

NeticaTM

Application for Belief Networks and Influence Diagrams

User's Guide

Version 1.06 for Windows®

Netica Application

User's Guide

Version 1.06

Copyright © 1996-2003 by Norsys Software Corp.

This document may be copied and stored freely, provided it is duplicated in its entirety, without modification, and including the copyright notice.

Published by:

Norsys Software Corp.
3512 West 23rd Ave.
Vancouver, BC, Canada
V6S 1K5
www.norsys.com

Netica and Norsys are trademarks of Norsys Software Corp.

Microsoft, MS-DOS, Windows and Windows NT are registered trademarks of Microsoft, Inc.

Macintosh is a trademark licensed to Apple Computer, Inc., Balloon Help is a trademark of, and Apple is a registered trademark of Apple Computer, Inc.

UNIX is a registered trademark of AT&T.

IBM, OS/2, and Presentation Manager are registered trademarks, and PowerPC is a trademark of International Business Machines.

Unicode is a trademark of Unicode, Inc.

PostScript is a registered trademark of Adobe Systems, Inc.

Intel is a registered trademark, and Pentium is a trademark of Intel Corporation.

Other brands and product names are trademarks of their respective holders.

While every precaution has been taken in the preparation of this document, we assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

Contents

1 Getting Started	6
1.1 Netica Application	6
1.2 Installation	8
1.3 Menu Commands and Tool Bar	8
1.4 Quick Tour	9
1.4.1 Probabilistic Inference.....	9
1.4.2 Network Construction and Transformation.....	11
1.4.3 Case Files.....	14
1.4.4 Decision Problems.....	14
1.4.5 Equations and Time Expansions	15
1.5 Introductory References for Belief and Decision Networks.....	16
2 Probabilistic Inference	18
2.1 Belief Networks and Probabilistic Inference	18
2.2 Netica's Probabilistic Inference	19
2.3 Example Belief Network.....	20
2.4 Compiling a Belief Network.....	21
2.5 Entering and Retracting Findings	22
2.6 Belief Updating and AutoUpdating	23
2.7 Negative and Likelihood Findings.....	23
2.8 Consistency	24
2.9 Most Probable Explanation.....	25
3 Constructing Belief and Decision Networks	27
3.1 Opening a Window	27
3.2 Adding Nodes	27
3.3 Adding Links	28
3.4 Undoing and Redoing	29
3.5 Selecting Nodes and Links	29
3.6 Moving Nodes and AutoGrid	30
3.7 Reshaping a Link	30
3.8 Saving a Network to a File	31
3.9 Deleting Nodes and Links.....	32
3.10 Disconnecting and Reconnecting Links	32
3.11 Cut, Copy, Paste and Duplicate Nodes	33
3.12 Drawing Size and Scrolling.....	34
4 Changing Node Characteristics	35
4.1 Obtaining the Node Dialog Box	35
4.2 Making Changes	35

4.3 Node Name	36
4.4 Node Title	36
4.5 Node is Discrete or Continuous	36
4.6 Node Kind	37
4.7 Node States	38
4.8 Node State Range.....	38
4.9 Node State Value	39
4.10 Multi-Purpose Box	39
4.10.1 Node Description.....	39
4.10.2 Node Equation	40
4.10.3 Input Name	40
4.10.4 Link Delay	40
4.10.5 Node States.....	40
4.10.6 Node Ranges.....	41
4.10.7 When Changed	42
5 Node Relationships	43
5.1 Obtaining a Relation Dialog Box.....	43
5.2 Meaning of the Tables	43
5.3 Changing Table Entries	44
5.4 Buttons in the Relation Dialog Box.....	45
5.5 Deterministic / Chance Pop-Up	46
5.6 Scrolling and Resizing	46
5.7 Unspecified and Impossible Cell Values	47
5.8 Selecting, Cutting and Pasting	48
5.9 Relation Menu Commands	49
5.10 Multiple Dialogs for One Node.....	50
6 Decision Problems	51
6.1 Constructing a Decision Network.....	52
6.2 No-Forgetting Links	53
6.3 Solving a Decision Network	53
6.4 Model Iteration	54
6.5 What-if Decision Analysis.....	55
7 Network Libraries	56
7.1 Disconnecting and Reconnecting Links	57
7.2 Making and Using Network Libraries	58
8 Transforming a Network	60
8.1 Link Reversals	60
8.2 Node Absorption.....	62
8.3 Probabilistic Inference by Node Absorption	63
9 Cases and Case Files	65
9.1 Multi-Case Files.....	66

9.2 Case File Format	67
9.3 Generating Random Cases.....	68
10 Learning from Cases	70
10.1 How To Do It	71
10.1.1 Single Case Learning.....	71
10.1.2 Learning From a Case File	72
10.2 Experience	72
10.3 Bayesian Learning.....	74
10.4 Fading.....	75
11 Display Style and Printing	77
11.1 Node Name and Title Display	77
11.2 Font and Size.....	78
11.3 Node Styles	78
11.3.1 Belief-Bar	78
11.3.2 Belief-Meter.....	79
11.4 Net Styles	80
11.5 Copying and Pasting Graphics	80
11.6 Printing	81
12 Equations	82
13 Bibliography	83
14 Index	85

1 Getting Started

Welcome to the Netica Application from Norsys. Netica is a versatile, fast, user-friendly program that you can use to find patterns in data, create diagrams encoding knowledge or representing decision problems, use these to answer queries and find optimal decisions, and create probabilistic expert systems. It is suitable for applications in the areas of diagnosis, prediction, decision analysis, sensor fusion, expert system building, reliability analysis, probabilistic modeling, and certain kinds of statistical analysis and data mining.

This guide is for Netica Application; it is not a manual for the Netica Programmer's Library, also known as the "Netica API" (Application Program Interface). The API is a module with much of the same functionality as Netica Application, but designed for programmers to embed in their programs. This guide is for software version 1.06; you can check your Netica software for its version number by choosing "About Netica..." from the Help menu. It is only for the Windows version of Netica Application. If you are using the Macintosh version, there is a separate guide for that.

This guide assumes that you are familiar with using a computer running the Microsoft Windows operating system. It also assumes familiarity with Bayesian belief networks or influence diagrams, although it is very suitable for someone who is just in the process of learning about them. Questions and comments about material in this document may be sent to norsys_info@norsys.com.

1.1 Netica Application

The Netica Application is a comprehensive tool for working with Bayesian belief networks and influence diagrams. It can build, learn, modify, transform and store networks, as well as answer queries or find optimal solutions using its powerful inference engine. Many new features are currently under development, and it will continue to be extended for years to come.

Characteristics:

- Compiles belief (Bayesian) networks into junction trees of cliques for fast probabilistic reasoning. Has an optimizing compiler to generate good junction trees.
- Can learn probabilistic relations from data
- Generates presentation quality graphics which can be transferred to other documents
- Provides easy graphical editing of belief networks and influence diagrams, including:
 - cutting and pasting of nodes and networks without loosing their probabilistic relations
 - many ways of displaying nodes (bar graphs, meters, etc.)
 - links with bends to keep complex diagrams orderly
 - allows comments, keeps track of author, when changed, etc. for each node
 - unlimited levels of undo / redo
- Can find optimal decisions for sequential decision problems (i.e. later decisions are dependent on the results of earlier ones)
- Can reverse individual links and “sum out” nodes of influence diagrams or belief nets, for model exploration
- Supports disconnected links, which makes possible libraries of probabilistic relationships.
- Has facilities to enter and update individual cases, store them in their own files, and apply them to other belief networks.
- Allows the entry of probabilistic relations by equation, with an extensive built-in library of probabilistic functions and other mathematical functions.
- Has facilities for the easy discretization of continuous variables.
- It is possible to represent networks with nodes whose values change over time (a persistence is defined for such nodes), and to have links with time delays (which allows cycles). The software can automatically convert these networks into expanded regular networks covering a limited period of time.
- Supports documentation and tracking of every node and network (with comments, titles, author, when last changed, locking, etc.)
- Has no built in limits on the size or complexity of networks, so they are limited only by available memory.
- Can work hand-in-hand with the Netica API Programmer’s Library (for example, sharing the same files).

1.2 Installation

Before installing or using Netica be sure that you accept the License Agreement which is provided with the software on a separate document. Netica requires a PC with an 80386 or later processor, running Microsoft Windows 95 or Windows NT 4.0. Netica can be installed to consume less than 2 MB of hard disk space.

To install Netica, the first step is to get NeticaZip.exe on your hard disk. If Netica was mailed to you, copy it from the supplied floppy disk, otherwise download it from www.norsys.com. When you double-click it, a folder called Netica will be created, containing Netica.exe, the license agreement, README.txt and a folder of example files. All the files required for Netica are in this folder, and you may place it wherever you want on your hard disk, and move it at any time. You should look at README.txt to see if it contains any special instructions or warnings.

Within this folder is a file called Netica (or Netica.exe), with a large square blue icon. It is the file you double-click to run Netica. You can try running it and doing the operations described in “Quick Tour” below to verify that everything is installed and working properly. The first time you run it, you will be requested to enter a password. Your password is written on the invoice which is mailed to you, and is also sent to you by email if you ordered it over the Internet. If you have two or more different versions of Netica installed at the same time, the default version (the one that runs when you double-click a document icon) will be the one that was run last.

To completely remove Netica from your hard disk, simply drag the folder containing Netica into the recycle bin. Netica does not add any files to any other folders (except, of course, your document .dne files get placed where you request).

1.3 Menu Commands and Tool Bar

In this manual, menu items are indicated in bold, with arrows indicating choices or submenus, so **File → Open** means choose “Open” from the “File” menu. Most menu items have a corresponding tool bar button, and you can discover what the buttons do by resting the cursor on them briefly. Some of the tool bar buttons discussed in this manual may not appear on the tool bar when you first start Netica. Simply double-click on a blank area of the toolbar, or choose **Window → Customize Toolbar**, and use the resulting dialog box to add any desired buttons. Once they are on the tool bar, buttons can be rearranged or removed by holding down the Alt key while dragging them.

1.4 Quick Tour

All you need to know to complete this tour is how to use a Windows computer. You do not need knowledge of belief networks, decision networks or Netica, although you may not fully understand what is happening until you have that knowledge, and have read other chapters in this guide.

First make sure you have completed the above described installation. You should duplicate the “Examples” folder in case you accidentally change some of the supplied files. Then run Netica by double-clicking on its icon. The first time you run Netica it will ask you for a password. Enter the license password from your invoice, or click on the “Reduced Mode” button if you don’t have one.

The workspace window will open, and in its lower left corner there will be an icon for a minimized window called “Netica Messages”. Usually when you try to do an operation that Netica can’t perform, it will display a dialog box telling you. However, for some minor operations, like clicking in the wrong place, Netica will just beep. In that case, you can look in the Netica Messages window and there will be an explanation. You can copy and paste information between the Messages window and any text file.

When you are finished with your Netica session, you can end it by doing a **File → Exit** menu command. If you have created or changed some work without saving it, Netica will ask if you want to save it before terminating.

1.4.1 Probabilistic Inference

First we will read a belief network stored on disk. Do a **File → Open** menu command, and when the standard file-opening dialog box appears, use it to open the file called “Asia” in the folder “Examples”. A window will appear containing a belief network consisting of several connected “nodes”. This network is a classic very simplified example network for diagnosing patients arriving at a clinic.

You can prepare the network for inference with a **Network → Compile** menu command, or by clicking on the lightening bolt tool bar button. When the compilation is complete, the default display style changes so that nodes are displayed with bar graphs showing the beliefs for each of their states. You can enter a finding (also known as an “observation”, or “evidence”) for a node by clicking on the name of the finding to the left of the bar graph. You can also enter a finding by right-clicking on the node and choosing it from the pop-up menu that appears. When the finding is entered, the displays of all the nodes will be adjusted to account for it. For instance, putting an ‘abnormal’ finding for ‘XRay Result’ node increases the belief that the patient has

lung cancer from 5.5% to 48.9%, but then indicating that the patient has made a visit to Asia decreases that belief to 37.1%, because the abnormal XRay is partially *explained away* by a greater chance of Tuberculosis (which the patient could catch in Asia). If you choose “Unknown” from the findings pop-up menu, then any finding for that node will be retracted, and if you choose “Likelihood” you will be queried for the probabilities of an uncertain finding (“virtual evidence”) for the node. You can also click directly on the name of the finding a second time to retract the finding, or hold down the shift key while clicking on the name of the finding to enter a negative finding.

Each time you enter a finding, or retract one, the beliefs at all the nodes will immediately be updated to account for the new information. If you wish updating to only occur when you do a **Network → Update** command, you can turn off the auto-update feature of the network by choosing **Network → Automatic Updating** from the menu (the menu item will be checked only when auto-updating is on).

If you want to observe or change the conditional probabilities of a node (which express its relation with its parent nodes), select the node by clicking once on it, and then do a **Relation → View / Edit** command, or click on the tool bar button with the green inverted V’s. A special window called the *relation* dialog box will open which displays the probabilities in a table. The left-hand side of the table contains a vertical list of parent configurations, and for each configuration the right-hand side has a few probabilities, expressed as percentages. Each column of the right-hand side corresponds to a different state of the node. So each number represents the conditional probability that the node takes on the state indicated by the column the number is in, given that the parents have the configuration indicated by the row the number is in. If the node is deterministic (e.g. the “TbOrCa” node), then the relation dialog box will display a function table. This is the same as the conditional probability table, except that the right-hand side simply has the state of the node which is the function value for that parent configuration.

To work with an example of a more complex belief network (also in the medical domain), open the file called “Alarm”. You can compile it and do inference in the same way as you did with Asia. If you have entered a number of findings for a particular case and wish to save them in their own file, do a **File → Save Case As** command and enter the file name in the dialog box which appears. If you want to work on a new case do **Network → Remove Findings**, and then enter the new findings. To recover the original case, do **File → Get Case** and choose its name from the dialog box which appears.

When you are done with a network window, you can get rid of it by clicking in its X box at the upper right, or by making it the active window and then doing **File → Close**.

1.4.2 Network Construction and Transformation

Do **File** → **New** to create a new belief or decision network window. To add a nature node (i.e. a “chance node” or “deterministic node”), move the cursor to the tool bar and click on the button with the yellow ellipse. When you return the cursor to the new window, it will change to an ellipse, and when you click in the window, a node will be added at the cursor. The node will be selected when first added (i.e. displayed with inverted colors), so by pressing the <Enter> key a dialog box will be opened for changing node attributes. You can use this to enter the name, title, etc. of the node.

Utility nodes (also known as “value nodes”) or decision nodes may be added in the same way as nature nodes, by using the green hexagon or blue rectangle tool buttons respectively. Links may be added by clicking on the tool button with the downward pointing arrow (not the upward arrow), then clicking on the node you want the link to come from (i.e. the “parent” node), and finally clicking on the node you want it to go to (i.e. the “child” node). Alternately, you can click down in the parent node, and while holding the mouse button down, drag the cursor to the child node, and then release it. If you double-click on a tool bar button then it will create a cursor that you can use to add several nodes or links (without it switching back to the pointer each time).

To select a node or link, click once on it with the regular pointer cursor, and it will become highlighted. A group of nodes may be selected by clicking down on the background within the window, and then moving the mouse with the button depressed (i.e. “dragging”), so that the selection rectangle is over them. You can add to or remove from a group of selected nodes by holding down the shift key while you select the new nodes. To delete nodes or links, select them and then press the <delete> key.

After doing any operation, you can “undo” it with **Edit** → **Undo** (or pressing Ctrl+Z), and by repeating this you can undo operations previous to that one (at least 4 operations, and sometimes more if they don’t take much memory). After undoing one or more operations you can redo them one-by-one with **Edit** → **Redo** (or pressing Ctrl+Shift+Z). When exploring with Netica, it is very useful to be able to try a few operations, and then easily undo them.

A node can be moved by clicking down on it, dragging it to its new position, and then releasing the mouse button. To move a group of nodes, first select them, and then click down on one of the selected nodes and drag it to its new position. To change the shape of a link, first select it by clicking on it, then click down on it again and drag the cursor to the point where you want the link bend to be.

A link can be disconnected by clicking on it to select it, then clicking down on the hilited square that forms at its non-arrow end, and dragging it away from the parent node (or by selecting the link and doing **Modify**→**Disconnect Links**). To reconnect the link to a new node, drag the non-arrow end over the new parent and release the mouse button. Disconnection / reconnection is useful to change the links of a network without loosing the conditional probability relations of the nodes. Remember that to just delete a link you simply select it and then press the <delete> key.

You can cut and paste nodes and subnetworks within a window, or between windows. Try opening a network such as Car_Diagnosis_0, select part of it, press Ctrl+C (to copy it to the clipboard), Ctrl+N (to create a new network), click in the middle of the new window (to indicate where to put it), and then Ctrl+V (to paste it into the new network). You can cut and paste other parts of the network, or other networks, add nodes, etc. and then connect up the disconnected links as described in the previous paragraph. In this way you can take knowledge from previous applications, perhaps save it in a “network library”, and re-use it to construct a new network for a new application.

Of course, at any point you can do a **File**→**Save** command to save the current version of the network to file, overwriting the previous one, or a **File**→**Save As** command to save it to a new file.

You can change the characteristics of a node by using a “node dialog box”, which is obtained by double-clicking on the node. When you make changes in the dialog box, they won’t actually be applied to the node until you click the “Apply” or “Okay” buttons. It is possible to open dialog boxes for a number of nodes at the same time, and even to open more than one box for the same node. There is a whole chapter devoted to using these dialog boxes, so here we will just mention a few things. You can change a node’s name or title in the obvious way by typing in the appropriate text field of the dialog box. To change the name of one of the node’s states, choose the state using the selector beside the “State:” label, and then type in the neighboring text field. You can add or delete states with the “New” and “Delete” buttons. Most people find it more convenient to enter or change state names using the text entry area at the bottom of the dialog.

The text entry area at the bottom of the dialog can be used to enter or change several different things; you choose the thing to work on with the selector directly above it. For instance, to enter some text documenting the node, you choose “Description” from the menu, and then type it in the text entry box. Some choices (such as “When Changed”) can’t be modified, and are for viewing only.

In order to change the probabilities of a node conditioned on the values of its parents, first open the node’s “relation dialog box”, as described in the previous section. You can click on a

number (or state name if it's a deterministic node) to change it. If no probabilities have yet been entered for this node, you can click on an empty cell to enter a number. The numbers must be entered as percentages. You can select some cells by clicking down in a cell that is not currently being edited, and dragging to enclose the cells before releasing the mouse button. You can select whole rows at a time by clicking and dragging slightly to the right of the double vertical line (this is how you must select deterministic cells, since clicking directly on them brings up the state menu). There are several commands in the **Relation** menu that you can use to set the values of selected cells (such as **Uniform Probabilities**, **Fill Missing**, **Normalize** and **Randomize**). Any changes you make in the dialog box will not actually be transferred to the node until you press the Apply or Okay button. These two buttons do the same thing, but the Okay button also removes the dialog box.

Node *absorption* removes a node without effecting the overall global relationship of the rest of the nodes (i.e. the joint probability distribution). Here is a small demonstration of node absorption, showing how it doesn't effect the beliefs of the rest of the network. Open "Car_Diagnosis_2". Compile it with **Network → Compile**, and notice that the belief is 25.4% that "Spark Quality" is "good". If you enter a finding of "dim" for node "Headlights", the belief changes to 1.47%. Now select nodes "Main Fuse", "Battery Age", "Voltage at Plug" and "Spark Plugs", and then click on the star tool bar button or do a **Modify → Absorb Nodes** command. The selected nodes will be absorbed, links will be added and removed, and probability tables adjusted to maintain the global relationship. Now do **Network → Compile**, and observe the belief is 1.47% that "Spark Quality" is "good" as it was before the absorption. If you then do **Network → Remove Findings**, the belief changes to 25.4%, which is what it was in the old network before any findings were entered.

Link reversal changes the direction of a link without effecting the overall global relationship of the rest of the nodes (i.e. the joint probability distribution). Select a link and then click on the double-arrow tool bar button or do a **Modify → Reverse Links** command. Netica may have to add extra links in order to maintain the joint probability distribution (or it may be able to remove some links).

To change the display style of some nodes, first select them and then make a choice from the **Style → Node** submenu. For the Asia network the most useful displays are belief-bar or belief-meter, although if you were creating a network for an end-user you may want to hide a node like TbOrCa, because it is not of interest, so you would give it a style of Hidden. If you want to hide all the links, you can use **Style → Net → Hide Links**. The **Style → Font** and **Style → Size** submenus may be used to change the font and size of all the nodes at once.

1.4.3 Case Files

Open the belief network called “Car_Diagnosis_0”. It is a simplified example network containing nodes for a few variables of interest when diagnosing a car that is not running. The nodes are linked up in a causal manner, but the network does not contain any information other than the node names, their states, and how they are linked. You can use a relation dialog box to examine a few nodes to verify that they have no probability tables.

To learn a probabilistic relation for each of the nodes we will use a file of fictitious cases of cars previously arriving at a garage, called “Car Cases”. You may want to examine this file with a text editor. The row of headings across the top are the names of the nodes in the network, and each possible value they can take are the state names of those nodes. “BatAge” is a continuous node, so its values are real numbers. The asterisks (*) indicate data values which are not known (i.e. “missing data”).

Make sure that the network window is the active window and that no nodes of the belief network are selected (otherwise the learning will only apply to the selected nodes), and then do **Relation → Incorp Case File**. When you are queried for a file, choose Car Cases, and enter 1 for the degree. Netica will use the cases to learn probabilistic relations for each node of the network, with the Message window displaying the fraction completed.

When it is finished, if you examine a few nodes with the node relation dialog, you will see the learned probability distributions. You can compile the network and do inference, paste parts of it into a decision network, absorb nodes, etc.

Netica can also be used to generate files of cases which follow the probability distribution of a belief network (i.e. sample from the belief network). These cases can be used as realistic examples of possible scenarios, or as synthetic data for learning experiments. Simply select those nodes of the network for which you want columns in the case file, and then do **Network → Simulate Cases**. You will be queried for the number of cases to generate, the name of the file to create, and how much missing data you want (enter 0 for none, 1 for all missing).

1.4.4 Decision Problems

Use **File → Open** to read the decision network (also known as an influence diagram) called “Umbrella”. By opening relation dialogs for the nature nodes (yellow ovals, also known as chance nodes) you can observe their conditional probability relations as described in the first section of this Quick Tour. By opening a relation dialog for the utility node (green hexagon, also known as a value node), you can see its utility function. If you look at the relation dialog for the decision node (blue rectangle), you will see that the node does not yet have any functional

relation, which indicates that there is no decision function associated with the node. Our goal is to find a decision function which will maximize the expected value of the utility node.

This is accomplished by doing **Network → Optimize Decisions**. After doing this command, when you look at the relation dialog for the decision node, it will display the discovered decision function (you will have to click the “Load” button if you didn’t freshly open the dialog). Absorbing all the nodes and then opening the relation dialog for the utility node will show the expected utility. You can then use an **Undo** command to return the network to its original state.

If you want to try another influence diagram, open “Car_Buyer_Neapolitan”, which is described in Neapolitan90, and is based on the classic “Car Buyer” influence diagram. It involves two sequential decisions about whether to do some tests and then whether to buy a certain used car. The optimal decisions turn out to be not to do any tests ($D = \text{none}$), but to buy the car ($B = \text{Buy}$ when $D = \text{none}$).

1.4.5 Equations and Time Expansions

Open the belief network called “Bouncing”. It is designed to model a ball which bounces back and forth in a straight line between two parallel barriers without loosing any energy at each bounce. Initially we have no knowledge of the position or velocity of the ball.

The Position and Velocity nodes stand for the position and velocity at each instance of time, and are connected together with brown *time delay links* as well as a regular black link. If you double-click on the Position node you can see the equation which defines the new position in terms of the old position, and the velocity. It involves the constant dt , whose value may be changed (by right-clicking on the dt node) to provide a different amount of time between time slices. By clicking down on the selector that originally says “Equation”, you can choose “Ranges” to see how the node has been discretized. Later, you can edit this list to change the discretization, by changing the numbers, or adding new numbers.

The first step is to generate probability tables for the current discretization. Select the Position and Velocity nodes and do a **Relation → Equation to Table** menu command. Enter 100 when prompted for the number of samples per cell. Then do a **Relation → Harden** menu command, and press <Enter> to accept the default value for hardening.

Next you generate a time expanded version of the network by doing a **Network → Expand Time** menu command (you can enter 7 when it asks you for the total time). This will make a new window with a new network in it. You will probably want to make this window wider, in the usual way, by resizing it. The new network is a regular belief network with each Position and

Velocity node representing the position and velocity at a new point in time, with nodes to the right corresponding to later times.

There are 3 disconnected links on the two first nodes. We have no initial knowledge of the position or velocity, so we delete them by clicking on each link to select it and pressing the <delete> key. Then select the first two nodes and do a **Relation → Uniform Probabilities** menu command. This will give the first two nodes uniform probability distributions, indicating we don't know anything about the initial position or velocity.

Finally we can compile the belief network for probabilistic inference with a **Network → Compile** menu command. Turn on automatic updating, if it isn't already, by choosing **Network → Automatic Updating** so the menu item is check marked. Experiment with setting the position or velocity (by clicking on the desired range) to indicate observations at certain times, and see how the beliefs for position and velocity at the rest of the times change. An interesting situation is when the ball is near the boundary, but you don't know which way it is going, since after a little while it will surely be moving away from the boundary (perhaps due to a bounce). Or if the positions at 2 times are known, can it infer the velocity? Or if just the velocities at two adjacent times are known, and one is the negative of the other, can it infer the position (because it must have bounced). You can also try entering some negative findings (that the ball isn't at some particular position or velocity) by holding down the shift key when you click on the range. Remember that the numerical results will not be exact, due to the discretization and sampling error in converting the equations to probability tables. You may also want to try a finer discretization by changing the Ranges.

That ends our quick tour of Netica. Now you will probably want to explore some aspect of Netica in more detail by reading the appropriate chapter. If you wish to learn more about belief networks or influence diagrams, the next section should get you started.

1.5 Introductory References for Belief and Decision Networks

For an overview of belief and decision networks, there are two special journal issues that are suitable. The first is the winter 1991 edition of *AI Magazine*, which contains "Bayesian Networks Without Tears" (Charniak91) and "Decision Analysis and Expert Systems" (Henrion&BH91). The other is volume 38 of the *Communications of the ACM*, which is devoted to belief and decision networks (Heckerman&MW95), and has introductory material and descriptions of real applications. Complete references for all the books and papers discussed here can be found in the bibliography.

For a very good, concise, but in-depth, examination of belief networks, the theory behind them, and the algorithms Netica uses for inference, read Spiegelhalter&DLC93. It concentrates on medical diagnosis, but the ideas are widely applicable. It examines some advanced aspects of the theory and algorithms, but is okay for a beginner, who can just skip a few of the sections.

Although much of the literature on belief networks is published within the artificial intelligence (AI) community, it is really the combined work of the statistics/probability, decision analysis, operations research, and AI communities. Many of the researchers who developed the theory of belief and decision networks attended the annual “Uncertainty in Artificial Intelligence” conference from 1985 to present, and so advanced material can be found in the conference proceedings (published by North-Holland and Morgan Kaufmann).

In 1988 Judea Pearl wrote *Probabilistic Reasoning in Intelligent Systems*, which was the most influential and widely cited book in the development of belief networks. It is an excellent foundational book, but it doesn’t contain the latest developments, and has a very mathematical and theoretical orientation.

Then, in 1990 Richard Neapolitan wrote *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, which is a little more modern and easier to read, but is limited in scope and is most useful to someone building a probabilistic reasoner from the ground up, or better yet, teaching a course on how to do that. It is suitable if you want to spend some serious time studying to understand many of the algorithms used internally in Netica; for a quick study read Spiegelhalter&DLC93.

Russell&N95 is a rather complete (and excellent) introductory textbook for artificial intelligence that does a good job of describing Bayes networks, decision networks, dynamic decision networks, policy iteration, etc. within the overall context of intelligent agents and AI software.

2 Probabilistic Inference

2.1 Belief Networks and Probabilistic Inference

A belief network (also known as a Bayesian network or probabilistic causal network) captures believed relations (which may be uncertain, stochastic, or imprecise) between a set of variables which are relevant to some problem. They might be relevant because we will be able to observe them, because we need to know their value to take some action or report some result, or because they are intermediate or internal variables that help us express the relationships between the rest of the variables.

A classic example of the use of belief networks is in the medical domain. Here each new patient typically corresponds to a new “case”, and the problem is to diagnose the patient (i.e. find beliefs for the unmeasurable disease variables), predict what is going to happen to the patient, or find an optimal prescription, given the values of observable variables (symptoms). A doctor may be the expert used to define the structure of the network, and provide the initial relations between variables (often in the form of conditional probabilities), based on his medical training and experience with previous cases. Then the network probabilities may be fine-tuned by using statistics from previous cases, and from new cases as they arrive.

When the belief network is constructed, one *node* is used for each scalar variable, which may be discrete, continuous, or propositional (true/false). Because of this, the words “node” and “variable” are used interchangeably throughout this document, but “variable” usually refers to the real world or the original problem, while “node” usually refers to its representation within the belief network.

The nodes are then connected up with directed *links*. If there is a link from node A to node B, then node A is sometimes called the *parent*, and node B the *child* (of course, B could be the parent of another node). Usually a link from node A to node B indicates that A causes B, that A partially causes or predisposes B, that B is an imperfect observation of A, that A and B are

functionally related, or that A and B are statistically correlated. The precise definition of a link is based on conditional independence, and is explained in detail in an introductory work like Neapolitan90 or Pearl88. However, most people seem to intuitively grasp the notion of links, and use them effectively without concentrating on the precise definition.

Finally, probabilistic relations are provided for each node, which express the probabilities of that node taking on each of its values, conditioned on the values of its parent nodes. Some nodes may have a deterministic relation, which means that the value of the node is given as a direct function of the parent node values.

After the belief network is constructed, it may be applied to a particular case. For each variable we know the value of, we enter that value into its node as a *finding* (also known as “evidence”). Then Netica does *probabilistic inference* to find *beliefs* for all the other variables. Suppose one of the nodes corresponds to the variable “temperature”, and it can take on the values cold, medium and hot. Then an example belief for temperature could be: [cold - 0.1, medium - 0.6, hot - 0.3], indicating the subjective probabilities that the temperature is cold, medium or hot. The final beliefs are sometimes called *posterior probabilities* (with *prior probabilities* being the probabilities before any findings were entered). Probabilistic inference done using a belief network is called *belief updating*.

If we want to apply the network to a different case, then all the findings can be retracted, new findings entered, and belief updating repeated to find new beliefs for all the nodes.

Probabilistic inference only results in a set of beliefs at each node; it does not change the network (knowledge base) at all. If the findings that have been entered are a true example that might give some indication of cases which will be seen in the future, you may think that they should change the knowledge base a little bit as well, so that next time it is used, its conditional probabilities more accurately reflect the real world. To achieve this you would also do *probability revision*, which is described in chapter 10 “Learning from Cases”.

2.2 Netica’s Probabilistic Inference

Netica uses the fastest known algorithm for exact general probabilistic inference in a compiled belief network, which is message passing in a *join tree* (or “junction tree”) of cliques. The algorithms used are described in SpiegelhalterDLC93 and Neapolitan90.

In this process Netica first “compiles” the belief network into a join tree. The join tree is implemented as a large set of data structures connected up with the original belief network, but invisible to you as a user of Netica. You enter findings for one or more nodes of the original belief network, and then when you want to know the resultant beliefs for some of the other

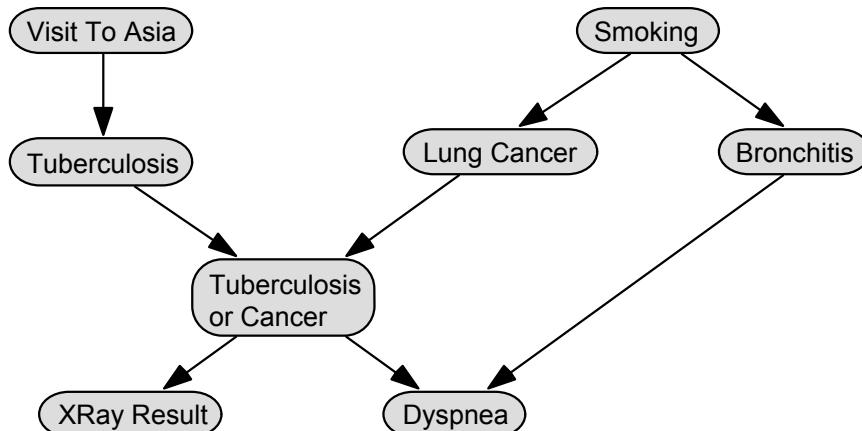
nodes, belief updating is done by a message-passing algorithm operating on the underlying join tree. It determines the resultant beliefs for each of the nodes of the original belief network, which it displays as a bar graph, or a needle meter, at each node. You may then enter some more findings (to be added to the first), or remove some findings, and when you request the resultant beliefs, belief updating will be performed again to take the new findings into account.

Netica can also do probabilistic inference by link reversals and node absorption. For most applications you will want to use the compiled join trees, because usually they are much faster, and don't destroy the existing network, but you may want to use node absorptions when you have some findings that are going to be repeated in many inferences (e.g. if you discover that something is always true in the context of interest). This section deals with join trees; see chapter 8 for information on node absorption.

2.3 Example Belief Network

Now let's look at an example of using Netica to do probabilistic inference. In this example we will read in a simple network from a file, compile it into a form suitable for fast inference, enter some findings, and see how the beliefs of various nodes change with each finding.

The network we will use is shown below. Although reasonable, it is a toy medical diagnosis example from LauritzenSpiegelhalter88 that has often been used in the past for demonstration purposes. To a certain degree, the links of the network correspond to causation. The two top nodes are “predispositions” which influence the likelihood of the diseases in the row below them. At the bottom are symptoms for the disease.

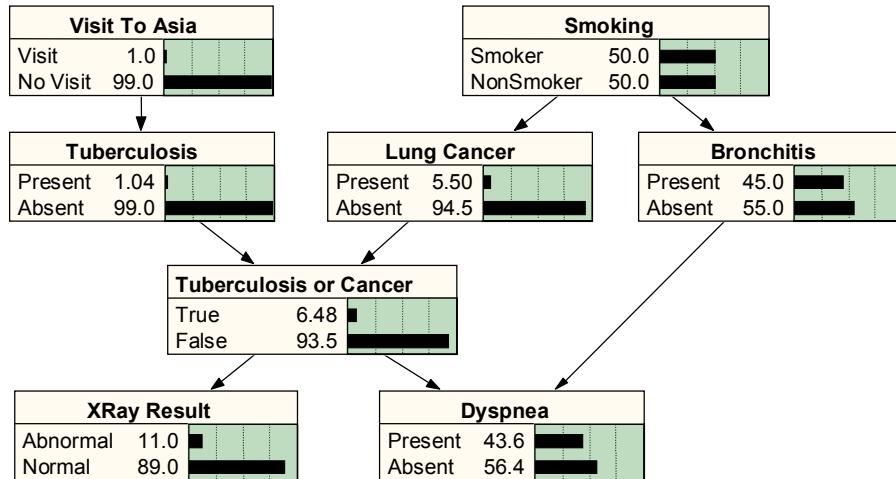


You can read it in to Netica by doing a **File → Open** menu command (in this document, menu selections are indicated in bold, with arrows indicating choices or submenus, so the previous means choose “Open...” from the “File” menu). When the standard file opening dialog box

appears, use it to open the file called “Asia”. This file is included with the Netica distribution in the “Examples” folder.

2.4 Compiling a Belief Network

To compile a belief network, first make sure it is the active window (i.e. the one in front with the darkened title bar). If it isn’t, you can make it active by clicking on an exposed part of it, or choosing its title from the **Windows** menu. Then do a **Network → Compile** menu command, or click on the lightening bolt tool bar button. The appropriate data structures for fast inference will be built internally. The default display style for the nodes will be changed to belief-bar, so if the nodes haven’t had their display styles set (see chapter 11 “Display Style and Printing”), then they will be displayed in belief-bar style. If you compile “Asia”, you will see something like the following:



Each state is shown with its belief level (probability) expressed as a percentage and as a bar graph (see chapter 11 for more details).

For small networks, the **Compile** command will seem to perform instantly. For large networks it may take a few seconds, in which case the progress will be displayed in the **Messages** window. Netica also has a **Network → Compile Optimize** command, which spends a long time working out a very efficient structure for the internal join tree. As it is working, it displays in the **Messages** window the memory requirements and projected inference time to do a belief updating with the best structure it has developed so far. The memory requirements are exact, but the inference time is only approximate (it is based on doing the inference using the same computer as the one doing the compiling, on having enough memory installed that it doesn’t need to use virtual memory, and on having a large share of the processor time under multitasking). Press the

Ctrl key and hold down the left mouse button at the same time when you are satisfied with the structure it has found, and you want it to stop trying to find a better one.

For those interested in a technical explanation, the quick compile does a minimum-weight search for a good elimination order, and the optimized compile searches for the best elimination order using a specialized algorithm which is a combination of minimum-weight search and stochastic search (with some facets of multi-start methods, simulated annealing and genetic algorithms).

If you save a compiled network to file, then when you read it back the next time you will have to recompile it. However, you may still want to save a network after compiling it, since some information gets saved with the network to make compiling go faster the next time it is read back. If you save a belief network after an optimized compile, then the next time you open it the quick compile command will give it the optimized structure quickly (as long as you don't change the network nodes or links in between).

2.5 Entering and Retracting Findings

A compiled belief network is ready to apply to a new case. As details of the case arrive, you can enter them as findings (also known as "evidence"). Of course you would not mix findings from one case with those of another.

When you know the value for some node you can enter it as a finding by clicking on it with the right mouse button. A pop-up menu will appear, known as the "findings pop-up menu", with each possible state of the node as a menu entry. Using it you can enter the finding. A faster method, which works when the network is compiled and the node is displayed in belief-bar or belief-meter style, is to do a regular click directly on the name of the finding you wish to enter.

After the finding has been entered the node will be darkened. If the node is displayed in belief-bar style then a solid outlined bar is drawn for the state matching the finding. It does not matter in what order you enter a set of findings.

If you wish to retract a finding that you have entered for some node, choose "Unknown" from its findings pop-up menu, or click again directly on the name of the finding of a belief-bar node. Entering a finding and retracting it is equivalent to never having entered it, even if there were other findings entered in between. Another way of retracting the findings for some nodes is to select them (see "Selecting Nodes and Links") and then do a **Network → Remove Findings** menu command, or click on the tool bar button with a red cross over the "case symbol" (yellow rectangle with black dot-lines). If you wish to retract all the findings entered for all the nodes in the network, make sure all or none of them are selected, and then do a **Network →**

Remove Findings menu command. This is useful if you are finished with one case and wish to move on to another.

2.6 Belief Updating and AutoUpdating

If you have been experimenting with entering findings into the “Asia” belief network, you will have noticed that belief updating is performed immediately after entering each finding, resulting in a new setting of belief-bars for each node in the network. That is fine for a small network like Asia, since the belief updating occurs very quickly. For a large network the belief updating will take longer, so you may want the ability to enter findings for many of the nodes, and then have it do the belief updating.

Belief updating is done immediately after entering a finding only if auto-updating is turned on for that network. It is on if the **Network → Automatic Updating** menu entry is checked (when the network is the active window). You can turn it on or off by choosing **Network → Automatic Updating** from the menu. When auto-updating is off, and you wish to do belief updating, use the **Network → Update** command, or click on the right arrow tool bar button (they will be dimmed if belief updating is not currently required for the network or if it is not compiled).

If auto-updating is turned off, and you enter a new finding, or if no findings are entered but belief updating was never done, then the belief-bars or belief-meters will not display valid results, and so they are drawn fuzzy-dotted rather than in solid black. The reason they are not removed completely is that you may want to see what the beliefs were at the point of the last belief updating.

2.7 Negative and Likelihood Findings

In the sections above we saw how to enter positive findings into a belief network to do probabilistic inference. A *positive finding* is the observation or knowledge that some discrete node definitely has a particular value. However, we may discover that some node definitely does *not* have some particular value, and not have any more information to help us determine what value it does have. This is called a *negative finding*.

For example, say the node “Temperature” can take on the values cold, medium, and hot. We may obtain information that the temperature is not hot, although it doesn’t distinguish between medium and cold at all. This is a single negative finding. If later we receive another negative

finding that the temperature is not medium, then we can conclude that it is cold. So several negative findings can be equivalent to one positive finding.

A third type of finding is a *likelihood finding* (also known as “virtual evidence”). In this case we receive uncertain information about the value of some discrete node. It could be from an imperfect sensor, or from a friend who is not always right. Say we have a thermosensor to measure “Temperature”, which is designed so that when the temperature is hot it is supposed to turn on. In actual practice we find that when the temperature is cold the sensor never goes on, when the temperature is medium it goes on 10% of time, and when it is hot it always goes on. If at a certain time we observe the sensor on, and want to enter this finding into the Temperature node, then we do so as a likelihood finding. A likelihood finding consists of one probability for each state of the node, which is the probability that the actual observation would be made if the node were in that state. For our temperature example, the likelihood finding would be $(0, 0.1, 1)$. A common mistake is to think that the likelihood is the probability of the state given the observation made (in which case the numbers would have to add to one), but it is the other way around.

You can enter a negative finding for a belief-bar node of a compiled network, by holding down the shift key while you click on the name of the finding you know its not. You can enter a likelihood finding for a node by choosing “Likelihood” from its findings pop-up menu. You will then be queried for the likelihood numbers. You can enter a negative finding by entering a likelihood finding consisting of all 1s, except a single 0 for the state that you know it’s not.

Whenever you enter a positive finding for a node, all the old findings for that node are automatically retracted first. However, if you enter more than one likelihood finding for a node, you will be queried if you first want the previous finding(s) to be removed, or if you want them to accumulate. By letting them accumulate you can enter several *independent* pieces of evidence (e.g. imperfect observations) for the same node. If they are not independent, and it is too inaccurate to approximate them as independent, then they should be entered by adding child nodes to the observed node (one for each observation), connecting them together to show the dependency, and then entering positive findings for the child nodes.

2.8 Consistency

If you enter several likelihood findings for the same node, letting them all accumulate, then Netica checks to make sure they are consistent, and displays a message if they aren’t. The only way they can be inconsistent is to have a negative finding (or zero likelihood) for every state of the node.

Checking for consistency between the findings of one node and those of another node (given the inter-node relations encoded in the network), can only be done by belief updating. So if the network is kept valid by belief updating after each finding is entered (see auto-updating), then you will be alerted as soon as you try to enter an inconsistent finding, otherwise you won't be alerted until the next belief updating is in progress.

2.9 Most Probable Explanation

Given findings for some nodes, you may want to find the most probable configuration of values for the rest of the nodes. This can be thought of as providing a plausible explanation for the observed findings, and is called the *most probable explanation* or MPE (it is also known as maximum a-posteriori probability, or MAP). Notice that you cannot determine the MPE simply by taking for each node the state with highest belief after regular belief updating (which finds the marginal posterior probability for each node). For example, in a lottery for which we know there is one winner, it might be most probable for each individual person to lose, but then the overall configuration would have everybody loosing, which contradicts the one winner evidence. Finding the MPE would select one representative person to win, and the rest would be losers.

Netica can be used to find the MPE. To indicate that you wish to do MPE updating for a network, instead of regular belief updating, make sure the network is the active window, and choose the **Network → Most Probable Expl** menu item. The menu item will then have a check mark beside it indicating MPE updating will be done (by choosing it again you can toggle the check mark off). You will have to recompile between each switch, as is indicated by all the belief bars turning gray. You then enter findings in the usual way and it does MPE updating immediately afterwards if **Network → Automatic Updating** is on, or otherwise just after a **Network → Update** command.

After updating, each node will have a belief bar at the 100% level, and usually some bars at lower levels. You can read off the most probable configuration by taking for each node the state with the bar at the 100% level. The shorter bars indicate the relative probabilities of the other states given that the other nodes are in the most probable configuration (scaled by the same factor used to bring the longest bar to 100%). Of course the bars don't add to 100%, so if you are ever using Netica and are confused that every node has a 100% bar and there are also some other nonzero bars, it is because MPE updating is on. To avoid the possibility of accidentally doing MPE updating when regular updating is expected, all networks start with the MPE feature turned off, even if MPE was on when the network was last saved.

Sometimes two or more states of the same node have bars that are at the 100% level. This indicates that there is more than one configuration with the highest probability (i.e., the

configurations have the same probability). If more than one node has this, then you should choose one of the states and enter artificial evidence that the node is in that state, to see how it changes the multiple 100% bars of other nodes. By trying each of the possibilities you can map out all the configurations that are at the highest probability level.

You must be careful using the MPE. Generally, it is not as good as posterior probability (i.e. regular updating) for decision problems, or providing prediction or diagnosis probabilities. Its results can change with the introduction of irrelevant variables. And, it can be deceptive when even the most probable explanation is extremely unlikely. What the MPE is useful for is explaining and aiding understanding. If an agent finds the results of regular belief updating questionable, and asks you to provide a scenario for which the beliefs are upheld, you can use the MPE to find that scenario. People sometimes find a completely specified scenario easier to understand. And sometimes you can gain insights by putting the belief network in MPE mode, entering the evidence, observing the most probable configuration, and then experimenting with adding extra “evidence” to explore a set of probable configurations close to the most probable one, while seeing how changing one node effects the others.

3 Constructing Belief and Decision Networks

3.1 Opening a Window

The first step in creating a new network is to open a window for it. Use **File→New Network** or click on the tool bar button with the three connected nodes and the golden glint. A window will appear in which you can build the network. The title of the window will be set when you do a **File→Save** or a **File→Save As** command.

You will often find it easier to build a network by modifying an existing one. To do so, read in the existing one with **File → Open**, save it under the name of the new network with **File → Save As**, and then make the desired modifications, occasionally saving it with **File → Save**.

You can open as many network windows as you wish. As with any Windows application, the current operation always applies to the active window, which is the one in front with the non-dim title bar. You can make a window active by clicking on an exposed part of it, or by choosing its title from the **Window** menu. When you are done with a window, you can close it by clicking in the X box at its upper right, or by making sure it is the active window and then doing a **File → Close** command (if you have made unsaved changes, Netica will ask if you want to save them before closing the window).

3.2 Adding Nodes

The tool bar buttons with the gray oval, blue rectangle and green hexagon are used for adding nature nodes (i.e. “chance” or “deterministic” nodes), decision nodes, or utility nodes (also known as “value” nodes) respectively. When you click on the appropriate button, and then move the cursor over the active window, it will change to the shape of the node to be added. Next

click on the place in the network window where you want the node to appear, and it will be added there, with the cursor returning to normal. Instead of the tool button, you can use the appropriate command from the **Modify** menu, or its shortcut key equivalent (F9 - F11).

To add a series of nodes, double-click on the button for the type of nodes you wish to add, or double-press one of the keys F9 - F11 (i.e. press it twice in quick succession). The cursor will change to the shape of the node as before, but now it will be darkened indicating that a number of nodes may be added without pressing the node button each time. To exit this node adding mode, click again on the node adding button, or press the node adding Fx key again, or click on the pointer button (or click on a different node adding button or press a different node adding key).

When a node is first added it will be selected (i.e. drawn with inverted color). Whenever a node is selected you can press the <Enter> key to bring up a dialog box for setting node characteristics, and when the dialog box first comes up, its field for setting the node's name is selected for changing. This makes it possible to quickly add a node with a certain name. Just click the appropriately shaped cursor where you want the node to be, press the <Enter> key, type the name, and then press the <Enter> key again. There is no need to think about the dialog box or wait for it to be fully drawn. Using the dialog box to change other node characteristics is described in chapter 4.

3.3 Adding Links

To add a link from one node to another, first click on the link tool bar button (it is the one with the arrow pointing down, not the up-facing pointer), do a **Modify → Add Link** command, or press F12. When you next move the cursor to the active window, it will change to a link shape indicating that it is ready to add a link. Click on the node that you wish to be the parent, and then click on the child node. Alternately, you can click down on the parent, and while holding the mouse button down, move to the child and then release it. The link will appear. A later section will explain how you can change the shape of a link.

To add a series of links, double-click on the link button, or double-press the F12 button. The cursor will change to a link shape as before, but now it will stay that way as you add multiple links. To exit this link adding mode, click again on the link adding button, or press the F12 key again, or click on the pointer button (or click on a node adding button or press a node adding key).

3.4 Undoing and Redoing

While building or changing the network you can undo the last operation by choosing **Edit → Undo** from the menu (or pressing Ctrl+Z). To undo operations previous to that one, repeatedly invoke **Edit → Undo** (or repeatedly press Ctrl+Z). After a series of undos, you can “redo” one or more of them by repeatedly invoking **Edit → Redo** (or pressing Ctrl+Shift+Z).

Netica will save a certain number of the previous steps for undoing (at least 4, but possibly many more if they don’t take up much memory). If you are undoing operations and the **Edit → Undo** menu item turns dim, it means you have undone all the operations that Netica has remembered (which may only be a couple, if you have only done a couple since the last file save). If you are redoing operations, and the **Edit → Redo** menu item turns dim, it means that you have restored all the operations that were previously undone.

The operations done in each window are remembered separately, so if you return to window A after working in another window for a while, you can still undo operations previously done in window A.

3.5 Selecting Nodes and Links

Many operations on nodes are done by first *selecting* one or more nodes, and then choosing the operation to do. Netica indicates a selected node by drawing it with inverted colors.

You can select a node by clicking once on it. To select a group of nodes you can click down on the background and then drag the selection rectangle to include at least a part of each of them. To unselect all the nodes currently selected, just click once on the background within the window. To select all the nodes in the network, you can use the **Edit → Select All** command (or press Ctrl+A).

When you select new nodes, any nodes that were previously selected will become unselected, unless you hold down the <shift> key while clicking or dragging. In that case the selected status of the new node(s) will be reversed, allowing you to add to, or remove from, the collection of selected nodes.

To select a link, click right on it. It will become drawn in hilited outline form. Once again, if the <shift> key is held down while clicking, it will reverse the selection status of only the link being clicked on, which can be used to add or remove from your collection of selected links. Links cannot be selected by dragging the selection rectangle over them. Nodes and links cannot both be selected at the same time, so whenever you select links, any nodes that are selected will become unselected, and vice-versa.

3.6 Moving Nodes and AutoGrid

To move a node, click down on it, drag it to its new position, and then release the mouse button. If you wish to move a set of nodes and the links between them, select the nodes you wish to move, then click down on one of them and drag the set to the new desired location.

There is an underlying “grid” of positions, and when you add a node or move a node, the node will be shifted slightly so that its center is directly over one of these grid positions, providing the autogrid is turned on. This allows you to quickly draw a network whose nodes are perfectly aligned, and with some links running perfectly horizontal or perfectly vertical. The autogrid is turned on when there is a check mark in front of the **Layout→AutoGrid** menu entry. To turn it off or on, choose the **Layout→AutoGrid** menu entry. If you want to change the spacing of the grid, to make it finer or courser, do a **Layout→Grid Spacing** menu command, and a dialog box will appear allowing you to make the setting.

Sometimes it is useful to be able to move a node, or set of nodes, a small predictable amount. You can use the arrow keys to do this nudging. Simply select the nodes you wish to move, and then press the key with the arrow in the direction you want them to move. The distance they will be moved with each press of the key is the grid spacing (providing the autogrid is turned on), so you can adjust this amount using the **Layout→Grid Spacing** menu entry. If the autogrid is turned off then they will be moved the smallest perceptible amount with each press of the key.

3.7 Reshaping a Link

When network diagrams become large it can be very difficult to view them if all the links between the nodes are straight lines. Even small diagrams can sometimes be made more legible by choosing suitable paths for the links.

To shape a link, first click once on the link to select it. The link will be drawn with the hilite color surrounding it. Then click down again on the selected link, and drag the cursor. A bend will be placed in the link and dragged by the cursor. You can repeat this to add as many bends as you wish.

To move the position of a bend, first select the link as before. The link will become hilited and there will be a square box at each of its bends, and at its two endpoints. To move a bend or an endpoint, click down in its hilited box, drag it to its new position, and then release the mouse button. Moving one bend into another, or into an endpoint, will combine them, which can be used to remove bends.

There are some special considerations when moving the endpoints of a link. The arrow end of a link can not be dragged very far from the node it points to (if you try to, the arrow will just bump against an invisible barrier). This prevents the creation of misleading looking diagrams (since the link is still considered to be connected to the node). If you drag the non-arrow end of a link too far from its parent node, then the link will become “disconnected” from that node. You can tell this has happened because the name of the link will suddenly appear at the endpoint you have moved. See section 7.1 for more information on this.

If the autogrid is on (there is a check mark in front of the **Layout** → **AutoGrid** menu entry) then the endpoints and bends of a link will always be placed on grid positions after a move. This makes it easy to quickly draw network diagrams with aligned link segments, and to make some segments perfectly horizontal or perfectly vertical. You can turn the autogrid on or off by choosing the **Layout** → **AutoGrid** menu entry, and set the spacing of the grid with **Layout** → **Grid Spacing**.

If you have just been moving a link bend or endpoint, and the link is still selected, you can nudge that bend or endpoint a little to get it exactly where you want by pressing the appropriate arrow key. With every press of the key, it will move a distance of the grid spacing, if the autogrid is on, or the minimum perceptible distance if the autogrid is off.

3.8 Saving a Network to a File

To save a network to a file, make sure its window is the active one (by clicking in it or choosing it from the **Windows** menu), and do a **File** → **Save As** command. You will be prompted for the file name and what directory to place it in. After you have saved a network its window title will no longer be “Untitled-x” where x is some number, but instead it will be the name of the file saved to (without a terminating “.dne” or “.dnet”). Then you can save it subsequent times using the **File** → **Save** command without being prompted for the name. If the **File** → **Save** command is dimmed, that means that the network has not been changed since it was last saved, or read, and so the screen version is the same as the file version.

The network gets saved to a file consisting only of ASCII text, so it may easily be transferred from one type of computer to another (e.g. from Windows to Unix to Macintosh) or sent by email. You may examine and modify the file with a text editor. It is in the DNET-1 file format, so it contains “// ~->[DNET-1]->~” somewhere in the first 3 lines. A document precisely describing the DNET-1 file format is available from the Norsys WWW site (called “DNET_File_Format.txt”).

3.9 Deleting Nodes and Links

To delete a node or nodes, first select them, and then press the <delete> key or do an **Edit → Delete** command from the menu. This will remove the nodes, and all links from other nodes going to them. Links going from them to their child nodes will be disconnected just before the deletion (see section 7.1). These disconnected links will be selected after the nodes are removed, so you can easily delete them as well by just pressing the <delete> key a second time.

If you wish to remove the nodes, but maintain the global relationship of the remaining nodes (i.e. the joint distribution), see the **Network → Absorb Nodes** command.

To delete a link, click on it to select it, and then press the <delete> key or do an **Edit → Delete** command from the menu. To delete a series of links, select them all and then press the <delete> key. When a link is deleted, the conditional probabilities of the child node are collapsed to eliminate their dependence on that parent, as if the parent took on its first state. When a link is deleted the parent node is not affected in any way.

3.10 Disconnecting and Reconnecting Links

If you wish to disconnect a node from one of its parents without modifying its conditional probabilities (perhaps to put it in a library, or to connect it to a different parent), the link between them may be “disconnected”. To do this, select the link (or links) and do a **Modify → Disconnect Links** command (or click on the knife toolbar button). Selecting a node (or nodes) and doing a **Modify → Disconnect Links** command will disconnect all links entering the node(s). A disconnected link turns into a short stub connected to the child node and labeled with the *link name*. This will be the name you gave it previously (see chapter 4 “Changing Node Characteristics”), or if you never did, it will be the name of the old parent node. If you want to disconnect a link without loosing all its bends, click on the link to select it, then drag the hilited box at the tail of the link far enough away from the parent node, and release the mouse button. You will know that you dragged it far enough if the link name appears at the tail of the link after releasing the mouse button.

A link cannot be disconnected from its child node. That would not make sense, since a disconnected link is really just a “parent place-holder” to maintain the node’s relation (e.g. conditional probabilities) until it is connected up to a new parent. To achieve the affect of disconnecting a link at the child end, it sometimes makes sense to do a link reversal (see “link reversal”), and then disconnect the link at the parent end.

To connect a disconnected link to a new parent, click on the link to select it, then drag the hilited box at the tail end of the link over the node you wish to connect it to, and release the mouse button. A link can be disconnected from one parent node and reconnected to a new parent in one motion, by first selecting the link, and then dragging the hilited box at the link tail from the old parent to the new. When reconnecting a link, the new parent must be compatible with the old one in certain ways. For example, since the conditional probability table is maintained during the reconnection, the new parent must have the same number of states as the old. If there is any incompatibility between parents, an error message will be displayed.

3.11 Cut, Copy, Paste and Duplicate Nodes

Selected nodes can be removed from the network and transferred to the clipboard with an **Edit → Cut** command from the menu (or pressing Ctrl+X keys). Alternately, they can be entered into the clipboard without removing them from the network using an **Edit → Copy** menu command (or pressing Ctrl+C keys). If no nodes are selected when the cut or copy command is done, the whole network will be cut or copied to the clipboard.

Once the nodes are in the clipboard, you can duplicate them into any network by opening the window for that network, optionally clicking in it where you want them to be placed, and then doing an **Edit → Paste** menu command (or pressing Ctrl+V keys). All links between the transferred nodes, all their characteristics, and all their conditional probability tables will be maintained during the transfers. Links that enter cut or copied nodes from a node not being cut or copied will appear as disconnected links in the duplicate. Links that exit cut or copied nodes going to a node not being cut or copied will not be copied.

Networks may be copied and pasted into other programs, such as Microsoft Word. See section 11.5 “Copying and Pasting Graphics” for more information on this.

Nodes may be duplicated within their network, by selecting them and doing an **Edit → Duplicate** menu command. All links between them, and all their characteristics and probability tables will be duplicated as in the copy/paste situation, but any links to them from parent nodes not being duplicated are handled differently. Instead of being disconnected, they are duplicated as well, which is especially useful to create nodes representing repeated, independent, imperfect observations of some node (i.e. “virtual evidence”).

Since every node name in a network must be unique, a node’s name must be changed when duplicating it into a network already having that name. This is done by adding a number to the end of the old name, or incrementing the number if there already is one. Of course node titles

don't have to be unique, so after duplicating or pasting you may have several nodes with the same title.

3.12 Drawing Size and Scrolling

Using the scroll bars you can move around to view or work on various areas of the network. The <home>, <end>, <page up> and <page down> keys may be used as well. While you are moving a node or a link bend, you can make the window scroll by attempting to drag them outside the window in the direction you wish to scroll. The further you go outside, the faster it scrolls.

The area within which you can create a network diagram is a rectangle of limited size. Once you scroll to the end of it you cannot scroll any further, and you cannot put any nodes outside of its boundaries. You can change the size of this area with the **Layout → Drawing Size** menu command. If you increase the size of the window larger than the current drawing size, then the drawing size will be automatically increased (including maximizing a window). The size of the page can be adjusted with the **File → Printer Setup** menu command, and is based on how much your printer puts on one page. There is more information on pages in section 11.6 “Printing”.

4 Changing Node Characteristics

We have seen how you can add or delete nodes or links by using menu commands and the tool bar. However, to change the characteristics of a node (e.g. its name, or how many states it has) you need to use what is called the *node dialog box*. If you want to change the way a node is related to its parents (e.g., its conditional probabilities), you use the *relation dialog box*, which is described in the next chapter.

4.1 Obtaining the Node Dialog Box

The easiest way to obtain the node dialog box for a node is to double-click on the node. Another method is to select the node and then press the <Enter> key. This last method is especially useful when adding new nodes to the network, since when a node has just been added it is already selected, and you just have to press the <Enter> key.

You may have several node dialog boxes on the screen at the same time, and you may alternate between using different node dialog boxes and directly working in the network window. Each node dialog box pertains to a single particular node. In fact you can even have more than one node dialog box for the same node (see section 5.10 “Multiple Dialogs for One Node”).

4.2 Making Changes

Whenever you make any changes within the node dialog box, they don’t actually affect the node until you click on the Apply or Okay buttons. Then all the settings are transferred to the node in the network. The only difference between the Apply and Okay buttons is that the Okay button removes the dialog after transferring the settings, while the Apply button doesn’t.

Clicking on the Load button will make all the settings in the dialog box the same as the node in the network.

If you click on the Close button, or in the dialog's go-away box (on the left end of the title bar), or do a **File** → **Close** menu command, the node dialog will be removed. If you previously have made some changes in the dialog box, and haven't pressed the Apply button, Netica will ask if you want to apply those changes to the node before removing the dialog box.

4.3 Node Name

You can change the name of a node using the "Name" text edit box of the node's dialog box. When the node dialog box first appears, the existing name of the node is selected, so if you just type a new name it will replace the existing name.

When you enter a new node name, Netica will check that it meets the restrictions of an *IDname*. It must start with a simple letter (a-z or A-Z), it must be composed only of simple letters, digits and underscores (_), and it must be 30 or fewer characters long. Also, it must be different from the names of all other nodes in the network. If you want to circumvent any of these restrictions, you can also give the node a title, which has no such restrictions.

4.4 Node Title

The node dialog box has a text edit box labeled "Title", so that you can give the node a title, which will be used to label the node on network diagrams. There are no restrictions on what you may put in a title, and by using the <Enter> key you can make the title more than one line long. By putting a blank line at the beginning or end of the title, or putting a space character at the beginning or end of a title line, you can effectively add some space between the outside border of a node and the writing within. If the title is longer or taller than the text entry box, you can scroll it using the arrow keys. The title is only used to label the node; anything more detailed should go in the "description" of the node. Keep in mind that some unusual characters may display differently when the font is changed, or if the network is displayed on another type of computer.

Nodes are not required to have titles. If a node does not have a title, then anytime that Netica would normally use a title, it will use the node's name instead.

4.5 Node is Discrete or Continuous

The node dialog box has a selector with the values "Continuous" and "Discrete", which allows you to choose whether the node represents a continuous or a discrete variable. A discrete variable is one with a well defined finite set of possible values, called *states*. Examples are: the

number of dimes in a purse, a statement which is either “true” or “false”, which party will win the election, the country of origin, voltage output of a digital device, and the place a roulette wheel stops. A continuous variable is one which can take on a value between any other two values, such as: indoor temperature, time spent waiting, water consumed, color wavelength, and direction of travel. A discrete variable corresponds to a digital quantity, while a continuous variable corresponds to an analog quantity.

Often a variable will be continuous at one scale, but discrete on another. For instance the amount of water consumed might be discrete if you count individual water molecules, but it is continuous at the scale we are concerned with. Likewise, the voltage output of a digital device might be discrete at the scale we are concerned with (“high” & “low”), but continuous on a finer scale (0.7 - 3.5V), and then discrete on a very fine scale (corresponding to the number of electrons on a capacitor). You only need consider the scale of interest when setting whether a node is continuous or discrete.

It is common to have a continuous variable that you want to break up into ranges so that you can treat it as a discrete variable, which is known as *discretizing* the variable. It is usually best to make it a continuous node, and then discretize it with a range list as described below in section 4.8 “Node State Range”, rather than just making it a discrete node. This provides better documentation of the node, and makes it easier if at a later time you want to discretize it another way (i.e. with a different number of states, or different cutoff points for the state ranges).

A utility node must always be made continuous and a decision node must always be made discrete.

4.6 Node Kind

Directly below the name entry area, the node dialog box has a selector which may be used to turn the node into a nature node, decision node, or utility node (also known as a “value node”). Nature nodes are sometimes called “deterministic nodes” (when they depend on their parents in a deterministic way), or “chance nodes” (when the dependence is probabilistic). When a network is composed entirely of nature nodes it is called a belief network (also known as a “Bayesian network”). If it also has decision and utility nodes, it is called a decision network (also known as an “influence diagram”). Decision nodes represent variables that the decision maker can control, and utility nodes represent variables the decision maker is trying to optimize. See chapter 6 “Decision Problems” for more information.

You can also change node kinds straight from the network window without using a node dialog box. To do this, select a node or set of nodes you wish to change, and while holding down the <

Ctrl > key click on the tool bar button indicating the kind of node you wish them to become. To convert them to nature nodes click on the yellow oval, to convert them to decision nodes click on the blue rectangle, and for utility node click on the green hexagon.

4.7 Node States

In the node dialog box, next to the label “States:”, is a down-arrow which yields a pop-up menu of the node’s states. By choosing a state from the menu, that state becomes the “current state” of the node dialog box. Its name will appear in the text edit box to the right of the pop-up menu, allowing you to change the name. It is not necessary that the states of a node have names, but if one state is named then they all must be. It is highly recommended that a discrete node be given state names, for clarity and documentation purposes, although it is not as necessary for discretized continuous nodes.

When you enter a new state name, Netica will check that it meets the restrictions of an IDname. It must start with a simple letter (a-z or A-Z), it must be composed only of simple letters, digits and underscores (_), and it must be 30 or fewer characters long. Also, it must be different from the names of all other states of that node, although it may be the same as a state name for a different node in the network.

To add a new state right after the current state, click on the “New” button, and to delete the current state, click on the “Delete” button. All discrete nodes must have at least one state, but there is no upper limit to how many they can have.

Many people find it faster and easier to enter or change the states of a node using the “Multi-Purpose Box” described in section 4.10.5 below, instead of the previously described method.

4.8 Node State Range

If a node dialog box is for a continuous node, then immediately below the state name edit box will be a couple of boxes labeled “Range”. These are to provide threshold values for the ranges of each state in order to discretize the continuous variable (see section 4.5 “Node is Discrete or Continuous” above). This method of entering the discretization ranges is mostly just for touching-up, or to coordinate the ranges with state names, but usually you will want to use the “Multi-Purpose Box” described in section 4.10.6 below instead.

The box to the left is the lower end of the range for the current state (inclusive), and the box to the right is the upper end of the range (exclusive). For instructions on changing the “current state”, see section 4.7 above.

To set or change the range of the current state simply type a number in the appropriate box. States are arranged in a contiguous and (increasing or decreasing) monotonic way, which means that the upper end of one state range will always be the lower end of range of the neighboring state. As you type one of these in, Netica will fill in the other as well, so that when you change the current state you will see the number you just entered in one of the Range boxes.

It is okay if the lower and upper bounds of the range are the same. The numbers you enter may be integers, decimal numbers, scientific notation numbers, “infinity”, or “-infinity”.

4.9 Node State Value

If a node dialog box is for a discrete node, then immediately below the state name edit box will be a box labeled “Value”. This is to associate a real number value with each state. You will rarely need this, but occasionally it is useful. For example, in a bang-bang control system, state 0 (“Off”) may map to -0.2 volts, and state 1 (“On”) may map to 6.0 volts. If this node is the parent of a node with an equation, then the real values will be supplied to the equation (see chapter 12 “Equations” for more information).

To set or change the real number associated with the current state simply type a number in the box. See section 4.7 above on how to change the current state. You may find it easier to use the “Multi-Purpose Box” described in section 4.10.6 below to enter or change the real number associated with a state.

4.10 Multi-Purpose Box

At the bottom of the node dialog box is a text edit box with a selector above it. This box is used to view or enter several different characteristics of the node, and the selector is to choose which characteristic to view or enter. This selector is known as the node dialog box’s “multi-purpose selector”, and each of the section headings below corresponds to one of its choices. To scroll in the text edit box, click down within the box and then drag the cursor in the direction you wish to scroll, or use the arrow keys to attempt to move the cursor past the box’s boundary. If you have a very large amount of text it is probably best to just paste it in from a text editor.

4.10.1 Node Description

When a node dialog box’s multi-purpose selector is set to “Description”, you can enter or view the *description* of the node. The description is an unrestricted block of text which can be used to store whatever information about the node you wish, such as its meaning, the meanings of its

states, how it is to be measured, the origin of its probabilities, copyright information, etc. The description can be as long as you wish.

4.10.2 Node Equation

When the multi-purpose selector of a node dialog box is set to “Equation”, you can view or enter the probabilistic or deterministic equation that provides the relation between the node and its parents. See chapter 12 “Equations” for more details.

4.10.3 Input Name

When the multi-purpose selector of a node dialog box is set to “Input Name”, you can view or enter a name for each of the links going to the node. As soon as you set the selector to Input Name, an additional selector will appear to the right of it. Each choice of the new selector will correspond to a link, and will be labeled with the current link name if the link has one (in parenthesis), and/or the name of the parent node it comes from if the link is not disconnected. See chapters 7 “Network Libraries” and 12 “Equations” for the purpose of link names.

4.10.4 Link Delay

When the multi-purpose selector of a node dialog box is set to “Link Delay”, you can view or enter a time-delay for each of the links going to the node. As soon as you set the selector to Link Delay, an additional selector will appear to the right of it. Each choice of the new selector will correspond to a link, and will be labeled with the link name if there is one, and/or the name of the parent node it comes from if the link is not disconnected.

4.10.5 Node States

When a node dialog box’s multi-purpose selector is set to “States”, you can enter or view how many states the node has, and what their names are. You can also do these operations using the State Name setting box described earlier, which has the same capabilities and effect. However, sometimes you may find this method more convenient, especially if you want to paste in the names of all the states at once, which you have copied from another node or even from another program.

You simply enter the names of all the states into the box, separated by space(s), tabs, commas or on separate lines. If the number of states in the list is different from the node’s current number of states, the node’s number of states will be changed.

An example entry is: low medium high

4.10.6 Node Ranges

When a node dialog box's multi-purpose selector is set to "Ranges", you can enter or view the discretization of a continuous variable or supply a number for each state of a discrete variable. You can also do these operations using the Range setting boxes described earlier, which have the same capabilities and effect (see sections 4.8 and 4.9 above). However, usually this method is more convenient, especially if you want to enter many evenly spaced values, or if you want to paste in the ranges of all the states at once, which you have copied from another node or even from another program.

You simply enter all the range levels into the box, separated by space(s), tabs, commas or on separate lines. See section 4.8 "Node State Range" above for a description of the meanings of these numbers. If the node is continuous, then there should be one more number than the desired number of states, and if the node is discrete then there should be one number for each desired state. If the number of states implied by the list is different from the node's current number of states, the node's number of states will be changed.

If you want to create a list of evenly spaced values there are a few shortcut methods you can use. Each of the following special notations will expand into a list of numbers as described:

[b, e] / n will form a list beginning with b, ending with e, and having n ranges.

[b, e] + d will form a list beginning with b, ending with e, and each separated by d (except the last separation may be less if e – b is not evenly divisible by d).

[b, e] /L n will form a list beginning with b, ending with e, and divided logarithmically into n ranges.

[b, e] +% d will form a list beginning with b, ending with e, each d percent bigger than the previous (except the last may be less than d % bigger, if they don't fit evenly).

Note that if e is less than b then a decreasing list will be formed, but n and d should still be entered as positive numbers. The closing bracket may be replaced with a closing parenthesis if desired, to indicate excluding the endpoint e from the list formed. More than one of the above notations can be combined to form a longer list.

Here are some example entries (one on each line):

-3.2 0 1 1e4 infinity

[0, 10] / 10

[0, 10] + 1, [10, 20] + 2, [20, 30] + 3, 33, 37

[1e6, 1] /L 6
[200, 10] +% 15

4.10.7 When Changed

When the multi-purpose selector of a node dialog box is set to “When Changed”, you can view the time and date the node was last changed, based on the clock in your computer. This value is similar to the one maintained by your computer operating system for when each file was last changed, but it can be more useful for belief networks since it actually records the time of change, not just the time of file saving, and it provides a separate value for each node. You cannot change this value; it is for observation only. If no value appears, it means that the time of the last change was not saved in the belief network file, and the node hasn’t been changed since reading from file.

5 Node Relationships

In the previous two chapters we saw how to create nodes, set their characteristics (such as name or number of states), and link them together. This chapter will explain how to enter the actual relationships nodes have with each other, which are most often in the form of conditional probability tables. We enter, change or view the relationship using a *relation dialog box*.

5.1 Obtaining a Relation Dialog Box

The way to obtain the relation dialog box for a node is to select it and then click the tool button with the green relation tree (looks like two inverted V's), or do a **Relation → View / Edit** menu command.

You may have several relation dialog boxes on the screen at the same time, and you may alternate between using different relation dialog boxes, different node dialog boxes, and directly working in the network window. To use one of the relation dialog boxes it must be the active window (which means it is in front with its title bar non-dim). To make a relation dialog box the active window, just click on an exposed part of it, or choose its name from the **Windows** menu.

5.2 Meaning of the Tables

For a belief or decision network to be fully specified, there must be a relation stored at each nature or utility node, which expresses the value of that node in terms of its parents (or as a constant if the node has no parents). If the node is deterministic then the relation will be a function which provides a value for the child for each possible configuration of parent values. If the node is probabilistic (i.e. a chance node), then the relation must provide a probability for each state of the child, for each possible configuration of parent values.

For example, suppose node A (which can take on values Low, Medium or High) and node B (which can take on values True or False) are the two parents of node C (which can take on values Small, Midsize or Large). It is best to think of the relation between them as being located at node C (the child), and its relation dialog box might look something like this:

A	B	Small	Midsize	Large
Low	True	10.000	35.000	55.000
Low	False	17.250	82.750	0.000
Medium	True	12.100		
Medium	False	33.333	33.333	33.333
High	True	x	x	x
High	False	54.328	45.672	3.2e-012

On the left-hand side is a vertical list of all the configurations of parent values. On the right-hand side is one column for each state of C. The numbers in the table provide conditional probabilities for the values of C, given that the parents take on the configuration of their row. For example, the 10.000 (percent) in the upper left corner means that $P(C=Small | A=Low, B=True) = 0.1$. The number $3.2e-12$ at the bottom right means $P(C=Large | A=High, B=False) = 3.2 \times 10^{-12}$. The two empty cells on the third row indicate probabilities that have not yet been specified, and the three x's on the fifth row indicate that the designer believes that the condition A=High while B=True is impossible.

5.3 Changing Table Entries

You will often bring up a relation dialog box just to view the relation it represents. For example, if you have just solved a decision network, you can use it to view the optimal decision functions that Netica has found. However, its main purpose is to enter or change node relations when building a belief or decision network, or doing what-if analysis with an existing network.

If the dialog is for a deterministic node, then you can change a table entry (also known as a *cell* of the table) simply by clicking down on top of it, and then making a choice from the pop-up menu which appears.

If the dialog is for a chance node, then you can change a probability by clicking on it to select it, and then typing in the new number. Or you can click on it to select it, then click within the selection where you want to change a few digits. Remember to enter the number as a percentage. If you want to enter probabilities as decimal fractions (e.g. 0.25), rather than percentages (e.g. 25), see the normalize command in section 5.9 below.

If a cell is being edited, the insertion point will be flashing where new digits will enter. You can use the left or right arrow keys to move it around within the number, and if it gets to the edge of the number, it will jump to the next cell in that direction. Further presses will jump to subsequent cells in that direction, and the up and down arrow keys can be used in the same way to jump to cells above or below the currently selected or edited cell. Pressing the tab key jumps to the “next” cell, which is the cell to the right of the current one, unless it is at the end of the line, in which case it is the first cell on the next line.

5.4 Buttons in the Relation Dialog Box

Whenever you make any changes within the relation dialog box, they don’t actually affect the node until you click on the Apply or Okay buttons. Then all the settings are transferred to the node in the network. The only difference between the Apply and Okay buttons is that the Okay button removes the dialog after transferring the settings, while the Apply button doesn’t.

Clicking on the Load button will transfer all the values from the node in the network to the dialog box. There are two main uses for this button. The first is if you make a mistake while you are changing the table, and you haven’t yet pressed the Apply button, you can “revert” to the original node by clicking on the Load button (if you have already pressed the Apply button, you should make the network window active, do an **Edit → Undo** command, then go back to the relation dialog and click on Load). The second use for the Load button is to update the relation dialog after something else has changed the relation at the node. For example, you might be solving a decision network repeatedly, while varying some part of it. After each solution you want to see how the optimal decision function has changed. You would bring up the relation dialog box for the decision node of interest, leave it up, and after each re-solution of the network you would click on the Load button to observe the changes in the tables. Another example could be that you are having Netica learn a belief network from a number of case files, and you want to observe how the conditional probabilities change as the learning process continues.

You can switch which node a relation dialog box is for, by using its pop-up menu labeled “Node”. From this menu you can choose any node in the network, and when you do, the contents of the dialog will be completely re-adjusted for the new node. If you had made changes

in the dialog box before switching nodes, and had not yet pressed the “Apply” button, then a message will be presented asking if you want to apply the changes before switching nodes.

If you click on the Close button, or in the relation dialog’s go-away box (on the left end of the title bar), or do a **File** → **Close** menu command while the dialog is the active window, the dialog will be removed. If you previously have made some changes in the dialog, and haven’t pressed the Apply button, Netica will ask if you want to apply those changes to the node before removing the dialog.

5.5 Deterministic / Chance Pop-Up

The relation dialog box has a selector that allows you to make a nature node deterministic or probabilistic (i.e. a chance node). If you use this selector before you have entered anything into the table cells, it will just modify the dialog box to be suitable for entering deterministic or probabilistic information. If you have already entered a relation, that relation will be converted. Converting a deterministic relation to a probabilistic representation is simple: each row of the result will consist of all 0 entries, except one 100% entry at the state the original deterministic function mapped to. Converting a probabilistic relationship to a deterministic one generally loses some information. For each parent configuration the function output will become the state that was most likely in the probabilistic relation (i.e. had the highest conditional probability).

5.6 Scrolling and Resizing

Since a node may have very many parent configurations, they often won’t all be visible in the relation dialog box at once. You can use the scroll bars to view whatever part of the list you are currently interested in. The <home>, <end>, <page up> and <page down> keys may be used as well.

You can also resize the relation dialog box to make it bigger or smaller by clicking in the box at the lower right corner and dragging to the new size. When you first bring up the relation dialog box, it is constructed with a size to match the tables required for the node its for. If you change which node the dialog box is for, it is sometimes convenient to resize the dialog box as well, so it better suits the tables of the new node.

5.7 Unspecified and Impossible Cell Values

There are two special values which can appear in any cell of the relation table, whether it is for a deterministic or probabilistic node. One is a blank cell, which means that the value of the cell has not been specified. You may be building up a network over a number of sessions with Netica, and you can use the blank cells to keep track of which parts of the relationships have not yet been determined. New relations start out with all blank cells, and you can set any cell or group of cells to blank by selecting the cells and then pressing the <delete> key or doing an **Edit → Delete** menu command. If the node is deterministic, you can set a cell to blank by choosing “Unknown” from the pop-up menu which appears when you click on the cell. If you try to do inference with a network which has some unspecified values, Netica will not allow it and display a message.

The other special cell is one that has a single x in it. This indicates that no value is required for the cell, because that configuration of parent values will never occur. It actually doesn’t say anything about the relationship at hand, but just that whoever was building the belief network didn’t feel it was necessary to enter a value because in the context of the overall network it is not required. If Netica is later doing inference and discovers that configuration of parent values can occur, it will display a message alerting you.

If you put any arbitrary value in such a cell, all inference results will be the same, because the value will not be used. The best value to put in the cell is the true value that would be applicable if the relationship between the parents was different, but if that value doesn’t exist, or you don’t know it, or have time to determine it, it is much better to use the x feature than to just put in an arbitrary value, for three reasons. First, during inference Netica performs an important check for you as to whether the configuration is really impossible. Second, it documents your understanding at the time you were building the belief network for anyone else who later works with it. Third, if you later change other parts of the network, or you cut and paste this node into a different application, or put it in a network library, those parent configurations may become possible.

You can set a chance cell to x by selecting the cell and then typing x, and a deterministic cell to x by choosing ‘Impossible’ from the cell’s pop-up menu. For chance nodes, if one cell has an x, then all the cells in that row *must* have an x (because they all correspond to the same parent configuration).

5.8 Selecting, Cutting and Pasting

To do operations on a restricted group of cells, you first *select* them, which turns their background to the hilite color.

To select cells of a probabilistic relation, you click down in one cell, and drag the mouse to another cell before releasing the mouse button. While you are moving the mouse, all the cells in the rectangle between the original cell and the current mouse position will be hilited. The cell you first click down in must not be the cell you are currently editing (i.e. the only selected cell, or the cell with the insertion point blinking in it) or Netica will think that you are doing an operation to edit just that cell. If you wish the selected area to extend beyond the window boundary, just drag outside the boundary and the window will automatically scroll. Once you have made a selection, you can hold down the shift key while you click on another cell to extend or reduce the selection.

To select cells of a deterministic relation, you click down on the area slightly to the left of the cell (if you click right on the cell, instead of selecting, you will get the pop-up menu for changing the cell's value). You can drag below or above the window to force automatic scrolling. Also, you can extend or reduce a selection by holding down the shift key when you click in another cell.

After you have selected some cells, you can copy their values to the clipboard by doing an **Edit → Copy** command. If you want to place those values in some other cells, you select the other cells and then do an **Edit → Paste** command. When you paste, the number of rows and columns of cells selected must be the number of rows and columns in the clipboard (i.e., the same as the number of rows and columns selected when you originally did the **Edit → Copy** command).

You can copy cell values from a spreadsheet or word processor to the clipboard, and then paste them into a Netica array of cells, or vice-versa. When you paste values into Netica cells, the number of rows and columns in the clipboard must match the number of selected Netica cells. If the data you are pasting consists of probabilities expressed as decimal fractions (e.g. 0.25), rather than percentages (e.g. 25%), then you should do a normalize command (see the following section) after pasting. When you copy Netica cells to the clipboard they are entered as text, with a tab between each entry on the same line, and a carriage-return / line-feed pair between lines and at the end of the last line. If you copy from a word processor or text editor to Netica, the format should be the same, except you may substitute space(s) and/or a comma for each tab, and linefeed(s) for carriage returns, if you wish. Blank lines will be ignored.

5.9 Relation Menu Commands

While working in a relation dialog box, the **Relation** menu is often useful. First you select some cells you wish the menu command to apply to, and then choose the command from the menu. For most of the commands, if no cells are selected, the command is applied to all the cells of the dialog.

Relation → Fill in Missing enters a probability into blank cells so that the probabilities of each selected row add up to 1.0 (i.e. 100%). Any row having no blank cells, more than one blank cell, or “impossible” values will simply be ignored. Using the tool bar button with the tiny number entering the small blue empty cell is another way to do this command.

Relation → Normalize will make sure that the probabilities of each selected row adds up to 1.0 (i.e. 100%), by dividing each number by the sum of the row. Any row having unspecified or “impossible” values will simply be ignored. This command is useful if you want to enter probabilities as decimal fractions (e.g. 0.25), rather than percentages (e.g. 25%). After you enter all the numbers, simply normalize them and they will be converted to percentages. The normalize command can also be done using the tool bar button with the large 1 in it.

Relation → Uniform Probabilities will set the selected rows of a probabilistic relationship to uniform probability vectors, which means that all states of the child are equally probable. So all the cells of these rows will contain the number 1 / (number of states), expressed as a percentage.

Relation → Randomize works on probabilistic or deterministic relationships. For deterministic relationships it sets each selected cell to one picked randomly from the possible states of the node, following a uniform distribution (i.e. each state equally probable). For probabilistic relationships it sets the cells of each selected row to randomly selected probabilities. Of course, the probabilities of each row will add to 1.0 (i.e. 100%). The distribution of the probabilities entered is *not* uniform (it favors numbers closer to 0). Using the tool bar button with the single die overlaid on the green relation tree is another way to do this command.

Relation → Remove will convert all the selected cells to blank cells. For an explanation of blank cells see section 5.7 above. If there are selected cells, then pressing the <delete> key will have the same effect. This command can also be done using the tool bar button with the red X over the green relation tree.

Relation → Check will check a probabilistic relationship to make sure all the probabilities are between 0 and 1 (100%), and that for each parent configuration (i.e. row) the probabilities add to 1 (100%). A message will be displayed indicating the selected rows are all okay, or an offending

row will be hilited and a message displayed indicating the problem. This command can also be done using the tool bar button with the black check mark.

Many of these commands can also be done from the network window, without using any relation dialog, by selecting the nodes of interest, and then choosing the menu command from the **Relation** menu. It will apply to the entire relationships of all the selected nodes.

5.10 Multiple Dialogs for One Node

You may have more than one relation dialog box (or node dialog box) for the same node. This can be confusing if you are not careful, so it is not recommended unless you need it.

However, in some situations it is very useful. Each dialog box stores all the settings for its node and doesn't loose them when you make changes in another node dialog box or to the node itself (unless you press the Load button). So you can have several different settings for a node, one set in each dialog box, and alternate between them just by pressing the Apply button on the dialog box of choice. Of course you can do all kinds of other operations in between (such as compiling the network, saving it to a file, making queries, etc.).

You open multiple dialogs for a node in the same way as you open a single dialog. Just do the action repetitively without closing the previous dialogs first, and answer 'no' to the question of whether you want to bring the existing dialog to the front.

6 Decision Problems

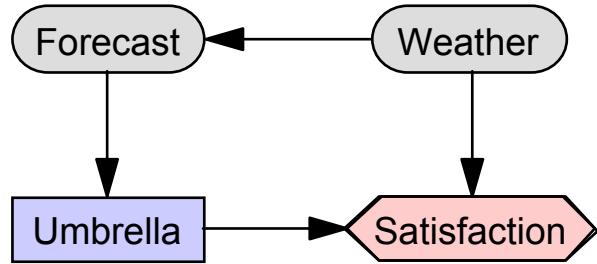
Chapter 2 discussed using belief networks for probabilistic inference. Belief networks are used to determine new beliefs (in the form of probabilities) as observations are made or facts are gathered. They are composed only of *nature nodes*. To form a *decision network* (also known as an “influence diagram”), you add *decision nodes* and *utility nodes* (also known as “value” nodes). Decision nodes, which are traditionally drawn as rectangular boxes, correspond to variables or events that you will be able to control. Utility nodes are drawn with a diamond or flattened hexagon shape, and correspond to quantities you want to maximize.

The decision network as a whole represents a planning problem that you face, or that some other agent, often called “the decision maker”, faces. Netica can find values for the decision nodes that will result in the largest possible expected value for the utility node (or sum of them if there is more than one). This is known as “solving” the decision network. Once the network is solved, you (or the decision maker) can begin to enact the plan by taking the prescribed action for each decision as it arises.

Links that go into a decision node have a special meaning and are called *informational links*. They indicate what will be known at the time the decision is to be made. That is, the decision maker will know the values of all the nodes which have links into that decision node, and will not know the values of any other nodes. The previous paragraph stated that when solving a decision network we find values for each of the decision nodes. But if there are some links entering a decision node, we actually find a decision value for each possible configuration of values of the parents. This is a contingent plan, with a form like: “if I observe A=high and B=no, then I will do X, if I observe A=low and B=yes, then I will do Y, and so on. So, solving a network finds a *decision function* for each decision node (i.e. a function which gives a decision value for each possible set of parent values). If there are a number of decision nodes, possibly corresponding to decisions made at different points in time, then solving the network will find a decision function for each of them, and this set of decision functions is known as a *policy*. It is a

full conditional plan, specifying what to do in each possible contingency, based on the information that will be available.

As an example, let's consider a very tiny decision network from Ross Shachter known as "Umbrella" (located in the Examples folder). It has 2 *nature nodes* representing the weather forecast in the morning (sunny, cloudy or rainy), and whether or not it actually rains during the day (rain or no_rain). It has a *decision node* of whether or not to take an umbrella, and a *utility node* that measures the decision maker's level of satisfaction. There is a link from Weather to Forecast capturing the believed correlation between the two (perhaps based on previous observations).



There is a link from Forecast to Umbrella indicating that the decision maker will know the forecast when he makes the decision, but no link from Weather to Umbrella; if he knew for certain what the weather was going to be, it would be easy to decide whether or not to take the umbrella. There are links from Weather and Umbrella to Satisfaction, capturing the idea that he is most happy when it is sunny and he doesn't take an umbrella (utility = 100), next most when it is raining and he takes an umbrella (utility = 70). He hates carrying an umbrella on a sunny day (utility = 20), but is most unhappy if it is raining and he doesn't have one (utility = 0).

6.1 Constructing a Decision Network

Constructing a decision network is pretty much the same as constructing a belief network, so most of the material in chapters 2, 3, and 4 is applicable. One difference, obviously, is that you must designate some nodes as decision and utility nodes. Usually you do this when you first add them, by using the tool bar button with the blue rectangle to add decision nodes, and the one with the green hexagon to add utility nodes. However, you can change node kinds using the node dialog (described in chapter 4) obtained by double-clicking on a node. You can also change node kinds by selecting the nodes you wish to change, and then clicking on the tool bar button for the desired kind, while holding down the <Ctrl> key.

You do not have to enter a relationship for decision nodes, since that is what is found when the decision network is solved. Utility nodes should always be made continuous, and do not need to be discretized. Also, there must be only one utility node (in this version, but not the forthcoming version of Netica), and it must not have any children.

6.2 No-Forgetting Links

Normally when a decision maker makes a later decision, he will know at least everything he did when he made earlier decisions. Also, he will know what earlier decisions he made. That is represented in the decision network by ensuring that the decisions are in a sequence, with a link from one to the next, and that each decision node later in the sequence has links to it from all the decisions earlier, and from all the parent nodes of earlier decisions. These links are called *no-forgetting* links, since they indicate that all the information that was available earlier is still available to the decision maker.

Not considering computation costs, having more information to make decisions always leads to a policy having greater (or equal) expected utility. So, to find the best policy, it is important to have no-forgetting links. After you have constructed a network, you can make sure it has all the required no-forgetting links by doing a **Network → Add No-Forgetting Links** menu command. Make sure you have added links so that there is a path running through all the decision nodes (to fix their sequence in time) before doing the command.

There are a couple of reasons why you might not want all the no-forgetting links included. With them the decision tables of the last nodes may become gigantic, requiring a great deal of computation time and memory. In such a case it is useful to be able to forget some items, especially if the decrease in expected utility is small or none. The other reason you may not want all the no-forgetting links is that you may be trying to model a decision situation where you have two or more co-operating decision makers who are forming a policy together, but who won't be able to communicate when the decisions are to be made. Netica currently requires all the no-forgetting links to be included in a decision network before it can solve it, but the next version won't (it will operate on the theory explained in Zhang&QP94).

6.3 Solving a Decision Network

Once the network is built, you can solve it using the **Network → Optimize Decisions** menu command. It will attach to each decision node a deterministic function (the optimal relationship is always deterministic). This function provides a value for the decision node for each possible configuration of parent values. Since the links into a decision node indicate what the decision

maker will know when he is about to make the decision, this function provides a decision for each possible information state. These decisions are the ones that maximize the expected value of the sum of the utility nodes.

After you do an **Optimize Decisions** command, and Netica has finished optimizing, it will pop-up a node relation dialog showing the decision functions for the first decision node. In general, you can use the node selector in the upper left to see the decision functions for other nodes. For the umbrella example, the decision will be:

<u>forecast</u>	<u>best decision</u>
sunny	don't take umbrella
cloudy	don't take umbrella
rainy	take umbrella

Decision networks can also be solved using node absorption, which is discussed in section 8.3. Select all the nodes and then click on the absorb tool bar button. The network will be decomposed and only the decision nodes and utility node will be left. Each decision node will have the optimal decision as its relation, but it may have some parent links removed. Any removed links are ones that are irrelevant to the decision. All the utility node's parents will be removed, and it will have the maximized expected value as its relation. After observing the relations (and perhaps copying and pasting them to a new network), you can restore the original network with an **Edit→Undo** menu command.

If you wish to see an example of a slightly more complicated decision network, with 2 sequential decision nodes, open “Car_Buyer_Neapolitan” in the Examples folder. It is described in Neapolitan90, and is based on the classic “Car Buyer” influence diagram. It involves two sequential decisions about whether to do some tests and then whether to buy a certain used car. The optimal decisions turn out to be not to do any tests ($D = \text{none}$), but to buy the car ($B = \text{Buy}$ when $D = \text{none}$).

6.4 Model Iteration

For some applications of decision networks (especially in areas like gambling, control or automated decision making), it is relatively easy to set up the relations to suitably reflect reality, and the decisions that the network makes are much better than decisions you would make unaided, because of Netica's superior ability to reason precisely with probability, and to deal with many interactions in very complex situations.

However for other applications (especially in areas like business decision making or public policy), the decisions that Netica makes may not match your “gut feelings”, and you may suspect that Netica is wrong. In that case, usually the decision network doesn't capture the decision

problem well enough. The usual procedure is to go back to the decision network model, change it to more accurately reflect reality, and solve it again. Then repeat the process until you are satisfied with the results.

But, of what use is the decision network if all you do is try to get it to match the decisions you would make anyway? Professional decision analysts who use decision networks on a regular basis tell us that they are invaluable for disciplined decision making. The process of building the decision network clarifies the problem (often slowly changing your gut feeling as you proceed), and the iteration process forces you to re-examine assumptions you have made. In a group decision making environment, constructing a decision network is often of great benefit in fleshing out the differences in people's beliefs and values, and allows people to discuss specifics rather than arguing in generalities.

Also, once the policy is formed, the decision network acts as a living document of the beliefs and values that lead to it. If those change over time, the decision network can be changed to generate a new policy. Or parts of it can be copied and pasted into new networks for new decision problems. If a policy results in a bad outcome, it is possible to go back and determine whether it was based on bad information, and if so, to change that information for the next decision making situation.

6.5 What-if Decision Analysis

With Netica's 'undo' feature it is easy to solve a decision network, undo it, change a utility value or a node relationship, and then re-solve the network to see how the optimal decisions and maximum expected utility are affected.

But what if you want to force a decision node to have a certain decision function, and see how the other decisions and maximum expected utility change? Just enter the desired decision function for the node in its relation dialog box (or paste it in, since you are often obtaining it from a previous solution of a decision network), select the decision node, and change it into a nature node by clicking on the gray ellipse toolbar button while holding down the < Ctrl > key. Then solve the network in the usual way, and observe the changes to the other decision functions and the change to the maximum expected utility. Of course you can use 'undo' to return to the original situation. In general, when you want to fix a decision node to a certain decision function while you continue your analysis, you temporarily change it into a nature node.

7 Network Libraries

Often the probabilistic relation between a node and its parents represents a small piece of local knowledge which may be applicable in a number of different networks to be used in different situations. That relation may have been learned from data, or entered by an expert. Each new network it is placed in captures the global relations between such local pieces of knowledge, and belief updating combines the local and global knowledge with the details of some particular case.

For example, suppose that you made a simple network consisting of a node called Weather connected to a node called Forecast. You could put the link between them either way, since in this situation you can't really capture causation (they are both caused by other variables, such as the weather at earlier times), but say you put the link from weather to forecast because often its better to put links from more immutable to less immutable variables. You could learn its probabilities (as described in chapter 10) from a set of cases consisting of the forecast and what the weather turned out to be. Then you could put it in a library to later graft into networks for inference involving the weather and its forecast, such as the decision problem discussed in chapter 6.

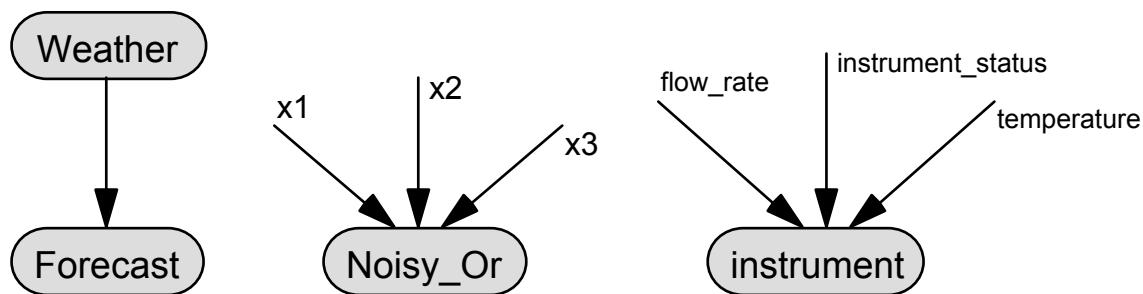


Figure 7.1 - Example of a tiny network library.

As another example, suppose you have a device for measuring the flow rate in a pipe. It produces biased readings depending on the ambient temperature, and it can malfunction in a few

different ways, each of them producing wrong or inaccurate readings. You can model the device with a 4 node network, consisting of one node for the reading on the device, and 3 parent nodes corresponding to: actual flow rate, ambient temperature, and device status (okay, broken-1, broken-2, etc.). You enter the probabilistic relationship, and then you disconnect the node from its parents and place it in a library. The rightmost node in diagram 7.1 shows how it will appear. A section below describes in more detail how to do this.

Later, if you have a network to model a situation in which you use the instrument to make two measurements of the flows in two connected pipes located in the same room, you just duplicate the device characteristics node from the library twice into the new network, and graft it to the appropriate nodes in that network (see diagram 7.2). Note that if the ambient temperature could be different between the two measurements, then the `room_temp` node would appear as two connected nodes, similar to the flow nodes, and the same goes for the `instrument_status` node if the device may have broken between measurements.

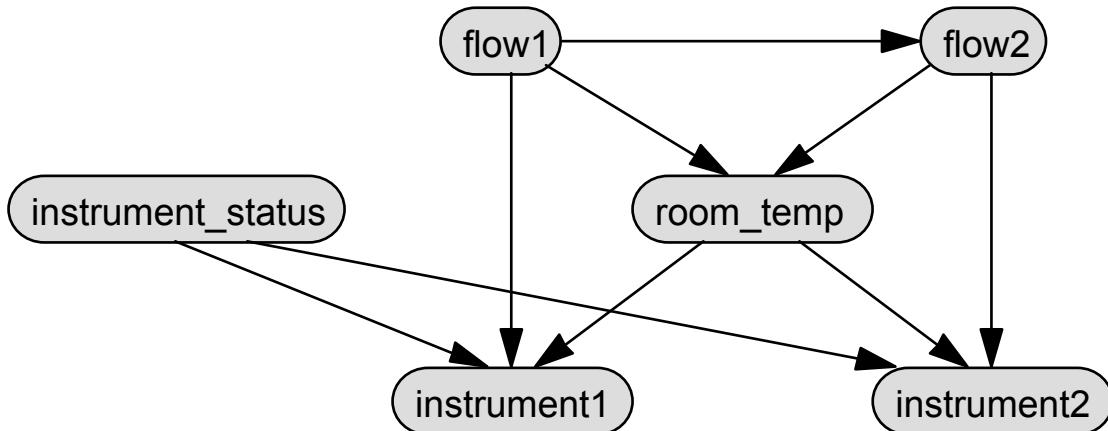


Figure 7.2 - A network which was built to analyze an instrument measuring the flows in 2 different pipes in the same room, using the “instrument” node from the library depicted in figure 7.1.

7.1 Disconnecting and Reconnecting Links

A link may be disconnected from its parent node, without the link actually being removed. That means that the child node can maintain the information on the conditional probability relation it had with its parent, without actually being connected to the parent. The intention is that later it will be reconnected to the parent, or more likely to some other node, before it is used for inference.

To disconnect some links, select them and then click on the knife tool bar button, or do **Modify** → **Disconnect Links**. Each link will turn into a short stub connected to its child node, and

labeled with the name of the link. If you have not previously named the links (see section 4.10.3), then Netica will name them with the names of the parent nodes. The disconnected links will look similar to the links of the rightmost node in figure 7.1. If you want to disconnect all the links entering a node, select the node and do a **Modify**→**Disconnect Links**.

If you want to disconnect a link while maintaining its shape, first click on the link to select it. Then click down on the hilited box at the end of the link (the end without the arrow), and drag it away from the parent node. When you release the mouse button you will know that you have dragged far enough if the name of the link appears beside it. All of the bends of the link will be maintained.

To reconnect a disconnected link to a new node, first click on the link to select it. It will be outlined with the hilite color, and there will be a box of hilite color at the disconnected end of the link. Click down in that box, drag it over the node that you want the link connected to, and then release the mouse button. If the connection was successful the name of the link will disappear. If the node is not compatible with the original parent (e.g. it has a different number of states), then an error message will be displayed. You may disconnect a link from one parent and connect it to a new parent in one motion if you wish.

7.2 Making and Using Network Libraries

Netica makes it easy to maintain libraries of disconnected nodes and subnetworks. To make a new library, just use the **File**→**New Network** menu command. Libraries are handled internally in the same way as regular networks. Nodes and subnetworks can be copied to it using **Edit**→**Copy** and **Edit**→**Paste**, which can transfer material from one network to another, and also copies all the links between nodes in a subnetwork. When a node is being duplicated, but one of its parents isn't, then the duplicated node will have a disconnected link where that parent was.

To use nodes in the library, you use **Edit**→**Copy** and **Edit**→**Paste** again, this time to duplicate from the library into the new network. Then you connect up any disconnected links as described in the section above, before compiling the network or using it for inference.

For example, to create a library with just the “instrument” node of figure 7.1, first you would make a network with the 4 nodes: flow_rate, temperature, instrument_status and instrument. Put links from nodes flow, temperature and instrument_status to instrument. Enter a probabilistic relationship for the node “instrument”. Now make a new network with **File**→**New Network**, select the “instrument” node in the original network, do an **Edit**→**Copy**, click in the new network, do an **Edit**→**Paste**, then a **File**→**Save**. You now have file which is a library with a single node in it.

At a later session, you can use the library to construct a network in which the instrument is used to measure the flows as described at the beginning of this chapter. Make a new network with **File → New Network**, add the nodes flow1, flow2, instrument_status and room_temp, link them together as shown in figure 7.2, and enter the probabilistic relationships between them. Then use **File → Open** to open the library file with the instrument node. Select that node, do an **Edit → Copy**, then click in the application network where you want the instrument node to appear. Do an **Edit → Paste**, click again where the other copy of it should be and do another **Edit → Paste**. Finally, hook up each of instrument’s disconnected links to their appropriate nodes using the method described in the previous section.

Now the application network is ready for probabilistic inference (you can do a **Network → Compile** menu command). Perhaps you have positive findings for the “instrument” node (i.e. what you read from its dial), and you use them to determine flows and their uncertainties in a way that properly accounts for random (uncorrelated) and systematic (correlated) errors, as well as all the background knowledge about the situation.

As a sister product, Netica Programmer’s Library (API) can be useful for automating the process of constructing networks that are composed of nodes or subnetworks from a library, perhaps by using templates or rules.

8 Transforming a Network

There are certain ways that Netica can transform a network model which modifies its representation without modifying its meaning. Netica can remove nodes and reverse the direction of links in such a way that any inference done with the resulting network yields precisely the same results as the original network (except of course findings can't be entered for removed nodes, and their resulting beliefs are unavailable).

Some reasons to do these transforms are to simplify the network, apply the network to more specific problems, gain understanding of the network or of the real world relations, or put the network in a form for easier probability or function assessment.

8.1 Link Reversals

Suppose P and C are discrete or discretized nodes within a network, and that P is a parent of C. Since there is a link going from node P to node C, the relation between the two nodes is expressed as probabilities for the states of C, conditioned on the state values of P (or a function providing C's value in terms of P's value if the node is deterministic). But sometimes you might want to know what the probabilities for the states of P are, conditioned on state values of C.

In order to do this, you *reverse the link* from P to C. When that link is reversed, the other links and the probability tables of the child and parent nodes are adjusted in such a way that any probabilistic inference done after the reversal will yield exactly the same results as before the reversal. In other words, the full joint probability distribution of the network does not change when a link is reversed. The global relationship between the nodes remains the same; just the local expressions of it have been changed. It is a probabilistic generalization of function inversion.

During the reversal, links may be added from the parents of the parent node to the child node, and from the parents of the child node to the parent node, which will increase the complexity of

the network. Technically speaking, all links added will be confined to the parent node and its Markov boundary. When links are added, the size of conditional probability tables may grow significantly, and sometimes enormously. The size of the tables is the product of the number of states of all the parents, so the size of the tables can grow exponentially. When link reversals result in a node having many parents, the operation may be slow, or Netica may report that there is not enough memory available.

Occasionally, during a reversal, links from the parents of the parent node to the child node, and from the parents of the child node to the parent node will disappear, resulting in a simpler network. For example, reversing the link from P to C might result in additional links going into P and C, but when the link is reversed again (so that it has its original orientation) those links will disappear, resulting in the original network (providing all the nodes have nondegenerate probability tables, and that rounding inaccuracies are small).

Often a network which is simpler due to the direction of its links provides a better model of the world. For example, it usually represents true causality more accurately, it may be better at generalizing, and of course it allows for faster computation. Sometimes link reversals can be used to search for more simple networks, given a network that was originally learned from data. In that case Netica might not always automatically remove links that should be removed, because although these links will be weak, they will not be completely ineffectual (perhaps because the probability tables learned are not “exact”). Links are termed *weak* when their removal has little effect on the full joint probability distribution, and therefore little effect on inference results (see Boerlage94). You would have to remove these weak links by hand.

To reverse a link you select it and then do **Modify**→**Reverse Links**, or click on the link reversal button, which is the tool bar button with two arrows pointing in opposite directions. If a node is selected when you click on the link reversal button, it will do all the link reversals necessary to make all links involving the node point to it, unless the node has a finding, in which case they will all be made to point away from the node. If all links are made to point away from the node, then many extra links may have to be added between the ancestors of the node (not just its Markov boundary), and Netica may report that there is not enough memory available..

If you have several links to reverse, you can select them all (e.g. by shift-clicking on them), and then click on the link reversal tool bar button. The amount of time and memory required to reverse a set of links depends greatly on the order in which they are reversed. Unless you know a good order to do the reversals, you should reverse them all at once rather than one-by-one, so that Netica can choose a good order to do them.

8.2 Node Absorption

Node absorption is the process of removing nodes from a belief network or decision network, and making any necessary adjustments to the resulting network so that any inference done with it yields the same results as before the nodes were removed (except of course you can't interact with the removed nodes). The local representation is changed, but the global relationships are not changed. In probability theory this is sometimes loosely called "summing out a variable". It leaves the full joint probability distribution of the remaining nodes unchanged.

As an example, suppose you have a large network that has been constructed over time by a combination of expert assistance and probability learning. It shows the relationships between hundreds of variables, and contains much valuable information that could be used in a number of different applications.

Now you want to use it in an application where only 10 of the variables will be of interest. In every query of the new application, a particular 4 of these 10 will always have the same findings. For example, one of the nodes in the original network might be Gender, and in the restricted application the network will only be used for females, so you would like to enter a permanent finding of 'female' for the Gender node. These nodes are called *context nodes*. In each of the queries, you will be receiving new findings for 4 other nodes, and then you want the resulting beliefs of the remaining 2 out of 10. The nodes that will always have new findings are called *evidence nodes*, and those whose beliefs you may want are called *query nodes*. The hundreds of other nodes might be involved in intermediate calculations, but you don't care about their values explicitly.

You can simplify the large network down to one with just 6 nodes using a process called *node absorption*. In this process nodes are removed from the network, but the links and probability tables of the remaining nodes are adjusted in such a way that any probabilistic inference done with the remaining nodes will yield exactly the same results as if the absorbed nodes were present. Node absorption can simplify a network considerably, since nodes are being removed, but in some cases it can make the network more complex because extra links are added.

Here is how you would reduce the large network. First enter the permanent findings for the context nodes in the usual way of entering findings (as described in section 2.5). Then select all the nodes to be absorbed (i.e. all the nodes except the evidence and query nodes), and click on the absorb tool button, which is the tool bar button with the five pointed star (or do a **Network→Absorb Nodes** menu command). The selected nodes will be removed, and some links may be added and/or reversed.

If you want, you can absorb the nodes a few at a time, by selecting each group and clicking on the absorb tool button. The final result of absorbing a set of nodes is not dependent on the order in which they are absorbed, but the time and memory required to do the absorbing may be greatly affected. If you have a set of nodes to absorb and you don't know a good order to absorb them, then it is best to absorb them all at once, so that Netica can pick a good order.

Returning to the example, the resulting 6 node network will give the same inference results as the original large one, for the restricted queries you will be making. If you are guaranteed that there will always be findings for every evidence node, then you can then further simplify things by removing any links that go from evidence node P to evidence node C, providing C does not have a query node as a parent. This means that if you can reverse links (as described in the previous section) to make all the evidence nodes ancestors of all the query nodes, then you can remove all the links between the evidence nodes. Any evidence node that is left completely disconnected by this operation is irrelevant to the query, and can be deleted. And now you can examine the conditional probability relations of the query nodes to see directly how they depend on the findings. You may just be able to look up the desired probabilities without doing belief updating at all!

There is a danger to keep in mind. Even though the reduced network has fewer nodes than the original, internally it may actually be more complex, sometimes much more complex, if many links were added during node absorbing or link reversing (remember that the size of a node's conditional probability table can be exponential in its number of parents). Generally speaking, absorbing out context nodes (i.e. nodes with findings entered) which have many ancestor nodes results in the worst increase in complexity. The next worst is absorbing out non-context nodes (i.e. nodes with no findings) which have many descendant nodes. Absorbing out context nodes with no ancestors, or non-context nodes with no descendants, will not add any links. Of course, if the number of query and evidence nodes is very small, the resulting network must be simpler, although the transformations to generate it might temporarily require a lot of memory.

8.3 Probabilistic Inference by Node Absorption

From the previous section you may have realized that it's possible to do probabilistic inference using node absorption, by entering all the findings, and then absorbing all the nodes except for a single query node. The resulting probability "relation" for that node will be a single belief vector (because the node won't have any parents), that is the same as the belief vector that would be obtained by compiling the belief network and doing belief updating. Of course, the network is destroyed in the process, but you can recover it by doing an **Edit → Undo** command. Normally it is better to do inference by compiling the network and doing belief updating as

described in chapter 2 “Probabilistic Inference”, but sometimes additional insights are gained by using node absorption for inference.

It should be mentioned that node absorption will also work with decision networks (see chapter 6 “Decision Problems”) to find optimal decisions. To solve a decision network, select all its nodes, and then do a node absorption command (click on the star tool button). The nature nodes will all be absorbed out. When a decision node is absorbed, it is not removed from the network; instead it is completely disconnected and its decision table set to the optimal decision function. Utility nodes are also left, so you can see the expected utility. The algorithm used is described in Shachter86, Shachter88 and Shachter89.

When using node absorption to solve decision problems, the decision nodes must have no-forgetting links, which are described in section 6.1. Also, there must be only one utility node, with no descendants, and if the nodes to absorb do not include all the nodes in the network, they must consist of a descendant subnetwork. So, if nodes are absorbed one-by-one, a suitable order must be used. These restrictions are explained in more detail in the Shachter references mentioned above. If any of these restrictions are not met, Netica will not produce an erroneous result, but will just absorb as many of the nodes as it can, and then display a message explaining why it was impossible to proceed.

9 Cases and Case Files

The set of all findings entered into the nodes of a single belief network is referred to as a *case*. A case usually provides some information about a particular object, person, event, etc. It may be transferred from the belief network to a file for later retrieval. Perhaps you want to work with a different case, but you need to be able to restore the original case to the network later on, in order to enter new findings recently obtained, or to make further queries.

To save to file all the positive findings from the belief network in the active window, make sure no nodes are selected, and then do a **File → Save Case As** menu command. You will be presented with the standard dialog box to choose the file name. After you enter it and press Okay, all the positive findings will be saved to that file. If some nodes are selected, then only the findings for the selected nodes will be saved (and Netica will beep and put a notice in the Messages window).

If you later want to read the case back in to its original belief network, you first make sure that the original belief network is the active window and no nodes are selected, and then do a **File → Get Case** menu command. You will then be presented with the standard dialog box for choosing a file, from which you can select the case file. After you press Okay, any existing findings in the network will be removed, the file will be read and the findings entered into the network. If some nodes are selected, then Netica only removes findings and reads new ones for the selected nodes (it also beeps and puts a notice in the Messages window that the whole case wasn't read). If the network is auto-updating, then belief updating will be done automatically to account for the findings of the case.

Continuing with the medical example started earlier, a case might correspond to a certain patient. When you want to work with a new patient, you save all the information gathered for the first patient into a case file before removing it from the network, perhaps using the patient's name as a file name. When it comes time to reconsider the first patient - perhaps some lab results have arrived - you just read that person's case file.

For another example, each case could be a political riding. The findings would be details about that riding (such as demographics, poll statistics, etc.), and the belief network could be used to predict the percentage of votes each political party will get in the next election. As new information about a riding arrived, its case file would be kept updated.

It is possible to read a case into a different network than the one it was originally saved from. Findings from nodes of the old network will be entered into nodes of the same name in the new network (the titles of the nodes are ignored). The state names of the nodes (if present) should also be the same. Any findings not corresponding to a node in the new network will simply be ignored. All name comparisons are case-sensitive.

Real number values that you have entered as findings for discretized continuous nodes are saved in case files, rather than just the state they correspond to. This enables reading the case into another network whose node for that variable has been discretized in a different way (perhaps finer, or courser).

If you read in a case, and the case file has a value that isn't any of the states of the corresponding node in the network, then an error message will be displayed. For example, if node 'color' has the states 'red' and 'green', and in the case file the value for color is 'blue', the message will be displayed. An exception to this occurs if one of the states of the network node is named 'other'. Then the case will be read without error, and the finding for the node will be 'other'.

To remove the current case from a belief net you can use the **Network → Remove Findings** menu command, with no nodes selected. It will leave the network with no findings entered, and if it is an auto-update network, then its beliefs will be updated to reflect that. If some nodes are selected when you do the **Remove Findings** command, then only the findings for those nodes will be retracted.

9.1 Multi-Case Files

The case files described above consisted of a single case, but it is possible to store many cases in one case file. These case files act as databases; they may be used to swap cases in and out of a network as additional findings are obtained or beliefs required, to transfer a case from one network to another, or as data to learn a new network.

There are several different ways you can make a file of cases. One possibility is to use a text editor or a word processing program and manually construct it according to the specification in the section below (be sure to save it as a "Text Only" file if you use a word processing program). If the data is in a spreadsheet program, usually you can just copy the data from the spreadsheet to

the clipboard, and then paste it into the text editor window, and it will come out in the required format. You will have to add the two header lines described in the section below.

If you wish to create a multi-case file consisting of random cases sampled from a particular distribution, Netica can make it as described in section 9.3 “Generating Random Cases” below.

Another way to create a multi-case file is to write a computer program that creates it. Using the Netica API Programmer’s Library may be especially useful for this purpose, since it can create such a file with just a few function calls. It can also be used to update and otherwise maintain case files, and to read them and do belief network learning and inference using them.

9.2 Case File Format

Case files (single-case or multi-case) are pure ASCII text files. They must contain “// ~->[CASE-1]->~” somewhere in the first 3 lines. Then comes a line consisting of headings for the columns. Each heading corresponds to one variable of the case, and is the name of the node used to represent the variable (sometimes the variables are called *attributes* and the entries in the column *values*, i.e. *attribute-value*). The headings are separated by spaces and/or tabs (it doesn’t matter how many).

The case data is next, with one case per line (a single-case file would only have one such line). The values of the variables are in the same order as the heading line, and are separated by spaces or tabs (the columns don’t have to “line up” as they do in the example files below). The value of a discrete variable is given by its state name, or if it doesn’t have a state name, then by its state number preceded by a ‘#’ character (the first state is #0). Using the state names is preferred, since the order of the states may be changed some time, and that would render a file with state numbers invalid.

The value of a continuous variable is given by a number in integer, decimal, or scientific notation (e.g. -3.21e-7). If it has been discretized, then the value may be given by a state name or state number instead, but the continuous number is preferred if it is available. That way the case file can be used for different discretizations of that variable in the future. Try to use the correct number of significant figures, since future versions of Netica may use this information.

If the values of some of the variables are unknown for some of the cases, then an asterisk (“*”) is put in the file instead of the value (this is known as “missing data”). A single-case file is the same as one with multiple cases, except it just has 1 case. There may be as many spaces or tabs at the end of a line as desired, and there may also be C or C++ style comments (i.e. a double slash “//”, followed by any text).

There are two special columns that a file may have which don't correspond to nodes. One provides an identification number for each case, which must be an integer between 0 and 2 billion. The heading for this column is "IDnum". Identification numbers do not have to be in order through the file. The other special column has the heading "NumCases", and indicates the frequency or multiplicity of the case. A multiplicity of m indicates m cases with the same variable values. It is not required to be an integer, so it can be used to represent a frequency of occurrence if desired. The missing data symbol ("*") should not appear in either of these columns.

As an example of a case file, here is a listing of "asia.cases" which is produced by the example in the section below (the case file you obtain may be a little different, since random numbers are involved). It has an IDnum column, but no frequency column.

```
// ~-->[CASE-1]->~
```

IDnum	VisitAsia	Tuberculosis	Smoking	Cancer	TbOrCa	XRay	Bronchitis	Dyspnea
1	No_Visit	Present	Smoker	Absent	True	Abnormal	Absent	Present
2	No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
3	No_Visit	Absent	Smoker	Present	True	Abnormal	Present	Present
4	No_Visit	Absent	NonSmoker	Absent	False	Normal	Absent	Absent
5	No_Visit	Absent	Smoker	Present	True	Abnormal	Present	Present
6	No_Visit	Absent	Smoker	Absent	False	Abnormal	Present	Present
...								
119	No_Visit	Absent	Smoker	Absent	False	Normal	Present	Present
120	No_Visit	Absent	Smoker	Present	True	Abnormal	Present	Present

Here is another example of a case file, this time for cars brought into a garage (notice BatAge, which is a continuous variable):

```
// ~-->[CASE-1]->~
```

Starts	BatAge	Cranks	Lights	StMotor	SpPlug	MFuse	Alter	BatVolt	Dist	PlugVolt	Timing
False	5.9	False	off	*	fouled	okay	*	dead	*	*	good
False	1.3	False	off	*	okay	okay	*	dead	*	none	bad
False	5.2	False	off	Okay	okay	okay	Okay	dead	Okay	none	good
True	4.1	True	bright	*	okay	okay	*	strong	Okay	strong	*
True	2.7	*	bright	*	wide	okay	*	strong	Okay	*	*
*	*	True	bright	*	fouled	okay	*	*	Okay	strong	good
False	1.7	True	off	Okay	okay	okay	Okay	dead	*	none	good
True	2.9	True	bright	*	*	*	*	strong	Okay	strong	*

Future versions of Netica will support more advanced operations with cases, including a more efficient file representation, and a way of using belief networks as "indexing functions" to do the kind of lookup common in case-based reasoning. However, the above described type of file format will always be supported as well.

9.3 Generating Random Cases

You can use Netica to generate a series of random cases whose probability distribution matches that of a particular belief network, which is sometimes known as *sampling*. These cases can be

used as example scenarios of what one should expect if the belief network matches reality. Or they can be manipulated and combined with other cases, and then used to learn a new network.

The cases will be stored in a file whose format matches the specification of a case file as described in the previous section. The sampling algorithms used are precise, so that the long range frequencies of the cases will exactly approach the probabilities of the belief network. If you have used an equation to define the relation between a node and its parents, then that equation will be used to generate the random cases, not any probability table which approximates the equation. Continuous variables which have been discretized will be provided with a continuous real number in each case, not just a state representing a range of values.

To generate the case file, make sure the belief network to be used is the active window, select the nodes for which you wish to have values in the case file, and then do a **Network → Simulate Cases** command. All the nodes of the network will be used to generate the cases, but columns will only be made for the selected ones. You will be queried for how many cases to generate, the file name for the case file, where to put it, and how much “missing data” you want. Normally you will enter 0 for the amount of missing data, but if you want to have a case file with asterisks for some fraction of the fields, enter that fraction (e.g. entering 0.25 means 25% of the values will be missing).

As an example, if you do a **Network → Simulate Cases** command with ‘Asia.dne’ from the Examples folder, and enter 120, “asia.cases” and 0 to the query dialogs, then you will obtain a case file similar to that shown in the preceding section.

10 Learning from Cases

Belief network learning is the automatic process of determining a suitable belief network, given data in the form of cases. Each case represents an example, event, object or situation in the world (presumably that exists or has occurred), and the case supplies values for a set of variables which describe the event, object, etc., as described in the last chapter. Each variable becomes a node in the learned network (unless you want to ignore some of them), and the various values it takes on become that node's states. Some cases may not have values for some variables that other cases do, which is known as *missing data*.

The learned network can be used to analyze a new case drawn from the same population as the cases used for learning. Typically the values for some variables of the new case will be known. These are entered as *findings*, and then probabilistic inference is done to determine beliefs for the values of the rest of the variables for that case. Sometimes we aren't interested in values for all the rest of the variables, but only some of them, and we call the nodes that correspond to these variables *query nodes*. If the links of the network correspond to a causal structure, and the query nodes are ancestors of the nodes with findings, then you could say that the network has learned to do diagnosis. If the query nodes are descendants, then the network has learned to do prediction, and if the query node corresponds to a "class" variable, then the network has learned to do classification. Of course the same network could do all three, even at the same time.

The belief network learning task has traditionally been divided into two parts: structure learning and parameter learning. *Structure learning* determines the dependence and independence of variables and suggests a direction of causation, in other words, the placement of the links in the network. *Parameter learning* determines the conditional probability relationship at each node, given the link structures and the data. Currently Netica only does parameter learning (i.e., you link up the nodes before learning begins).

While learning, Netica assumes independence between each of the conditional probabilities of a node's relation to its parents. This works very well when there is lots of data (or nodes have few parents), but can result in under-confidence and poor generalization otherwise . Netica may not

have seen some parent configuration, so the learned probability distribution is uniform for it, but Netica might do better if it assumed that parent configuration behaved similarly to other similar parent configurations (i.e., assumed a little more independence between variables, and a little less independence between conditional probabilities).

A future version of Netica will make it possible to not assume independence of the conditional probabilities, if that is desired. Netica will be able to learn relations from a set of special models. It will also be able to suggest changes to the structure, thereby partially doing structure learning.

10.1 How To Do It

There are two ways to learn probabilities from cases: singly (one-by-one), or in batch mode from a file of cases.

10.1.1 Single Case Learning

To learn from a single case, you must first have a network constructed. Nodes in the network may already have their conditional probability relations, which you entered manually or previously learned, and which you now want to improve using learning. On the other hand, there might not be any conditional probability relations, and you want to learn them from scratch. For information on constructing the network, see chapters 3 and 4.

If the case is not already in the belief network, you enter it into the network as findings (see chapter 2). Only positive findings will be used; negative and likelihood findings will be ignored. Then you select the nodes whose probabilities you would like to have revised to account for the case, if possible. Usually you would like all possible nodes to have their probabilities revised, so you would select all the nodes (or don't select any nodes, which is equivalent). Then click on the tool bar button having an arrow pointing from the yellow case symbol to the green relation symbol (inverted V's), or do a **Relation → Incorporate Case** menu command. You will be queried for a "degree", which is normally 1. By making it 2, you can achieve the same effect as learning the same case twice, and equivalently for other numbers. By making it -1, you can exactly unlearn a case that was earlier learned with degree = 1, and so on for other negative numbers. Don't try to unlearn cases that were never learned, or to unlearn them with greater degree than they were learned. During the learning, selected nodes for which the case provides sufficient data will have their probabilities revised a small amount to account for the case, and their appropriate experience levels increased slightly (as described in section 10.2 below).

10.1.2 Learning From a Case File

The batch mode way of revising probabilities does exactly the same thing as the one-by-one way, but for a whole file of cases at once. See chapter 9 “Cases and Case Files” for information on creating a file of cases. Before learning begins, you must have a belief network constructed whose nodes are the variables (i.e. attributes) of the cases. It is okay if it has additional nodes related or unrelated to the cases in the file.

If you don’t already have a network constructed, or the network you have doesn’t include all the variables in the case file that you wish, you can use the **Modify → Add Case File Nodes** menu command. It will scan through a case file and add to the current network new nodes for any variables that it discovers in the case file that aren’t already in the network. The states of the new nodes will be all the possible values discovered from the case file. If your network already has a node with the same name as some variable from the case file, but that node doesn’t have all the states that are mentioned in the case file for that variable, then those states will be added to the node (unless the node has a state called ‘other’). After Netica has added all the nodes, you move them to the positions you want, delete any that you aren’t interested in, and then add links between them and to or from any other nodes in the network to capture the dependencies that you wish.

When you do a **Relation → Incorp Case File** menu command, Netica will ask you for a case file and a degree. It will then do the same thing as the one-by-one method described in the preceding section for each of the cases in the file. The degree that you supply will be applied to each case of the file in the same way as described in the preceding section, except if the case file has a NumCases column (see section 9.2), then the degree will first be multiplied by the multiplicity from that column. As Netica processes the cases, it reports its progress in the Messages window.

10.2 Experience

There has been considerable controversy over the best way to represent uncertainty, with some of the suggestions being probability, fuzzy logic, nonmonotonic logic, belief functions, Dempster-Shafer, etc. Currently probability and fuzzy logic are the most practical methods for most applications. Of these two, probability has a much sounder theoretical basis (at least with respect to the way they are actually used). However, a deficiency of using only probability is the inability to represent ignorance in an easy way.

As an example, suppose you had to draw a ball from a bag full of black and white balls and you couldn’t tell how many white balls and how many black balls there were in the bag. If you had

to supply a probability that you were going to draw a white ball, it should be 0.5, providing you had no additional information.

Contrast this with the case where you can count the balls in the bag beforehand (there are 10 of each), and you will shake the bag before you draw. In this situation the probability of drawing a white ball is 0.5, but whereas in the first case you were in a state of ignorance, now you feel much more informed.

If you needed to do probabilistic inference or solve decision problems as in the previous chapters, then the 0.5 probability would be sufficient in either situation. In both situations you should believe and act as if there was an equal chance of drawing a white or a black ball. So the concept of experience is not required for these types of problems, and you do not need to be able to represent ignorance (ignorance is the endpoint of the experience spectrum). However, for learning and communicating knowledge, it is useful to be able to represent the degree of experience as well as the probability, as we shall see.

Suppose you are now going to sequentially draw a number of balls from the bag. If you drew 3 white balls in a row, then in the first situation your probability that the next ball will be white should be greater than 0.5, because you are learning (perhaps incorrectly) that there seem to be a lot of white balls. In the second situation your probability of the next ball being white should be less than 0.5, because you know that now there are more black than white balls left in the bag.

One way to handle this using just probabilities is to keep track of your beliefs about the ratio of white to black balls in the bag. Then you will have many probabilities, one for each possible ratio. Each of these probabilities will change as you draw a ball, and when you are asked to supply a probability that the next ball drawn will be white, they will all be involved in the calculation. These are sometimes called *second order probabilities*, but in this example they are really just a probability distribution over possible ratios. If you discretized the possible ratios then it would be easy to set up a belief network for this, with the ratio being one of its nodes. That approach works fine for this simple problem, but you can imagine that if you had many interrelated variables, that it would become too complicated.

Instead Netica uses the concept of *experience*, which is a measure of the number of cases relating to some situation that Netica has seen.

At each node Netica stores one experience number for each possible configuration of states of the parent nodes, and with it a vector of probabilities (one probability for each state of the node). The experience value corresponds closely to the number of cases that have been seen (normally it is 1 more than the number of cases) or its equivalent. This form of experience has sometimes been called the “equivalent sample size” or “ess”. To save space, Netica doesn’t store

experience numbers for nodes that haven't been involved in any learning and haven't had a manual entry of experience.

Before learning begins (providing there has been no previous learning or entry of probabilities by an expert) the network starts off in a state of ignorance. All probabilities start as uniform, and experience starts off as 1.

For each case to be learned the following is done. Only nodes for which the case supplies a value (finding), and supplies values for all of its parents, have their experience and conditional probabilities modified (i.e., no missing data for that node). Each of these nodes are modified as follows. Only the single experience number, and the single probability vector, for the parent configuration which is consistent with the case is modified. The new experience number (exper') is found from the old (exper) by:

$$\text{exper}' = \text{exper} + \text{degree}$$

where degree is the multiplicity of the case (set by you just before learning begins). It is normally 1, but is included so that you can make it 2 to learn two identical cases at once, or -1 to "unlearn" a case, etc.

Within the probability vector, the probability for the node state that is consistent with the case is changed from prob_c to prob_c' as follows:

$$\text{prob}_c' = (\text{prob}_c * \text{exper} + \text{degree}) / \text{exper}'$$

The other probabilities in that vector are changed by:

$$\text{prob}_i' = (\text{prob}_i * \text{exper}) / \text{exper}'$$

which will keep the vector normalized (exper and exper' act as the old and new normalization constants).

10.3 Bayesian Learning

If you are not interested in a technical discussion of how Netica's learning is a form of Bayesian learning, you should skip this section.

Generally speaking, it is a wise idea to relate any proposed machine learning method to a Bayesian method to better understand its assumptions, strengths and weaknesses. If it can be cast, at least approximately, into a form of Bayesian learning, then you can check to see if the prior probabilities are suitable for the problem. If it does not even roughly correspond to any

form of Bayesian learning then its results are probably not very accurate, and it should not be used unless it has other valuable qualities, such as being particularly simple or fast.

Using the equations of the previous section is equivalent to a system of true Bayesian learning, under the assumptions that the conditional probabilities being learned are independent of each other, and the prior distributions are Dirichlet functions (if a node has 2 states, these are “beta functions”). For more information see Spiegelhalter&DLC93, section 4.1 (with the word “precision” equivalent to our “experience”).

Assuming the prior distributions to be Dirichlet generally does not result in a significant loss of accuracy, since precise complex priors aren’t usually available, and Dirichlet functions can fairly flexibly fit a wide variety of simple functions. Assuming the conditional probabilities to be independent generally results in poor performance when the number of usable cases isn’t large compared to the number of parent configurations, as mentioned at the beginning of this chapter.

10.4 Fading

When a belief network is supposed to capture relationships between variables in a world which is constantly changing, it is useful to treat more recent cases with a higher weight than older ones. An example might be an adaptive belief network that is constantly receiving new cases and doing inferences while it slowly changes to match a changing world.

Netica achieves this partial forgetting of the past by using *fading*. Every so often you select the nodes to be faded, do a **Relation → Fade** menu command, enter a degree from 0 to 1, and it will reduce the experience and smooth the probabilities of the node by an amount dictated by the degree, with 0 having no effect and 1 creating uniform distributions with no experience. Fading once with *degree* = 1-d, and again with *degree* = 1-f, is equivalent to a single fading with *degree* = 1-df.

Each of the probabilities in the node’s conditional probability table is modified as follows (where prob and exper are the old values of probability and experience, and prob’ and exper’ are the new values):

$$\text{prob}' = \text{normalize}(\text{prob} * \text{exper} * (1 - \text{degree}) + \text{degree})$$

exper’ is obtained as the normalization factor from above (remember that there is one experience number per vector of probabilities). So:

$$\text{prob}' * \text{exper}' = \text{prob} * \text{exper} * (1 - \text{degree}) + \text{degree}$$

When learning in a changing environment, you would normally fade every once in a while so that what has been recently learned is more strongly weighted than what was learned long ago. If an occurrence time for each case is known, and the cases are learned sequentially through time, then the amount of fading to be done is: $\text{degree} = 1 - r^{\Delta t}$ where Δt is the amount of time since the last fading was done, and r is a positive number less than (but close to) 1 and depends on the units of time and how quickly the environment is changing. Different nodes may require different values of r .

11 Display Style and Printing

The **Style** menu allows you to change the way networks are displayed and printed, without changing what they mean or the way they behave. The size and font of the nodes and link labels can be adjusted, and individual nodes can be displayed in a manner which is best for that node. If an end-user who is not familiar with belief networks is going to be using the finished network, then it can be displayed in a suitable manner, perhaps by hiding links and some nodes, and displaying the rest of the nodes as bar graphs, meters, or data entry ovals, depending on the application.

Network diagrams can be printed with a printer, in color and magnified / reduced if desired. Presentation quality graphics can be created by copying and pasting from a network into other applications.

11.1 Node Name and Title Display

Every node has a name, but there are character and length restrictions on names (see section 4.3). To provide you with greater flexibility in labeling nodes, some or all of the nodes can also be given a title, which has no such restrictions. On the displayed network, each node can be labeled with its name, its title, or both together (separated by a colon or parenthesis).

To display all the nodes by name, choose **Name** from the **Style → Node** submenu, to display them by title choose **Title**, or to display them using both, choose **Name: Title or Title (Name)**. The choice that you make will apply to all the nodes of the network. If you are displaying nodes by title, and some nodes don't have a title, then their name will be used instead. The starting style of a new network is **Title**, but you might not notice this if you don't use titles, since then just the node names will appear.

11.2 Font and Size

If you wish to determine or change the font used for the writing within nodes, select some nodes and then click **Style→Font**. A dialog box will appear showing the current font and its size, and will allow you to change them. The choice that you make will apply to all the nodes of the network. Whenever you choose a new font, or a new font size, all the nodes will be resized so that the writing within them fits nicely. In fact, the way you enlarge or shrink the nodes of a network is to choose a larger or smaller font size.

If a link is selected when you click **Style→Font**, then the new font and size will not apply to the nodes of the network (so no resizing will be done), but rather to the labels of all disconnected links.

11.3 Node Styles

There are several different forms in which a node can be displayed, and these are known as *node styles*. There is a default style for the whole network, but each node can be set to its own style individually, which overrides the default. To set the default style for the network, make sure no nodes are selected when you choose that style from the **Style→Node** menu. To have some nodes override that style, select them and then choose the desired style from the menu. If you have given some nodes an overriding style, but now you want them back to the default style, select them and then choose **Style→Node→Default**.

If you use the menu to change the default node style of the whole network, but none of the nodes change, it is because they all have overrides to the default. To remove these overrides, you could select them all and then choose **Style→Node→Default**.

The various node styles are: Labeled-Box, Hidden, Circle, Belief-Bar and Belief-Meter. Labeled-Box style draws a box whose shape and color are determined by the node kind, with the name / title of the node written within it. Hidden style draws nothing, and Circle style draws the node as a small circle. Belief-Bar and Belief-Meter styles are described in detail below.

11.3.1 Belief-Bar

To display the belief levels of a multistate nature node, the belief-bar style is most useful. It is illustrated in figure 11.1. Nodes can be displayed in this style by selecting them, and then choosing **Style→Node→Belief Bars**.

Each node will be displayed as a box labeled at the top with the node's name, title or both depending on what was chosen using the **Style → Node** submenu. The name of each state is shown in a section to the left, along with a number expressing the belief (probability) of that state as a percentage. If the name is too long to fit, it will end with an ellipsis (...). If the percentage is too small to display numerically, but nonzero, then it will be displayed as "0+". In a section to the right are bar graphs depicting the belief levels. Vertical dotted lines mark the 0.25, 0.5 and 0.75 levels. If the belief in one state is certainty, due to a finding for this node, then its bar will be bordered above and below by black lines. If the current belief percentages and positions of the bars are invalid for the findings entered (probably because the latest findings haven't been taken into account yet), the bars will be drawn in a fuzzy dotted manner.

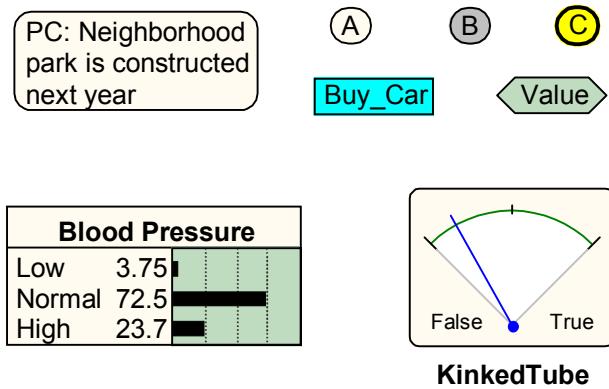


Figure 11.1 - Example of nodes displayed in different styles. Upper left is Labeled-box Name:Title style, lower left is Belief-Bar style, lower right is Belief-Meter style, and upper right are a few different kinds of nodes in Labeled-box Name style.

11.3.2 Belief-Meter

Belief-meters are often the display of choice for binary variables like true/false propositions or okay/faulty conditions. An example of one is illustrated in figure 10.1. Below the meter is the node's name, title or both depending on what was chosen using the **Style → Node** submenu. The two states are printed at the bottom of the meter, on the left and right. The needle indicates the relative belief in each state. For the rightmost state, the left end of the scale corresponds to a belief probability of 0% and the right end to 100%. The 50% point is when the needle is vertical, so the center of the scale is marked with a tick. If the belief is certainty due to a finding for this node, then the needle will be all the way to one side and the background gray of the meter will be darkened. If the current position of the needle is invalid for the findings entered (probably because the latest findings haven't been taken into account yet), the needle will be drawn as a dotted line.

If the belief-meter style is used for a node with more than two states, then all the states except the first will be grouped together and will be called “Other”. The first state will be rightmost on the meter, so its belief will be well represented by the needle (0% full left and 100% full right).

11.4 Net Styles

The **Style → Net** submenu allows you to choose the way network links are displayed. **Style → Net → Hide Links** allows you to hide all the links in the network. This is especially useful when you are creating something for an end-user who will make queries to the network, but does not care about how nodes are linked together. In that case, you may also want to hide some of the nodes which are of no interest to the end-user. To avoid confusion, it is not possible to display some links and hide others. When the links are hidden, **Style → Net → Hide Links** will be checked marked, and choosing it from the menu again will redisplay the links.

The remaining 3 choices of the **Style → Net** submenu are for those who are familiar with the process of belief network compiling, and wish to see how it is being done. They allow you to switch between showing the link structure of the network which was originally constructed (**Style → Net → Regular Dag**), the link structure of the Markov network which can be derived from it (**Style → Net → Markov Net**), and the link structure of the triangulated graph derived from the Markov network (**Style → Net → Triangulated**). The triangulated graph is used internally to determine the cliques when compiling the network, and a number is displayed with each node to indicate its position in the “elimination order” of the compilation. The **Triangulated** and **Markov Net** styles are only for viewing the network, printing it, changing styles, and copying its graphics (with no nodes selected). If you try to do any other operation the style will automatically change back to **Regular Dag**. Before you select these from the menu, you may want to save your network, since they will lose information about link bends.

11.5 Copying and Pasting Graphics

After cutting or copying all or part of a network, you can paste the graphics into other Windows applications, such as Microsoft Word, PowerPoint, Works, etc. If some nodes are selected when the **Copy** command is done, then graphics for only those nodes will be placed in the clipboard. If no nodes are selected, then graphics for the whole network will be. The nodes will be drawn with the same design style and font as they are shown on the screen, so you may want to adjust this with the **Style** menu before copying to the clipboard.

To see what is currently in the clipboard, you may find it useful to use the “Clipboard Viewer” program which is supplied with MS Windows. On a normal installation it can be found at **Start** → **Programs** → **Accessories** → **Clipboard Viewer**.

After you paste the graphic into another program, you may want to adjust its size and shape so that the aspect ratio (i.e. height / width ratio) is similar to the belief network it came from, in order to achieve the best appearance. If the aspect ratio is very different from the original, some of the characters may overlap.

Netica graphics are pasted into other programs as “enhanced metafiles”. When pasting into Word 97 it is often more convenient to have the graphic as a Word 97 “picture”. You can achieve this by first pasting it into Word, then while it is still selected do a **Cut**, then do an **Edit** → **Paste Special** and choose “Picture” from the dialog (and perhaps remove the check from ‘float over text’). Sometimes Word does not do a perfect job of converting it, but just double-clicking on the graphic to bring up the picture editor, and then closing it again without making any changes, seems to fix any imperfections.

If you want to paste a network into a bitmap program, such as “Paint” (which comes with Windows 95), it is recommended that you use Alt+PrintScreen to make a file containing the network graphics (i.e. a “screen dump”), and read that into the bitmap program.

11.6 Printing

You can print a network on a printer with the **File** → **Print** menu command. First you may want to do **File** → **Printer Setup** to select printing options, such as page size, margins, magnification / reduction, and which printer to use.

If the network diagram is larger than one page, then the printer will print parts of it on multiple pages in such a way that the pages can be trimmed and then attached together to produce the overall diagram. If you wish to see what part of the diagram will end up on which page you can turn on the page breaks, by choosing the menu entry **Layout** → **Show Page Breaks** if it is not already checked. This will draw a light blue line at the page divisions. Choosing it again will turn off the page break display. When you change the magnification / reduction (or margins, etc.) using **File** → **Printer Setup**, the page break lines will shift showing how much fits on each page.

12 Equations

The relation between a node and its parents can be entered using an equation if desired. This is possible whether the nodes are continuous or discrete, and whether the relation is probabilistic or deterministic. Before inference Netica turns the equation into a table, and uses the table in the same way as if you had entered it by hand as described in previous chapters.

For documentation on using the equation feature, contact Norsys Software Corp. by email at norsys_info@norsys.com.

13 Bibliography

Each bibliography reference is also listed in the index, so you can find where in this document it is mentioned.

Boerlage, Brent (1994) Link Strength in Bayesian Networks, Tech. Report 94-17, Dept. Computer Science, Univ. of British Columbia, BC.

Charniak, Eugene (1991) "Bayesian networks without tears" in *AI Magazine* (Winter 1991), **12**(4), 50-63.

Dean, Thomas L. and Michael P. Wellman (1991) *Planning and Control*, Morgan Kaufmann, San Mateo, CA.

Heckerman, David (1993) "Causal independence for knowledge acquisition and inference" in *Uncertainty in Artificial Intelligence: Proc. of the Ninth Conf.* (July, Washington, DC), David Heckerman and Abe Mamdani (eds.), Morgan Kaufmann, San Mateo.

Heckerman, David and Jack Breese (1994) "A new look at causal independence" in *Uncertainty in Artificial Intelligence: Proc. of the Tenth Conf.* (July, Seattle, WA), Ramon Lopez de Mantaras and David Poole (eds.), Morgan Kaufmann, San Mateo, CA.

Heckerman, David, Abe Mamdani and Michael P. Wellman (1995) "Real-world applications of Bayesian networks" in *Communications of the ACM*, **38**(3), 24-26. Introduction to this special issue of CACM on belief networks.

Henrion, Max, John S. Breese and Eric J. Horvitz (1991) "Decision Analysis and Expert Systems" in *AI Magazine* (Winter 1991), **12**(4), 64-91.

Lauritzen, Steffen L. and David J. Spiegelhalter (1988) "Local computations with probabilities on graphical structures and their application to expert systems" in *J. Royal Statistics Society B*, **50**(2), 157-194.

Matheson, James E. (1990) "Using Influence diagrams to value information and control" in *Influence Diagrams, Belief Nets and Decision Analysis*, Robert M. Oliver and J. Q. Smith (eds.), John Wiley & Sons, Chichester.

Neapolitan, Richard E. (1990) *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & Sons, New York.

Pearl, Judea (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA. 2nd edition 1991.

Qi, Runping (1994) *Decision graphs: Algorithms and applications to influence diagram evaluation and high-level path planning under uncertainty*, PhD Thesis, Dept. of Computer Science, Univ. of British Columbia, Canada. Also Tech. Report 94-27, Dept. of Computer Science, Univ. of British Columbia.

Qi, Runping and David Poole (1995) "New method for influence diagram evaluation" in *Computational Intelligence*, **11**(3).

Russell, Stuart and Peter Norvig (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ.

Shachter, Ross D. (1986) "Evaluating influence diagrams" in *Operations Research*, **34**(6), 871-882.

Shachter, Ross D. (1988) "Probabilistic inference and influence diagrams" in *Operations Research*, **36**(4), 589-604.

Shachter, Ross D. (1989) "Evidence absorption and propagation through evidence reversals" in *Proc. of the Fifth Workshop on Uncertainty in Artificial Intelligence* (Windsor, Ont.), 303-308. Later republished in: Henrion, Max (ed.) (1991) *Uncertainty in Artificial Intelligence 5*, North-Holland, Amsterdam.

Smith, James E., Samuel Holtzman and James E. Matheson (1993) "Structuring conditional relationships in influence diagrams" in *Operations Research*, **41**(2), 280-297.

Spiegelhalter, David J., A. Philip Dawid, Steffen L. Lauritzen and Robert G. Cowell (1993) "Bayesian analysis in expert systems" in *Statistical Science*, **8**(3), 219-283.

Zhang, Lianwen (Nevin) (1993) *A computational theory of decision networks*, PhD thesis, Dept. of Computer Science, Univ. of British Columbia, BC, Canada. Also released as Tech. Report 94-8, Dept. of Computer Science, Univ. of British Columbia, BC, Canada.

Zhang, Lianwen (Nevin), Runping Qi and David Poole (1994) "A computational theory of decision networks" in *International Journal of Approximate Reasoning*, **11**(2), 83-158.

Zhang, Lianwen (Nevin) and David Poole (1996) "Exploiting causal independence in Bayesian network inference" in *Journal of Artificial Intelligence Research*, **5**, 301-328.

14 Index

*, 67
absorb node tool, 62
absorbing a node, 62
active window, 21
add link tool, 28
Add No-Forgetting Links menu command, 53
added links during reversal, 60
adding links, 28
adding nodes, 27
adding states to node, 38
address of Norsys, 2
agents, 17
AI, 17
algorithms used by Netica
 decision problems, 64
 node absorption, 64
 probabilistic inference, 16, 17
Alt+PrintScreen, 81
application areas, 6
Apply button
 in node dialog box, 35
 in relation dialog box, 45
arrow keys, 30, 31
 in relation dialog box, 45
artificial intelligence, 17
Asia medical example, 20, 69
 diagram, 21
asia.cases, 68, 69
aspect ratio, 81
asterisk, 67
attribute-value, 67
autogrid, 30, 31
auto-updating, 23, 24
batch mode learning, 72
Bayesian network, 18
beep sound, 9
belief, 19
belief functions, 72
belief network, 18
 adaptive, 75
 compiling, 21
 file format, 31
 learning, 70
 new, 27
belief network libraries, 56
belief updating, 19
manual, 23
slow, 23
belief-bars
 automatically changing to, 21
 fuzzy, 23
 picture, 21
Belief-Bars node style, 78
belief-meter
 fuzzy, 23
Belief-Meter node style, 79
bends in links, 30
beta functions, 75
bibliography, 83
bitmap program, 81
blank cells in relation dialog box, 47
Boerlage94, 61, 83
border of node, 36
Car Buyer example, 54
case, 65
 identification number, 68
 transferring networks, 66
case file, 65
 comments, 67
 creating, 66
 creating by sampling, 68
 example, 68
 format, 67
 learning from, 72
CASE-1 file format, 67
causal network, 18
cell of relation dialog box, 44
chance node, 37
chance node tool, 27, 37
characters, overlapping, 81
Charniak91, 16, 83
checking probabilistic relation, 49
child node, 18
Circle node style, 78
classification, 70
Clipboard Viewer, 81
clique tree, 19
Close button
 in node dialog box, 36
 in relation dialog box, 46
closing relation dialog box, 46
compile network tool button, 21

compiling belief network, 21
computation costs, 53
conditional plan, 51
configurations of parents, 44
consistent findings, 24
constructing
 decision networks, 52
 network libraries, 58
 networks, 27

Cont./Discrete selector of node dialog, 36

context node, 62

contingent plan, 51

continuous value findings, 66

continuous vs. discrete, 36

co-operating agents, 53

copying / pasting graphics, 80

copying nodes, 33

copyright notice, 2

current state of node dialog box, 38

customizing for end-user, 77, 80

cutting and pasting

 in relation dialog box, 48

cutting nodes, 33

database of cases, 66

date published, 2

Dean&W91, 83

decision function, 51

decision network, 51

 constructing, 52

 solving, 53

 solving by node absorption, 64

decision node, 37, 51

decision node tool, 27, 37, 52

default node style, 78

degree of fading, 76

degree of learning, 71

delay of link, 40

Delete button

 in node dialog box, 38

delete key, 32

deleting links, 32

deleting nodes, 32

deleting states of a node, 38

Dempster-Shafer, 72

description of Netica, 7

description text of node, 39

Determin./Chance selector of relation dialog box, 46

deterministic, 18

deterministic node, 37

deterministic vs. probabilistic node

 changing, 46

diagnosis, 70

Dirichlet distribution, 75

disclaimer, 2, 8

disconnecting links, 32, 57

discrete vs. continuous, 36

discretizing, 37, 38, 41, 66

display style, 77

DNET-1 file format, 31

double-press, 28

dragging, 11

Drawing Balls example, 72

drawing size, 34

duplicating nodes, 33

Edit-Copy menu command, 33, 48, 58, 80

Edit-Cut menu command, 33

Edit-Delete menu command, 32

Edit-Duplicate menu command, 33

Edit-Paste menu command, 33, 48, 58

Edit-Redo menu command, 29

Edit-Select All menu command, 29

Edit-Undo menu command, 29, 45

elimination order, 80

empty cells in relation dialog box, 47

end key, 34, 46

enhanced metafile, 81

enter key, 28, 35

entering findings, 22

equation of node, 40

equations, for node relations, 82

equivalent sample size, 73

ess, 73

evidence, 19, 22

evidence node, 62

examples

 Asia (medical), 20, 69

 diagram, 21

 Car Buyer, 54

 Drawing Balls, 72

 Flow Instrument, 56, 58

 Political Riding, 66

 reducing large net, 62

 relation dialog box, 44

 Umbrella, 52

 Weather, 56

Examples folder, 8

exiting Netica, 9

expected value

 finding maximized, 54

experience, 72, 73

explaining, 26

explaining away, 9

F12 key, 28

F9 - F11 keys, 27

fading, 75

 degree of, 76

familiarity assumed, 6

features of Netica, 7

file format

 belief network (DNET-1), 31

 case file (CASE-1), 67

File-Close menu command, 27, 36, 46

File-Exit menu command, 9

File-Get Case menu command, 65

File-New Network menu command, 27, 58

File-Open menu command, 27

File-Print menu command, 81

File-Printer Setup menu command, 34, 81

File-Save As menu command, 27, 31

File-Save Case As menu command, 65

File-Save menu command, 27, 31

filling in missing node probabilities, 49

findings, 19, 22

 consistency, 24

 independent, 24

 likelihood, 24

 negative, 23

 order of entry, 22

 positive, 23

reading from file, 65

removing, 22, 66
 retracting, 22
 saving to file, 65
 sets of, 65
 transferring networks, 66
 findings pop-up menu, 22
 fixed decision function, 55
 Flow Instrument example, 56, 58
 folder of examples, 8
 font, 78
 forgetting, 53, 75
 frequency of cases, 68
 function inversion, 60
 functional relation, 18
 fuzzy logic, 72
 generating random cases, 68
 generating random node relations, 49
 generating uniform node probabilities, 49
 getting started, 6
 grid spacing, 30
 group decision making, 55
 Heckerman&MW95, 16, 83
 Heckerman93, 83
 Henrion&BH91, 16, 83
 Hidden node style, 78
 hiding links, 80
 home key, 34, 46
 icon, large blue square, 8
 IDname, 36
 IDnum, 68
 ignorance, 72
 imperfect observation, 24
 impossible conditions, 47
 inconsistent findings, 24
 independent findings, 24
 influence diagram, 51
 informational link, 51
 input names of node, 40
 installing Netica, 8
 Instrument example, 56, 58
 introduction to BNs, references, 16
 irrelevant informational links, 54
 iterated policy making, 54
 join tree, 19
 junction tree, 19
 keys

- arrow, 30, 31, 45
- delete, 32
- end, 34, 46
- enter, 28, 35
- home, 34, 46
- page down, 34, 46
- page up, 34, 46
- shift, 29

 kind of node, 37
 Kind selector of node dialog box, 37
 knife tool bar button, 57
 Labeled-box node style, 78
 large network diagrams, 81
 Lauritzen&S88, 83
 Layout-AutoGrid menu command, 30, 31
 Layout-Drawing Size menu command, 34
 Layout-Grid Spacing menu command, 30, 31
 Layout-Show Page Breaks menu command, 81
 learning
 belief networks, 70
 degree, 71
 fading, 75
 forgetting, 75
 from case file, 72
 from cases, 70
 from single case, 71
 parameter, 70
 structure, 70
 unlearning, 71
 legal disclaimer, 2, 8
 libraries

- belief network, 56

 LicAgree.txt file, 8
 license agreement, 8
 license password, 8
 likelihood finding, 24
 link, 18

- adding, 28
- deleting, 32
- disconnecting, 32, 57
- hiding, 80
- informational, 51
- name, 32, 40
 - style, 78
- reconnecting, 33, 58
- removing bends, 30
- reshaping, 30

 link delay, 40
 link reversal, 60

- links added/removed, 60
 - order of, 61

 link reversing tool, 61
 living document, 55
 Load button

- in node dialog box, 35
- in relation dialog box, 45

 magnification / reduction during printing, 81
 MAP, 25
 margins, printing, 81
 Markov network, 80
 Matheson90, 83
 maximizing expected utility, 51
 maximum a-posteriori probability, 25
 medical domain, 16, 18, 20, 65
 menu commands, 20
 menu items, 8

- Edit
 - Copy, 33, 48, 58, 80
 - Cut, 33
 - Delete, 32
 - Duplicate, 33
 - Paste, 33, 48, 58
 - Redo, 29
 - Select All, 29
 - Undo, 29, 45
- File
 - Close, 27, 36, 46
 - Exit, 9
 - Get Case, 65
 - New Network, 27, 58
 - Open, 27
 - Print, 81
 - Printer Setup, 34, 81
 - Save, 27, 31

Save As, 27, 31
 Save Case As, 65
Layout
 AutoGrid, 30, 31
 Drawing Size, 34
 Grid Spacing, 30, 31
 Show Page Breaks, 81
Modify
 Add .. Node, 27
 Add Case File Nodes, 72
 Add Link, 28
 Disconnect Links, 32, 57
Network
 Absorb Nodes, 62
 Add No-Forgetting Links, 53
 Automatic Updating, 23
 Compile, 21
 Most Probable Expl, 25
 Optimize Decisions, 53
 Remove Findings, 22, 66
 Simulate Cases, 69
 Update, 23
Relation
 Check, 49
 Fade, 75
 Fill in Missing, 49
 Incorp Case File, 72
 Incorporate Case, 71
 Normalize, 49
 Randomize, 49
 Remove, 49
 Uniform Probabilities, 49
 View / Edit, 43
Style, 77
 Font, 78
Net, 80
 Hide Links, 80
 Markov Net, 80
 Regular DAG, 80
 Triangulated, 80
Node
 Belief Bars, 78
 Belief Meter, 79
 Circle, 78
 Default, 78
 Hidden, 78
 Labeled Box, 78
 Name, 77
 Name.Title, 77
 Title, 77
 Title (Name), 77
Size, 78
 Window, 27, 35, 43
Messages window, 9
 missing data, 67, 70, 74
 missing state, reading case, 66
 missing toolbar buttons, 8
 model iteration, 54
 model search, 61
 Modify-Add .. Node menu command, 27
 Modify-Add Case File Nodes menu command, 72
 Modify-Add Link menu command, 28
 Modify-Disconnect Links menu command, 32, 57
 most probable explanation, 25
 moving nodes, 30
MPE, 25
MSPaint, pasting to, 81
 multi-case files, 66
 multi-line title, 36
 multiple agents, 53
 multiple node dialog boxes, 50
 multiple relation dialog boxes, 50
 multiplicity of cases, 68
 multi-purpose box, 39
 multi-purpose selector, 39
 Name edit box of node dialog box, 36
 name of node, 36, 77
 name of node changing automatically, 34
 nature node, 37, 51
 Nature/Dec./.. selector of node dialog box, 37
 Neapolitan90, 17, 83
 negative finding, 23
 Netica API, 6, 59, 67
 Netica Application, 6
 Netica description, 7
 Netica Messages window, 9
 Netica Programmer's Library, 6, 59, 67
 NeticaZip.exe, 8
 network libraries, 56
 network reduction, 62
 network transforms, 60
 Network-Absorb Nodes menu command, 62
 Network-Add No-Forgetting Links menu command, 53
 Network-Automatic Updating menu command, 23
 Network-Compile menu command, 21
 Network-Most Probable Expl menu command, 25
 Network-Optimize Decisions menu command, 53
 Network-Remove Findings menu command, 22, 66
 Network-Simulate Cases menu command, 69
 Network-Update menu command, 23
New button
 in node dialog box, 38
node, 18
 adding, 27, 72
 adding states, 38, 72
 changing, 35
 copying, 33
 cutting, 33
 darkening, 22
 deleting, 32
 description text, 39
 discrete vs. continuous, 36
 duplicating, 33
 equation, 40
 font, 78
 input names, 40
 kind, 37
 link delay, 40
 moving, 30
 name, 34, 36, 66, 77
 pasting, 33
 ranges, 38, 41
 relationship with parents, 43
 removing states, 38
 size, 78
 state values, 39
 states, 38, 40
 title, 36
 title displayed, 77
 type, 36

when changed, 42
 node absorption, 62
 for probabilistic inference, 63
 order of, 62
 to maximize expected utility, 54
 node dialog box, 35
 Apply button, 35
 Close button, 36
 Cont./Discrete selector, 36
 Delete button, 38
 Kind selector, 37
 Load button, 35
 multiple, 50
 Name edit box, 36
 Nature/Dec./.. selector, 37
 New button, 38
 obtaining, 35
 Okay button, 35
 parent selector, 40
 Ranges edit box, 38
 State Value edit box, 39
 States edit box, 38
 Title edit box, 36
 node libraries, 56
 node styles, 78
 Belief-Bars, 78
 Belief-Meter, 79
 Circle, 78
 default, 78
 Hidden, 78
 Labeled Box, 78
 no-forgetting links, 53, 64
 nonmonotonic logic, 72
 normalizing node probabilities, 49
 Norsys address, 2, 6
 not enough memory, 53
 nudging links, 31
 nudging nodes, 30
 NumCases, 68
 Okay button
 in node dialog box, 35
 in relation dialog box, 45
 operations research, 17
 optimal decisions, 51
 Optimize Decisions menu command, 53
 optimized compiling, 21
 other state, 79
 other state, reading case, 66
 overlapping characters, 81
 page breaks, 81
 page down key, 34, 46
 page size, 81
 page up key, 34, 46
 Paint, pasting to, 81
 parameter learning, 70
 parent node, 18
 password, 8
 pasting into other applications, 80
 pasting nodes, 33
 Pearl88, 17, 84
 plan, 51
 policy, 51
 Political Riding example, 66
 positive finding, 23
 posterior probabilities, 19
 prediction, 70
 printing, 81
 multiple pages, 81
 prior probabilities, 19
 probabilistic causal network, 18
 probabilistic inference, 19, 20
 by node absorption, 63
 probabilistic vs. deterministic node
 changing, 46
 probabilities don't sum to 1, 25, 49
 probability, 72
 Qi&Poole95, 84
 Qi94, 84
 query node, 62, 70
 quick tour of Netica, 9
 quitting Netica, 9
 random cases, 68
 random node relations, 49
 Ranges edit box of node dialog box, 38
 ranges of node, 38, 41
 real number findings, 66
 reconnecting links, 33, 58
 redoing, 29
 references for BNs, 16, 83
 relation dialog box, 43
 Apply button, 45
 arrow keys, 45
 blank cells, 47
 cell of, 44
 changing cell values, 45
 Close button, 46
 closing, 46
 cutting and pasting, 48
 Determin./Chance selector, 46
 empty cells, 47
 example, 44
 Load button, 45
 multiple, 50
 navigating cells, 45
 obtaining, 43
 Okay button, 45
 page down key, 46
 page up key, 46
 Relation menu commands, 48
 resizing, 46
 scrolling, 46
 selecting cells, 48
 switching node, 45
 tab key, 45
 undoing, 45
 x cells, 47
 Relation-Check menu command, 49
 Relation-Fade menu command, 75
 Relation-Fill in Missing menu command, 49
 Relation-Incorp Case File menu command, 72
 Relation-Incorporate Case menu command, 71
 Relation-Normalize menu command, 49
 Relation-Randomize menu command, 49
 Relation-Remove menu command, 49
 relationship of node, 43
 Relation-Uniform Probabilities menu command, 49
 Relation-View/Edit menu command, 43
 Remove Findings tool button, 22
 removing link bends, 30
 removing Netica, 8

removing node relationship, 49
 removing states from node, 38
 requirements, system, 8
 reshaping a link, 30
 resizing
 relation dialog box, 46
 resizing nodes, 78
 retracting findings, 22
 retrieving a case, 65
 reversing a link, 60
 running Netica, 8
 Russell&N95, 17, 84
 sampling, 68
 saving a case, 65
 saving compiled network, 22
 saving network to file, 31
 scenario, 26
 screen dump, 81
 scrolling
 multi-purpose box, 39
 network window, 34
 node description, 39
 relation dialog box, 46
 second order probabilities, 73
 selecting cells of relation dialog box, 48
 selecting nodes and links, 29
 sequential decisions, 54
 Shachter's algorithms, 64
 Shachter86, 84
 Shachter88, 84
 Shachter89, 84
 shift key, 29
 relation dialog box, 48
 size of nodes, 78
 Smith&HM93, 84
 solving decision networks, 53
 space at node border, 36
 Spiegelhalter&DLC93, 16, 84
 spreadsheet program
 copying to/from, 48, 66
 state name, 38, 67
 state names
 ‘other’, 79
 State Value edit box of node dialog box, 39
 state values of node, 39
 States edit box of node dialog box, 38
 states of node, 38, 40
 statistics, 17
 structure learning, 70
 Style menu, 77
 style of display, 77
 Style-Font submenu, 78
 Style-Net submenu, 80
 Style-Net-Hide Links menu command, 80
 Style-Net-Markov Net menu command, 80
 Style-Net-Regular DAG menu command, 80
 Style-Net-Triangulated menu command, 80
 Style-Node- Title (Name) menu command, 77
 Style-Node-Belief Bars menu command, 78
 Style-Node-Belief Meter menu command, 79
 Style-Node-Circle menu command, 78
 Style-Node-Default menu command, 78
 Style-Node-Hidden menu command, 78
 Style-Node-Labeled Box menu command, 78
 Style-Node-Name menu command, 77
 Style-Node-Title menu command, 77
 Style-Size submenu, 78
 sum of probabilities $\neq 1$, 25, 49
 summing out a variable, 62
 switching node of relation dialog box, 45
 synthetic data, 68
 system requirements, 8
 tab key
 in relation dialog box, 45
 terminating Netica, 9
 text editor
 copying to/from, 48, 66
 time node was changed, 42
 Title edit box of node dialog box, 36
 title of node, 36, 77
 tool buttons
 absorb node, 62
 add link, 28
 chance node, 27, 37
 compile network, 21
 decision node, 27, 37, 52
 disconnecter, 32
 disconnecter knife, 57
 edit / view relation, 43
 fill missing probabilities, 49
 link reversing, 61
 normalize probabilities, 49
 randomize relation, 49
 remove findings, 22
 remove relation, 49
 shapes
 blue rectangle, 27, 37, 52
 die on relation tree, 49
 double arrow, 61
 down arrow, 28
 gray oval, 27, 37
 green hexagon, 27, 37, 52
 green inverted V's, 43
 knife, 57
 lightening bolt, 21
 number enters blue cell, 49
 numeral one, 49
 right arrow, 23
 scissors, 32
 star, 62
 X over green relation tree, 49
 Xed case symbol, 22
 update beliefs, 23
 utility node, 27, 37, 52
 toolbar buttons
 missing, 8
 trademark notices, 2
 transferring cases between networks, 66
 transferring nets between computers, 31
 transforming a network, 60
 triangulated network, 80
 type of node, 36
 Type selector of node dialog box, 36
 Umbrella example, 52
 uncertainty, 72
 Uncertainty in AI Conference, 17
 undoing, 29
 uniform node probabilities, 49

uninstalling Netica, 8
unique node names, 34
unlearning, 71
unspecified probabilities, 47
Untitled-x window title, 31
Update tool button, 23
utility node, 37, 51
utility node tool, 27, 37, 52
value node, 37, 51
variable, 18
version number, 6
virtual evidence, 24
weak links, 61
Weather example, 56

what-if analysis, 55
when changed node, 42
wild state, reading case, 66
Window menu, 27
Windows menu, 35, 43
Word 97, pasting into, 81
word processor
 copying to/from, 48, 66
wrong decisions, 54
x cells in relation dialog box, 47
Zhang&P96, 84
Zhang&QP94, 84
Zhang93, 84