currently selected node. An OR node is true when any one of its children are true. It is false when all of its children are false. Functionally it passes the value of its most true child using the traditional fuzzy max function.

"AND". Hangs a new AND node from the currently selected node. An AND node is true when all of its children are true. It is false when any one of its children are false. Functionally it performs a weighted average of the values of its children unless one of the children are fully false. It is a fuzzy min function modified to better handle missing data conditions. The weighting factors are pulled from each child node's weight.

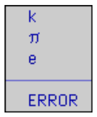
"NOT". Hangs a new NOT node from the currently selected node. A NOT node simply inverts the value of its child.

"Sequential OR". Hangs a new SOR node from the currently selected node. A SOR node is a special class of node designed to select between alternative decision scenarios were there is a definite hierarchy of quality level associated with each possible data gathering method. In other words, the SOR node is a sort of data route selector; it provides a method for selecting the best choice of paths within the scope of the currently given data. Connections to the children of a SOR node are represented with dotted lines to indicate their relative position in the hierarchy:



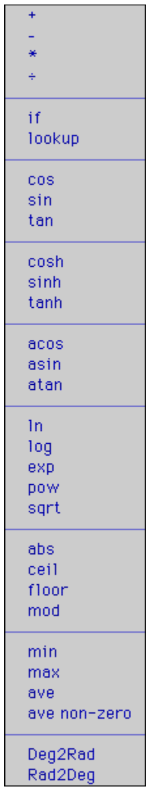
The path selected is the first (left to right) path where the child node's absolute value is equal to or greater than the child node's weight.

"Constant." Hangs a new constant node from the currently selected node. The constant is selected from the popup menu below. If "k" is chosen a dialog is opened where a value can be filled in. This node returns the numeric value of its name.



k	"Constant". Hangs a new constant node from the currently selected node. This node returns the numeric value of its name if its name is numeric, otherwise it returns an error. For example: "4.5" returns 4.50000, "2E4" returns 2.0 x 10^4 , "Bob" returns ERROR.
π	"Pi". Hangs a new Pi node from the currently selected node. This node always returns the value of Pi (3.141592654...).
e	"e". Hangs a new e node from the currently selected node. This node always returns the value of e (2.718281828...).
ERROR	"Forced Error". Hangs a new error node from the currently selected node. This node always returns an error. Its typical use is as one of the arguments of an "if" node, so that the whole calculation network is considered invalid if that branch of the if node is selected. The error node can also be handy for debugging.

"Function". Hangs a new function node from the currently selected node. The function is selected from the popup menu below. Full error trapping is employed. If a function has the wrong number of children or a child node has an illegal value, the function node becomes invalid and the calculation is not used (the data link becomes undetermined by passing an error message up the network).



+ "Addition". Hangs a new addition node from the currently selected node. This node returns the sum of its children.

- "Subtraction or Negation". Hangs a new subtraction node from the currently selected node. If there is only one child, a simple negation is performed. If there are two children, then the right child is subtracted from the left child.

* "Multiplication". Hangs a new multiplication node from the currently selected node. This node returns the value of all of its children multiplied together. If there is only one child, that child's value is passed.

÷ "Division". Hangs a new division node from the currently selected node. This node returns the result of dividing the left node by the right node. If there is only one node or the right node is zero, the node becomes invalid and the calculation is not used.

if	"If". The if node comes in four flavors, dependent upon the number of children it has. The first node to an if node can be a goal.
	"1 node if". Returns 1 if its child is true, 0 otherwise.
	"2 node if". Returns the value of the second child if the first (left) child is true, 0 otherwise.
	"3 node if". Returns the value of the second child if the first child is true, otherwise it returns the value of the third child.
	"4 node if". Returns the value of the second child if the first child is true, the value of the third child if the first child is undetermined and the value of the four child if the first child is false.
lookup	"Lookup". Not yet implemented.
cos	"Cosine". Returns the cosine of the value of its child. The child's value should be in radians.
sin	"Sine". Returns the sine of the value of its child. The child's value should be in radians.
tan	"Tangent". Returns the tangent of the value of its child. The child's value should be in radians.
cosh	"Hyperbolic cosine". Returns the hyperbolic cosine of the value of its child. The child's value should be in radians.
sinh	"Hyperbolic sine". Returns the hyperbolic sine of the value of its child. The child's value should be in radians.
tanh	"Hyperbolic tangent". Returns the hyperbolic tangent of the value of its child. The child's value should be in radians.
acos	"Arc cosine". Returns the principal value, in radians, of the arc cosine ($\cos^{-1}$) for a value of a child in the range of -1 to +1.
asin	"Arc sine". Returns the principal value, in radians, of the arc sine ($\sin^{-1}$) for a value of a child in the range of -1 to +1.
atan	"Arc tangent". Returns the principal value, in radians, of the arc tangent ($\tan^{-1}$) for a value of a child in the range of -1 to +1.
ln	"Natural logarithm" . Returns the natural log (base e logarithm)of its child.
log	"Base 10 logarithm". Returns the base 10 log of its child.
exp	"Exponential: e^x". Returns e raised to the power of the node's child.
pow	"Power: x^y". Returns the left node raised to the power of the right node.
sqrt	"Square root". Returns the square root of the child node.
abs	"Absolute value". Returns the absolute value of the child node.

ceil	"Ceiling". Rounds the child up to next higher integer.
floor	"Floor". Rounds the child down to the next lower integer.
mod	"Modulus". Returns the remainder of the division of the node's right child into the left.
min	"Minimum". Returns the value of the lowest value child.
max	"Maximum. Returns the value of the highest value child.
ave	"Average". Returns the average value of its children.
ave-nz	"Non zero average". Returns the average value of its children which have a non-zero value.
deg2rad	"Degrees to radians". Converts from a value in degrees to one in radians.
rad2deg	"Radians to degrees". Converts form a value in radians to one in degrees.

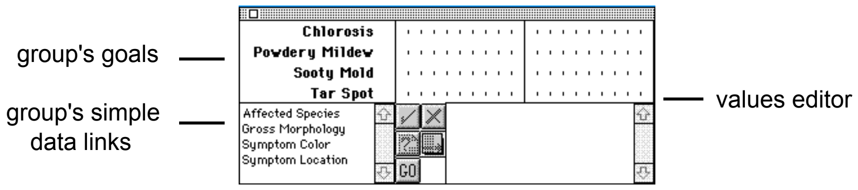
Goal Group Interface Window

The goal group interface window is used to monitor the goal group's status as information is supplied to data links. The goals in the group are graphically shown at the top of the window. The simple data links that can influence the group are listed in a selector in the lower left of the window. The lower right of the window is a values editor where the values of the selected data link can be modified.

The best choice within the group is indicated with an arrow once an evaluation has been made. If the group is of the continuous type, a line is also drawn through the centroid of the group.

The space to the right of the goal names is used to build a bar graph to indicate the goals' trueness levels. The center line represents a zero value (undetermined). Full right represents +1.0 (completely true) and Full left represents -1.0 (completely false). On color capable machines trueness level is also represented by color (red - false, black - undetermined, green - true).

Double clicking on a goal name will open the editor for that goal. Double clicking on a data link name will open the data link enhancement dialog (handy for perusing the data link's arguments).

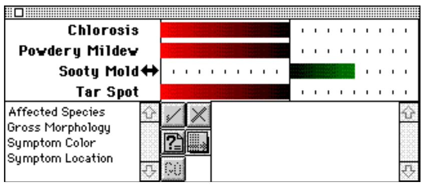


"Accept". Accept the entry and apply it to the data link. The previous values associated with the data link will be overwritten.

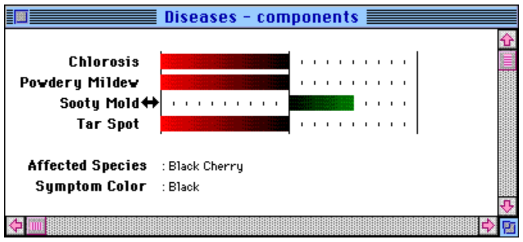
"Cancel". Cancel the entry and reinstate any previous values associated with the data link.

"Choices". A popup menu of choices associated with the selected data link. The choices list can be set using the data source dialog.

"Go". Evaluate the goal group and update the window:



"Components". Creates a new window that shows the current state of all the goals in the group and all intermediate goals and data links that contributed to the evaluation:



Chlorosis
Powdery Mildew
Sooty Mold
Tar Spot

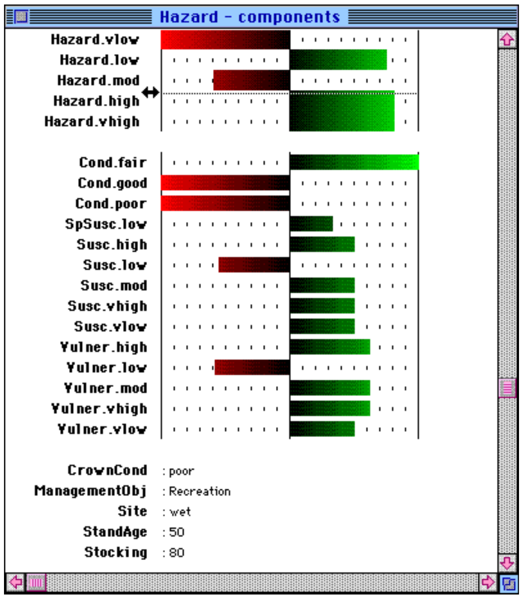
Affected Species : Black Cherry
Symptom Color : Black

Goal Group "Components" Window

The components window is a sort of zoomed in view of the goal group showing the factors that influence the evaluation of the group. It is created dynamically and only shows the major objects that had some bearing on the outcome. Double clicking on a object name opens the appropriate editor for that object.

The objects are listed in the following order. All except the group goals are listed in the window in alphabetic order.

Goals in the group
Intermediate goals
Calculated data links
Simple data links

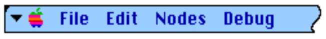


Hazard.vlow
Hazard.low
Hazard.mod
Hazard.high
Hazard.vhigh

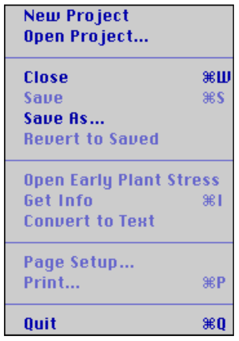
Cond.fair
Cond.good
Cond.poor
SpSusc.low
Susc.high
Susc.low
Susc.mod
Susc.vhigh
Susc.vlow
Vulner.high
Vulner.low
Vulner.mod
Vulner.vhigh
Vulner.vlow

CrownCond : poor
ManagementObj : Recreation
Site : wet
StandAge : 50
Stocking : 80

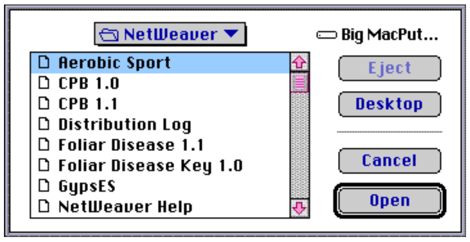
1
The Menus



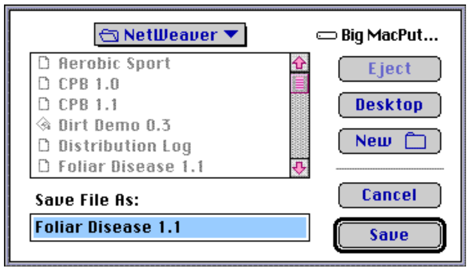
File



New Project	Closes the current project and opens a new, empty project.
Open Project...	Provides a file chooser dialog to select a project file and then opens and builds the project:



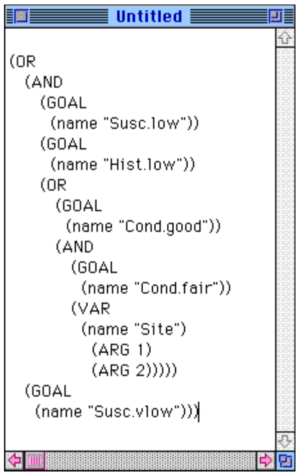
Close	Closes the topmost window.
Save	Saves the project to disk using the current project name.
Save As...	Saves the project to disk using a new file name:



Revert to Saved	Removes all changes made to the project since the project was read from disk and restores it to its original condition.
Open Node	Opens the currently selected node in the project window. This is the same as double clicking on the object name in the project window.
Get Info	Opens the enhancement dialog for the selected node. This is the same as shift double clicking on the object name in the project window. This item also works

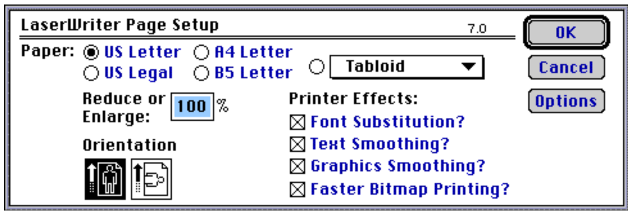
	on nodes within networks.
Convert to Text	Converts the network in the topmost window into a script in a new text

window:



```
(OR
  (AND
    (GOAL
      (name "Susc.low"))
    (GOAL
      (name "Hist.low"))
    (OR
      (GOAL
        (name "Cond.good"))
      (AND
        (GOAL
          (name "Cond.fair"))
        (VAR
          (name "Site")
            (ARG 1)
            (ARG 2)))))
  (GOAL
    (name "Susc.vlow")))
```

Page Setup... Standard Page Setup dialog:



LaserWriter Page Setup 7.0

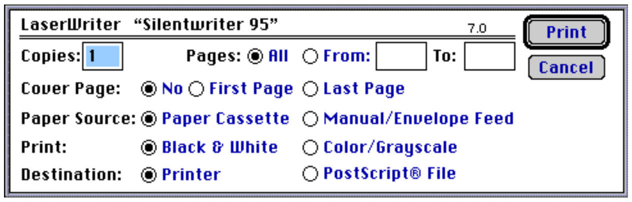
Paper: [✓ US Letter] A4 Letter US Legal B5 Letter Tabloid

Reduce or Enlarge: 100 %

Orientation

Printer Effects:
☒ Font Substitution?
☒ Text Smoothing?
☒ Graphics Smoothing?
☒ Faster Bitmap Printing?

Print... Standard Print dialog:



LaserWriter "Silentwriter 95" 7.0

Copies: 1 Pages: [✓ All] From: To:

Cover Page: [✓ No] First Page Last Page

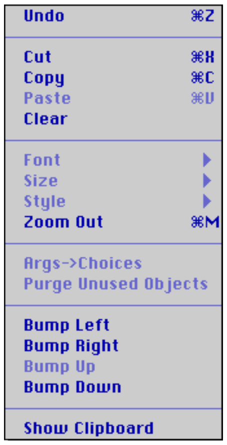
Paper Source: [✓ Paper Cassette] Manual/Envelope Feed

Print: [✓ Black & White] Color/Grayscale

Destination: [✓ Printer] PostScript® File

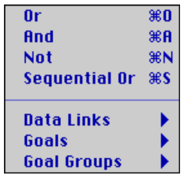
Quit Quit NetWeaver.

Edit



Undo	
Cut	
Copy	
Paste	
Clear	Standard editing functions. Undo is not currently implemented in NetWeaver. Cut, copy, paste, and clear can be used to edit networks.
Font	
Size	
Style	Submenus that provide standard text enhancement functions.
Zoom Out/Zoom In	Toggles between the normal, legible view to a "zoomed out" view.
Args->Choices	Converts a selected simple data link's arguments into a choice list.
Purge Unused Objects	Scans the project for orphaned objects (objects that are not linked to other objects) and prompts the user for each one before disposing the unused object.
Bump Left	Bump the currently selected node one position to the left. This menu item is only active for nodes that have at least one sibling node to left. Same as:
Bump Right	Bump the currently selected node one position to the right. This menu item is only active for nodes that have at least one sibling node to right. Same as:
Bump Up	Bump the currently selected node back up. This menu item is only active for nodes that have already been extended down using the Bump Down button or menu item. Same as:
Bump Down	Bump the currently selected node down. This action inserts a null node to visually move the node down one layer. Same as:
Show Clipboard	Opens a window that shows the contents of the clipboard.

Nodes



Or

Hangs a new OR node from the currently selected node. An OR node is true when any one of its children are true. It is false when all of its children are false. Functionally it passes the value of its most true child using the fuzzy max function. Same as:

And

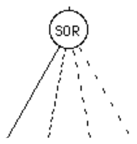
Hangs a new AND node from the currently selected node. An AND node is true when all of its children are true. It is false when any one of its children are false. Functionally it performs a weighted average of the values of its children unless one of the children are fully false. It is a fuzzy min function modified to handle missing data. The weighting factors are pulled from each child node's weight. Same as:

Not

Hangs a new NOT node from the currently selected node. A NOT node simply inverts the value of its child. Same as:

Sequential Or

Hangs a new SOR node from the currently selected node. A SOR node is a special class of node designed to select between alternative decision scenarios were there is a definite hierarchy of quality level associated with each possible data gathering method. In other words, the SOR node is a sort of data route selector; it provides a method for selecting the best choice of paths within the scope of the currently given data. Connections to the children of a SOR node are represented with dotted lines to indicate their relative position in the hierarchy:



The path selected is the first (left to right) path where the node's absolute value is equal to or greater than its weight.

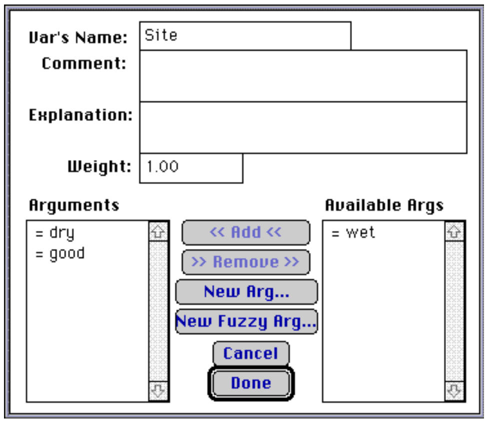
Same as:

Data Links

This submenu provides a list of all of the data links in the project. Only the data links that do not cause circularity are available. The first item in the menu creates a new data link.

When a data link is used within a calculated data link network, its value list is converted into a floating point number and used directly.

When a data link is used within a goal dependency network, an argument list node is created to hold a list of arguments to compare against the data link. Once the data link is added to the node a dialog is provided to allow the selection of appropriate arguments for this instance of the data link:



Var's Name: Site
Comment:
Explanation:
Weight: 1.00

Arguments		Available Args
= dry	<< Add <<	= wet
= good	>> Remove >>	
	New Arg...	
	New Fuzzy Arg...	
	Cancel	
	Done	

Same as:

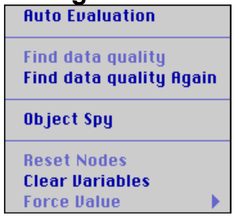
Goals

This submenu provides a list of all of the goals in the project. Only the goals that do not cause circularity are available. The first item in the menu creates a new goal. The selected goal is added as a child node to the currently selected node. Same as:

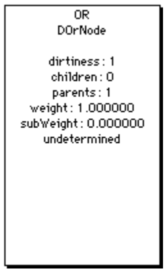
Goal Groups

This popup menu provides a list of all of the goal groups in the project. Only the goal groups that do not cause circularity are available. The first item in the menu creates a new goal group. The selected goal group is added as a child node to the currently selected node. Same as:

Debug



Auto Evaluation	Toggles the system between auto evaluation and evaluation on command. When in auto evaluate mode, every change is automatically acted upon and changed objects updated.
Find	Finds the first instance of the selected object.
Find Again	Finds the next instance of the selected object.
Object Spy	Toggles the object spy function on and off. When the object spy function is on, holding the mouse button down on a node in a network or on a name in an object selector pops open a window that shows internal information about that node. The type of information shown in the window is related to the class type of the node.

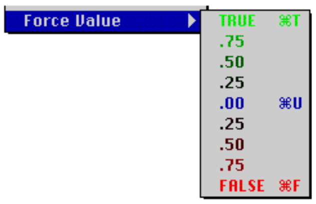


```
OR
DOrNode

dirtiness : 1
children : 0
parents : 1
weight : 1.000000
subWeight : 0.000000
undetermined
```

Same as:

Reset Nodes	Resets all nodes to undetermined.
Clear Variables	Removes all values from simple data links.
Force Values	Applies a value from the submenu to the selected node in a network. This overrides any previous value and ignores the value it should have based on its children.


Force Value
TRUE ⌘T
.75
.50
.25
.00 ⌘U
.25
.50
.75
FALSE ⌘F