

**RepGrid • WebGrid • RepGrids  
RepNet • RepDoc • RepScript**

# Rep Plus

**Conceptual Representation Software**

## RepNet Manual

**Managing, Editing and Scripting  
the Rep Plus Graphics System**

May 2021

Brian R Gaines and Mildred L G Shaw

<http://cpsc.ucalgary.ca/~gaines/repplus/>

# Contents

Contents	i
<b>1 RepNet—Interactive Graphics</b>	<b>1</b>
1.1 Net syntax of node and link types . . . . .	1
1.2 Examples of conceptual structures represented in nets . . . . .	3
<b>2 Creating, saving, opening and printing nets</b>	<b>8</b>
2.1 File menu options for creating, opening, saving and printing nets . . . . .	8
2.1.1 New Net option . . . . .	8
2.1.2 Open and Open Recent options . . . . .	8
2.1.3 Save . . . . .	9
2.1.4 Page Setup and the Print options . . . . .	9
2.2 Rep Plus Manager options for managing nets . . . . .	10
2.2.1 New button . . . . .	11
2.2.2 Open button . . . . .	11
2.2.3 Copy and Copy Styles buttons . . . . .	11
2.3 Default nets . . . . .	11
2.3.1 Default file management . . . . .	11
<b>3 Creating and editing nets</b>	<b>13</b>
3.1 Entering new nodes . . . . .	13
3.2 Wrapping the label text . . . . .	14
3.3 Editing a node . . . . .	14
3.4 Selecting, positioning and deleting nodes . . . . .	15
3.5 Entering and deleting links . . . . .	16
3.6 Editing the state of individual nodes . . . . .	18
3.7 Enclosure nodes . . . . .	19
3.8 <i>Undo</i> in the <i>Edit</i> menu . . . . .	21
<b>4 Copying and duplicating nets or subnets</b>	<b>22</b>
4.1 Copy and paste . . . . .	22
4.2 Drag and drop . . . . .	22
4.3 Dragging files to a net . . . . .	23
<b>5 Contextual menus</b>	<b>24</b>
5.1 Contextual menu when cursor is over a node . . . . .	24

5.1.1	Aligning nodes . . . . .	25
5.1.2	Nudging nodes . . . . .	25
5.1.3	Spreading nodes . . . . .	26
5.1.4	Separating nodes . . . . .	27
5.1.5	Sizing nodes . . . . .	27
5.1.6	Flipping nodes . . . . .	28
5.1.7	Sending nodes to a different layer or location within a layer . . . . .	28
5.1.8	Grouping nodes . . . . .	30
5.1.9	Wrapping nodes . . . . .	30
5.1.10	Showing and setting node locations . . . . .	31
5.2	Contextual menu when cursor is not over a node . . . . .	32
5.2.1	Export the net as an SVG, PNG or JPEG image . . . . .	32
5.2.2	Import a PNG or JPEG picture to the net as part of a node . . . . .	34
5.2.3	Saving the net as a master file . . . . .	35
5.2.4	Locking and unlocking the net . . . . .	35
5.2.5	Writing the net data . . . . .	35
<b>6</b>	<b>Creating/editing types and styles for nodes, links and states</b>	<b>36</b>
6.1	Net, node, link and state styles . . . . .	36
6.1.1	Using the colour swatches to select colours . . . . .	37
6.2	Node types and styles . . . . .	38
6.3	Link types and styles . . . . .	40
6.4	State types and styles . . . . .	43
<b>7</b>	<b>Graphic structures from other applications in RepNet</b>	<b>45</b>
7.1	Standard styles of RepGrid plots . . . . .	45
7.2	Histogram plot styles . . . . .	47
7.3	Scree plot styles . . . . .	48
7.4	Display and Focus plot styles . . . . .	48
7.5	Crossplot and PrinGrid plot styles . . . . .	49
7.6	Socionets plot styles, link weights and and interactive features . . . . .	49
7.7	Techniques for creating graphic structures in RepNet . . . . .	50
<b>8</b>	<b>Scripting RepNet</b>	<b>51</b>
8.1	Running a script and interacting with it . . . . .	51
8.1.1	Messages sent to the script of an interactive (locked) net . . . . .	51
8.2	Files: opening files from a net . . . . .	52
8.2.1	Files script . . . . .	52

8.3	PrinComp: creating a plot from an analysis of a grid . . . . .	53
8.3.1	PrinComp script . . . . .	53
8.4	PrinGrid Trajectories: annotating an analysis of a composite grid . . . . .	58
8.4.1	PrinGrid Trajectories script . . . . .	58
8.5	HistoGrids: interacting though a net with the RepGrids Histo tool . . . . .	59
8.5.1	HistoGrids script . . . . .	60
8.6	CNet: interacting with a logical inference engine programmed in scripts . . . . .	61
9	Bibliography	62

# 1 RepNet—Interactive Graphics

The RepNet tool in Rep Plus is a general purpose graphics creation and editing tool that can be used in its own right to develop graphic tools for representing conceptual structures such as *concept maps* and *semantic networks* and is also used by various Rep Plus tools to support their graphic analyses.

RepGrid and RepGrids graphic analyses are output as RepNet nets which allows them to be edited in RepNet and stored as editable graphics. RepNet may be also used for entering, editing and sharing graphic representations of conceptual networks. It allows *visual languages* to be specified and representations within those languages to be constructed, edited and programmed to support user interaction.

The theory and application of a wide variety of conceptual representation techniques is described in the literature on argument forms, entailment meshes, bond graphs, concept maps, petrinets, influence diagrams, causal maps, belief nets, semantic networks and so on (Toulmin, 1958; Pask, 1976; Blundell, 1982; Novak and Gowin, 1984; Reisig, 1985; Oliver and Smith, 1990; Sowa, 1991; Lehmann and Rodin, 1992; Novak, 1998; Mintzes et al., 1998, 2000; Bryson et al., 2004; Gaines, 2009, 2015). RepNet may be used to replicate and support the different visual language used in such conceptual representation schema.

RepNet is programmable through RepScript enabling interactive graphic tools to be developed. The scripts may also include major functionality such as the *CNet* inference engine that supports logical structures expressed in semantic networks, including those of *conceptual nets* that parallel *conceptual grids*.

Nets and grids are complementary constructivist techniques, and may be used together to help people to explore their ideas about a topic, make them more explicit and discuss them with others (Gaines and Shaw, 1994, 1995b,a; Gaines, 2003).

## 1.1 Net syntax of node and link types

The graphic structures created and managed in RepNet are *nets* comprised of *nodes* which may be joined by *links*. A node type and a link type each have an associated *style* that determines the appearance of nodes or links of that type.

As well as its *type*, each node has a textual *label* which names it and is generally displayed, a textual *note* which annotates it and is not generally displayed, and a *state* indicated by a visual marker. The label or note, or both, may be an empty string.

Links may be styled in various ways, usually as lines or directed arrows with possible additional markings. The collection of node and link types available provide a syntax in which can be represented the semantics of the kind of net that can be drawn.

Figure 1 shows an example of a net. the semantic network on the left has 11 nodes linked by 8 links having 2 different link types. The node at the top right contains a PNG screen dump of a RepGrid constructs pane. The node at the bottom right is the plot generated by displaying the same grid.

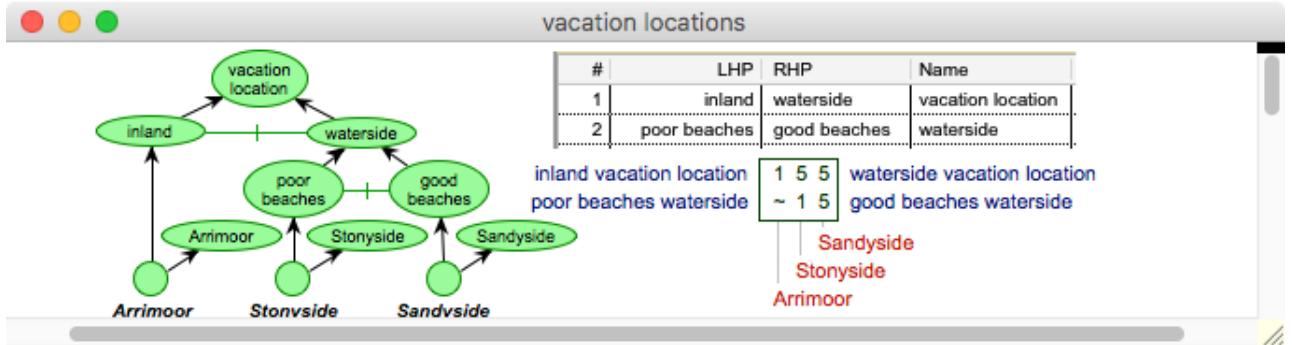


Figure 1: RepNet example

Thus a net can be comprised of a variety of graphic data structures. Double-clicking in a node opens an editing panel at the top of the window (Figure 2) It shows: the label of the node in a text editor at the top; the available node types in a row in the middle; and the available link types in two rows at the bottom. New nodes can be entered by clicking on a node type to select it, typing in a node label and keying *return*. New links can be created by clicking on a link type to select it and dragging between the two nodes to be linked. Nodes can be dragged to new positions and the links adjust accordingly.

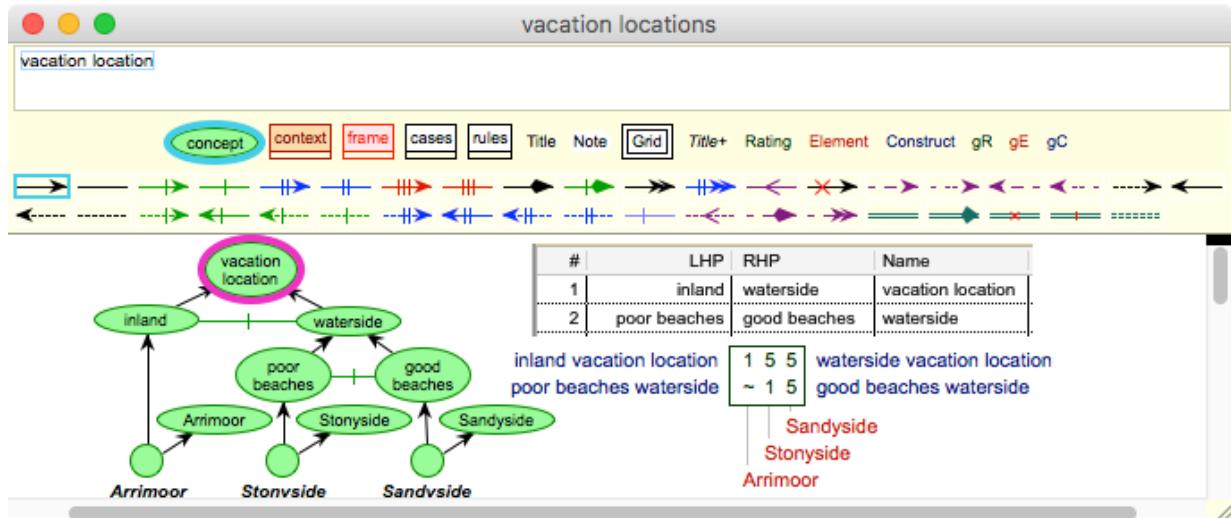


Figure 2: RepNet editor example

The node and link types are user-defined and can be edited and created through a popup editing palette (§6). The net structure is fully accessible to RepScript enabling nets to be created from programs and user interaction to be supported (§8). Details are provided in later sections, and various application examples are given in the following section.

## 1.2 Examples of conceptual structures represented in nets

There are many examples of RepNet graphics in the RepGrid and RepGrids manuals. The following are additional examples of more general graphic structures from the literature visually represented in RepNet. Figure 3 shows a Toulmin (1958) *argument diagram* in a net syntax having six types of nodes: *Grounds*, *Modality*, *Claim*, *Backing*, *Warrant* and *Rebuttal*, and one type of link, an arrow.

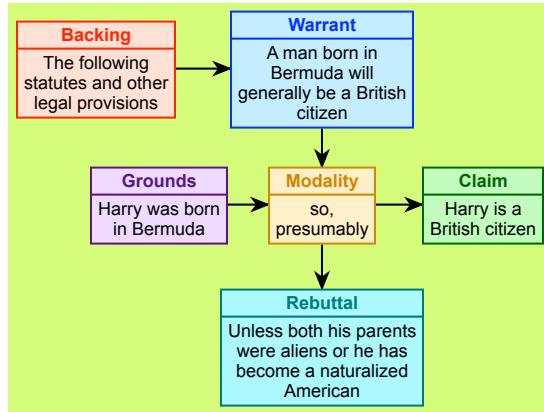


Figure 3: RepNet *argument diagram* example from Toulmin (1958)

Figure 4 shows a *concept map* (Novak and Gowin, 1984) in a net syntax having three types of nodes: *Ideas* styled as text with an oval surround, *Examples* of them styled as text with a rectangular surround and *Link labels* styled as text without a surround that labels the relation between nodes formed by the one link type, an arrow.

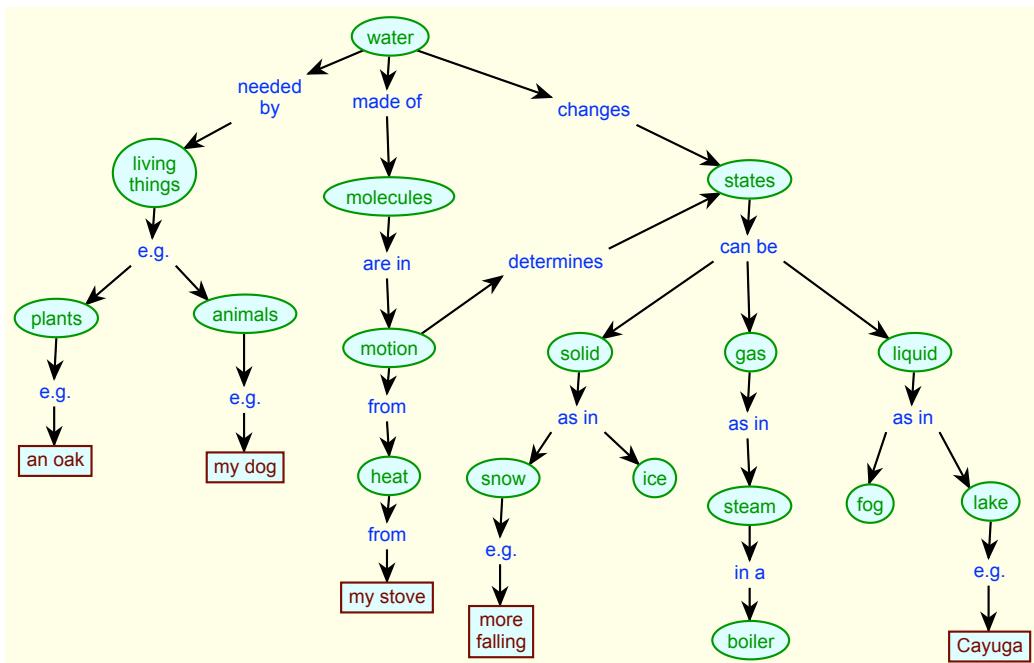


Figure 4: RepNet concept map example from Novak and Gowin (1984)

Figure 24 is a research activities net (Gaines and Shaw, 1995b; Shaw and Gaines, 1998).

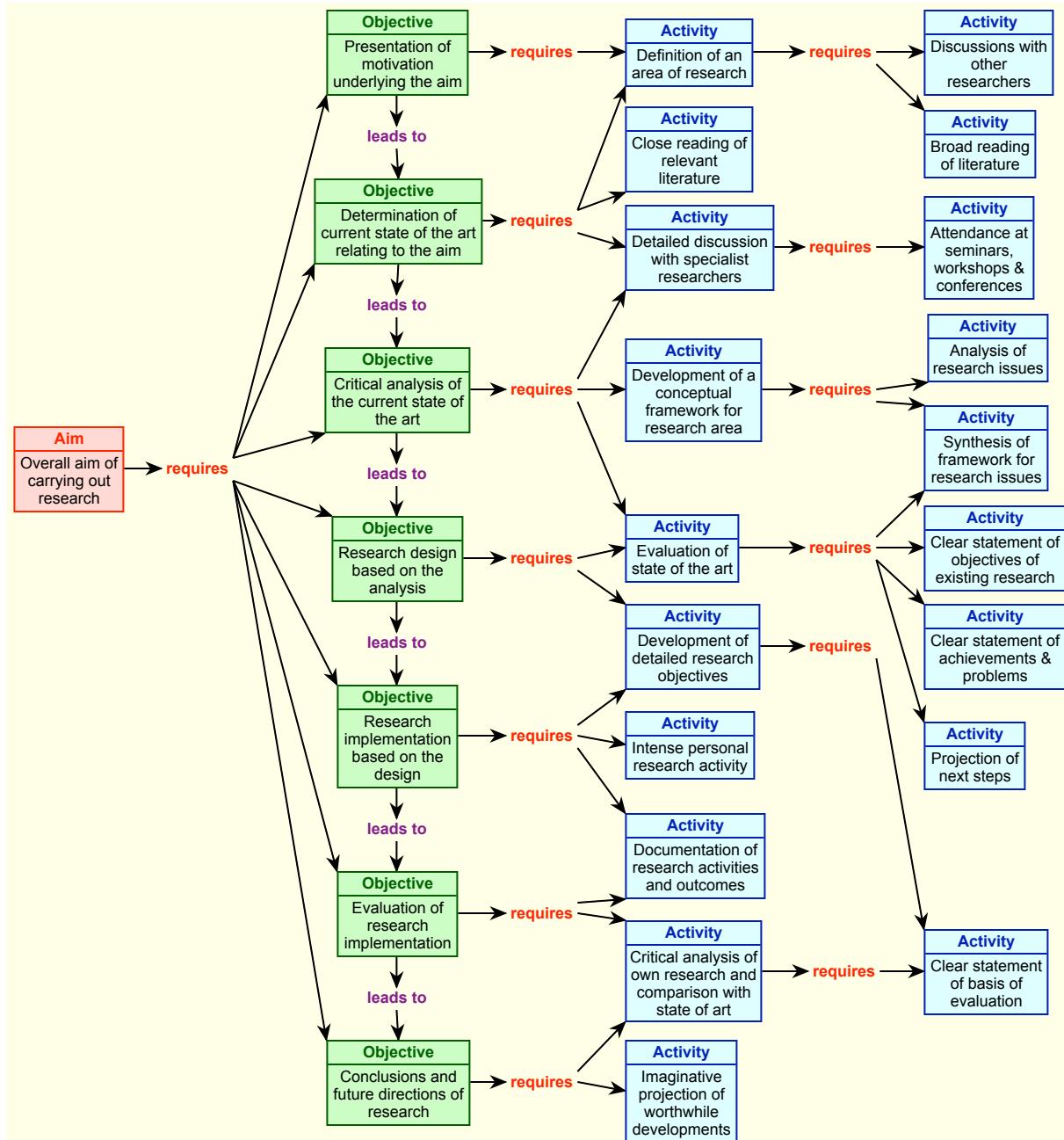


Figure 5: Research program and dissertation writing management concept map

It has five types of nodes: **Aim**, **Objective**, **Activity**, **leads to** and **requires**. The last two label the relations between nodes indicated by the one link type, an arrow. Graduate student use this conceptual framework to design and manage their research programs targeted on the production of a dissertation, editing the map to replace the abstract concepts with their instantiation in their own research activity.

Figure 6 shows a *semantic network* for an ontology of family relationships (Baader et al., 2003, p.52) in a net supporting a visual language for description logics (Gaines, 2009). The node types used are styled as ovals for concepts, rectangles for cardinality constraints and unenclosed text for relations. The link types used signify *defined by* or *defined by opposite* if there is a cross mark. The CNet script allows the net to be exported in symbolic form as description logic statements (Gaines, 2009). The cardinality constraint  $\exists$  is equivalent to  $\geq 1$ .

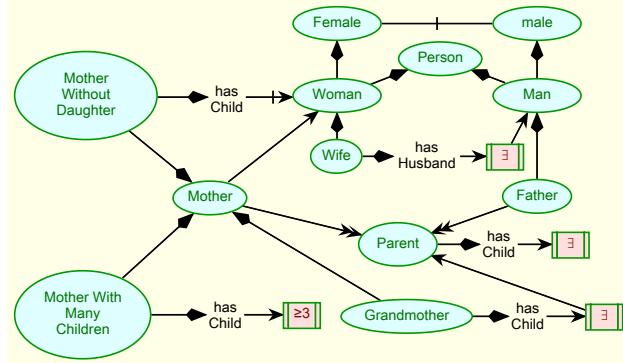


Figure 6: RepNet semantic network for a family ontology (Baader et al., 2003, p.52)

Figure 7 shows a *conceptual grid net* representing the conceptual dimensions of *art objects* (Gaines, 2009; Gaines and Shaw, 2012; Gaines, 2015) as defined by Gaut (2000; 2005). Just one node type is used to represent a templet that may be fitted to experience, and two links signifying either *ordination* or *opposition*.

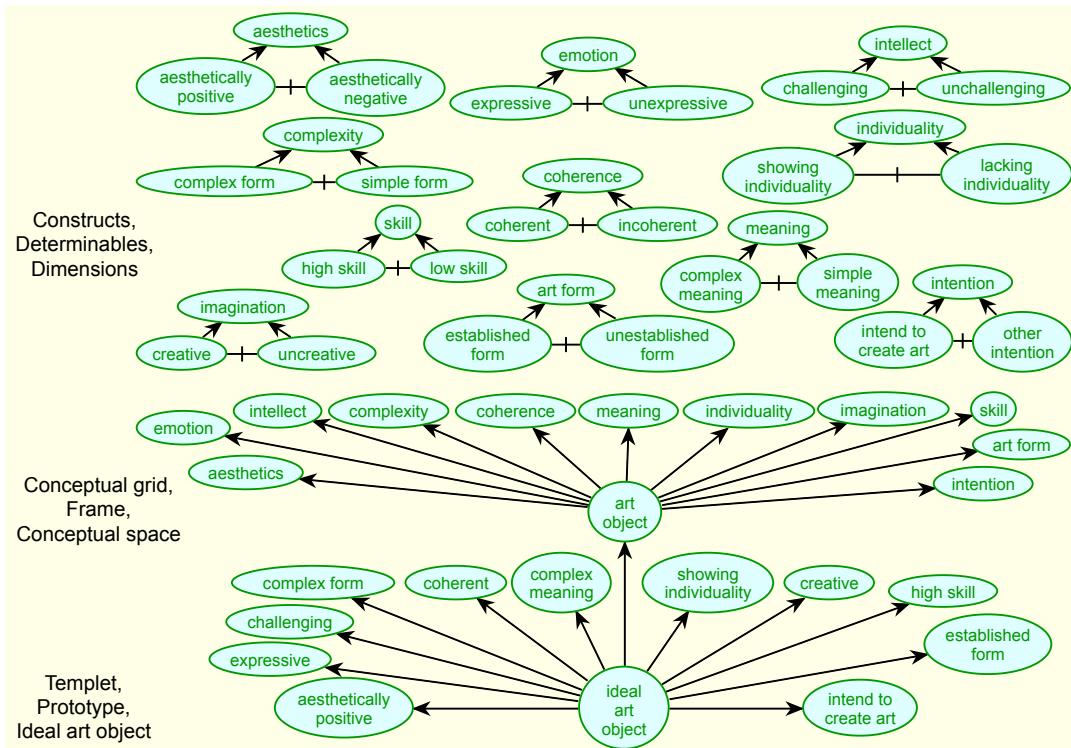


Figure 7: RepNet conceptual grid net providing a frame for construing *art objects* (Gaut, 2000)

The example illustrates how a conceptual grid may be represented in a net. At the top are *constructs* represented by two poles in opposition having a common range of convenience. These structures are readily extended to represent rating scales (Gaines, 2009; Gaines and Shaw, 2012). In the centre the *art object* node represents the frame provided by the constructs in the grid. At the bottom the *ideal art object* node represents an *element* in the grid with links to the poles representing its ratings.

In terms of the issues in the philosophy of art of providing an adequate *definition* of what it is to be an *art object*, the structure suggests that the *definition* is not so much to be found in the lower level prototypes but rather in the frame of which they are exemplars. That is, if one chooses to construct some entity as an *art object* then one is expected to be able to construe it in terms of a particular set of dimensions, such as the constructs suggested by Gaut (2000; 2005).

Figure 8 shows a net representing a conceptual grid that provides an expert system solution to a contact less prescription problem (Gaines, 2009; Gaines and Shaw, 2012) which has been used as a standard test case in the artificial intelligence literature (Cendrowska, 1987). An additional node type is used to indicate grid data representing cases used for testing anticipatory inference, and an additional link type is used to represent a preference between templets.

The top structures represent the constructs and frame as before. Below them are Kelly's (1955, p.121) anticipatory *intersects* (RepGrid's *classes*) as a hierarchy of templets with preferential links between them providing anticipatory schemata. At the bottom the anonymous node represents an element specifying the attributes of a particular case, or type of case. The rectangular node type on the bottom right provides a link to a grid representing a set of cases used to test the inferential schema.

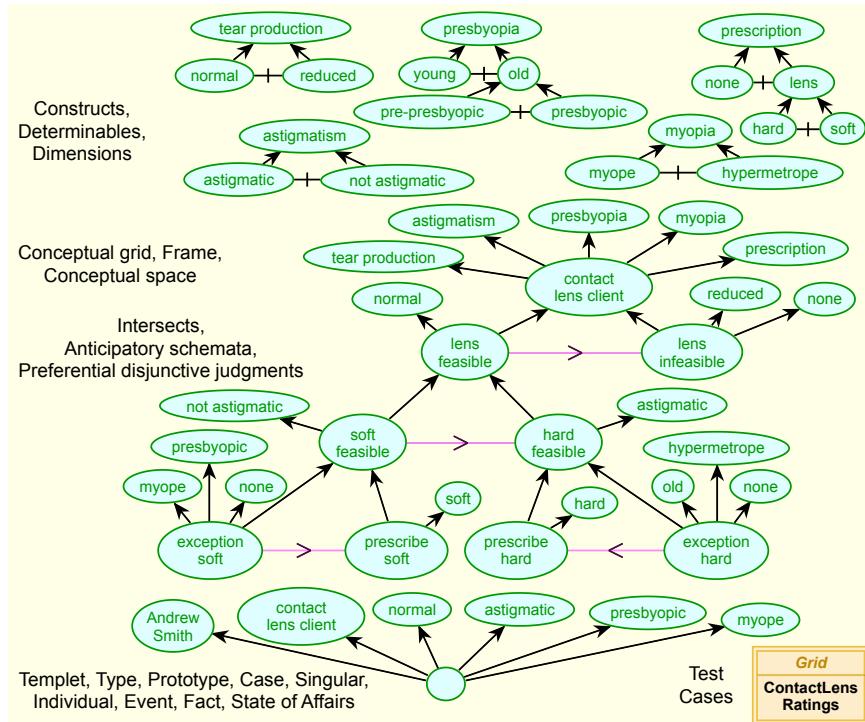


Figure 8: A net representing the inferential structures for solving an expert systems problem

CNet, a RepScript program that supports interaction with such nets and inference based on them, can be used to infer the appropriate prescription based on the way in which a client is construed in terms of the constructs. It can also check the inferences for sets of test cases held in grids. The anticipatory hypothesis is that one of the intersect templets must fit any specified case and corresponds to abductive *inference to the best explanation* (Lipton, 2004; Bird, 2007; Gaines and Shaw, 2012).

RepNet may also be used as a general-purpose graphics tool to prepare illustrations. Figure 9 shows a net illustrating the embedding of images as constituents of nodes in a net.

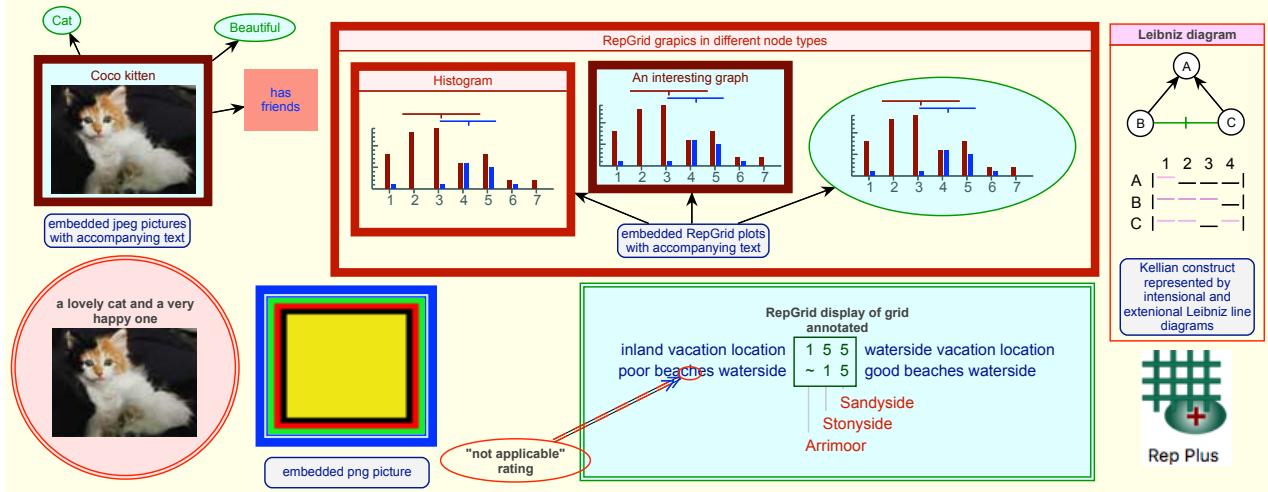


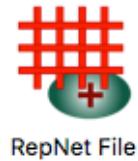
Figure 9: Embedding images in nodes

The node at the top left shows a JPEG image of a kitten embedded in a node with links to its attributes, and that below shows embedded in a node having a different style and a different textual label. The node at the top centre shows graphic output from the RepGrid *Synopsis* tool embedded in three different node types that are grouped in an enclosure node. The node at the to left shows a CNet analysis of the semantics of a net graphed by a script that generates intensional and extensional Leibniz diagrams and graph them in RepNet. The node at the bottom right of centre shows a RepGrid display of a small grid with graphic annotation entered in RepNet. The node at the bottom right shows the Rep Plus application icon embedded as a PNG image.

The following sections document the techniques involved in creating, editing, exporting and scripting RepNet graphics.

## 2 Creating, saving, opening and printing nets

A red grid icon shown on the right is used for RepNet files. They contain lists of the node, link and state types, and lists of the nodes and links that have been created. They can contain user-defined data specific to a particular study, and system-defined data relating to parameters chosen, window size and position, and so on. The data is held in a text format that is both human and machine readable.



### 2.1 File menu options for creating, opening, saving and printing nets

Figure 10 shows the File menu options relevant to creating, opening, saving and printing nets.

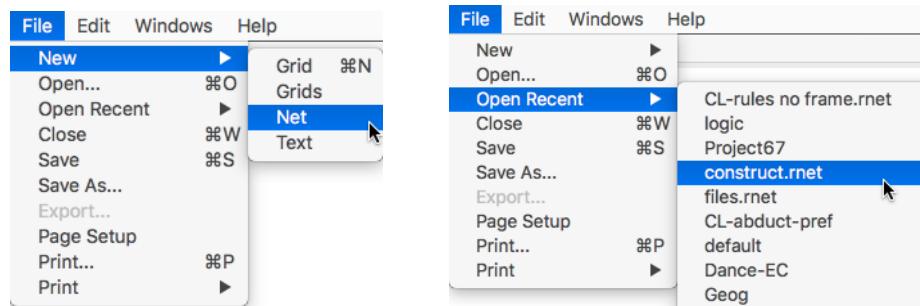


Figure 10: *File* menu options for RepNet

#### 2.1.1 New Net option

The *New Net* option creates a new net which is a copy of the currently specified *default net* (§2.3.1).

#### 2.1.2 Open and Open Recent options

The *Open* (CMD-O) and *Open Recent* options will open any Rep Plus file, including a net. *Open* brings up a dialog for opening a file, with net files indicated by their distinctive icons.

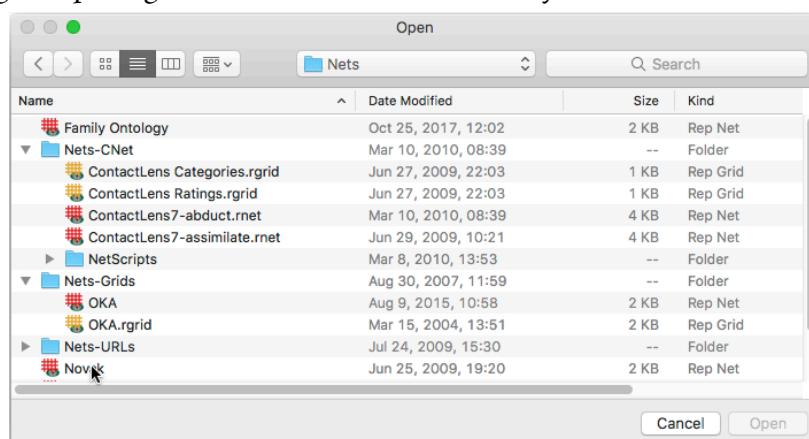


Figure 11: Selecting a net file to open

*OpenRecent* lists recently opened Rep Plus files, including nets, and one may be selected to be opened (Figure 10 right). Net files may also be opened by double-clicking on a net file, or dragging it to the Rep Plus application icon or Manager window. RepNet can decode net files from previous versions of RepNet but those it saves are not backwards compatible.

### 2.1.3 Save

and Save As options

The *Save* (CMD-S) and *Save As* options save any net showing in the top window, either to its existing file or to a specified new file.

Note that the *Save* option is always available for a net since, for example, the size and location of the net window are stored in file but changing them is not treated as an undoable action.

### 2.1.4 Page Setup and the Print options

A net in the front window may be printed by using the *Print* option, and the printing may be scaled through the *Page setup* option.

The page area for printing may be shown through by an alt/option-click in the net outside the nodes (Figure 12). If the net is larger than the size shown then it will printed across as many pages as are needed (allowing large nets to be printed at full, or large, size for posters created by glueing multiple pages together).

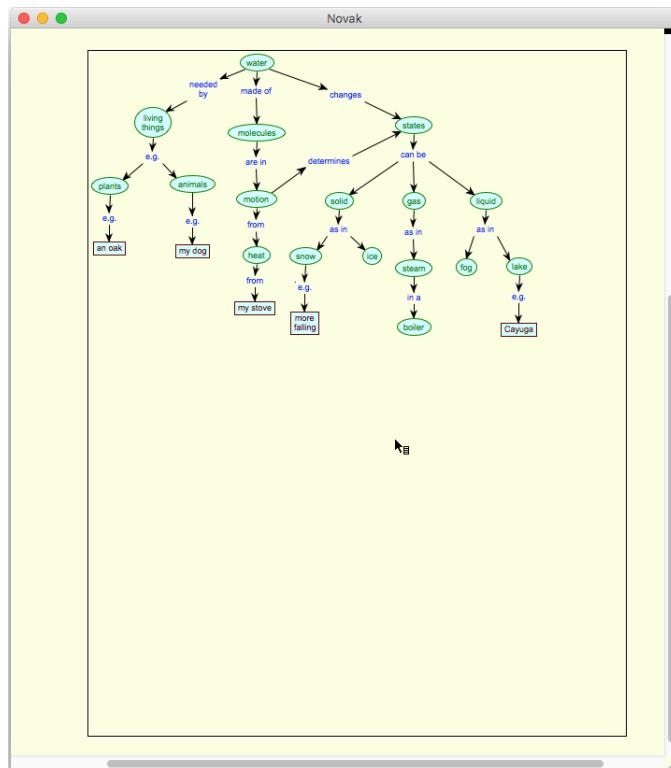


Figure 12: Showing the page area for printing by an alt/option-click in the net outside the nodes

A net may be scaled to fit a page, and/or only selected nodes printed, through an additional *Print* option in the *File* menu that has a submenu providing options to fit the net to one page, print only selected nodes (or the entire net if none are selected), or both (Figure 13).

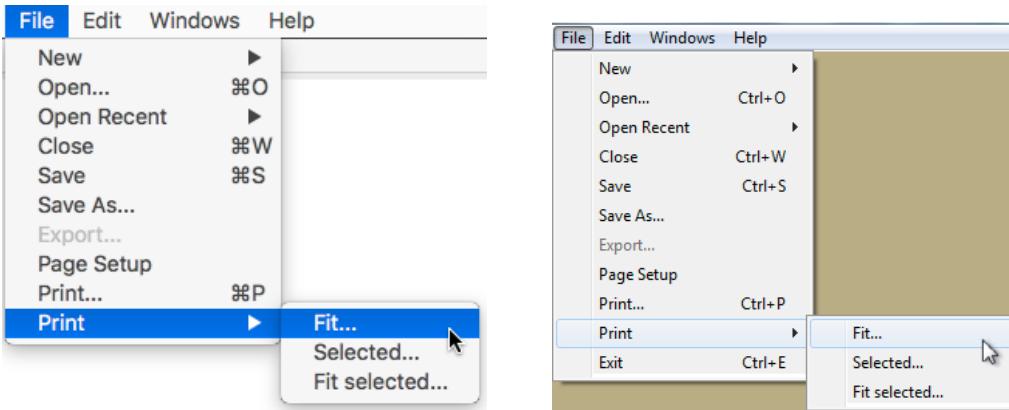


Figure 13: Menu options to fit the net to the printed page, print only selected nodes, or both

## 2.2 Rep Plus Manager options for managing nets

The *RepNet* panel at the centre bottom of the Rep Plus Manager window provides options for creating a new net, opening an existing one, creating a copy of an existing one (with all its data or only its styles), and for managing the available default nets (Figure 14).

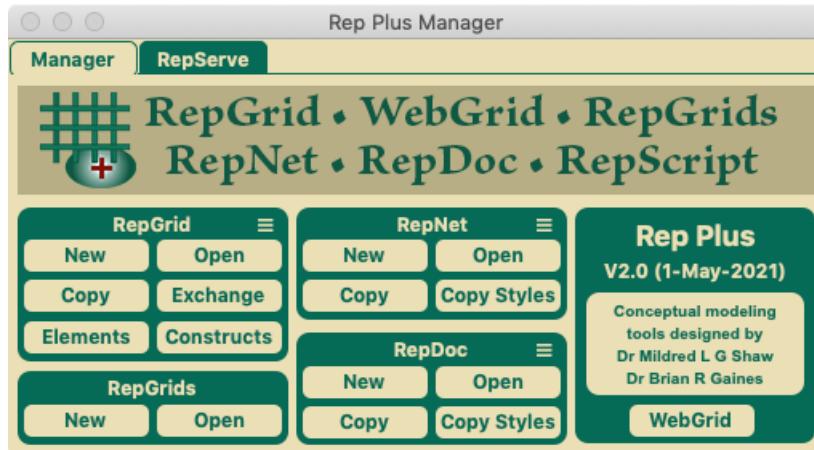


Figure 14: RepNet panel at the centre bottom of the Manager Window

A net file may be dragged to any of the buttons as an alternative to using the file selection dialogue (Figure 15). Dragging to *New* is equivalent to dragging to *Copy*.

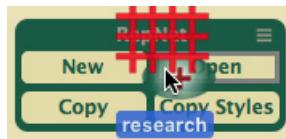


Figure 15: Dragging a net file to one of the buttons in the RepNet panel

The *New* and *Open* options duplicate the equivalent functionality in the *File* menu but will only open RepNet files. In addition, a popup menu at the top right corner may be used to manage the available default nets (§2.3.1).

### 2.2.1 New button

Clicking on the *New* button in the *RepNet* area of the *Rep Plus Manager* window creates a new net with the current default values (§2.3.1).

The net that is created is a copy of the current default net stored in the *NetScripts* directory in the *RepPlus* directory in the *Documents* or *MyDocuments* directory, or in the *NetScripts* directory in the same directory as the *Rep Plus* application where a default net is preinstalled.

### 2.2.2 Open button

Clicking on the *Open* button in the *RepNet* panel brings up the dialog for opening a file in the same way as the *File* menu *Open* option. The difference is that when a file is selected RepNet will attempt to open it as a net, reporting an error if the data cannot be interpreted as such.

### 2.2.3 Copy and Copy Styles buttons

Clicking on the *Copy* or *Copy Styles* button in the *RepNet* panel brings up the dialog for opening a file and creates a copy of the selected file, either the entire file or just the styles, respectively. The latter makes it possible to use any RepGrid file as a master for the styles that it defines.

## 2.3 Default nets

Users may store as many potential default nets as they wish in the *NetScripts* directory in the *Rep-Plus* directory in the *Documents* or *MyDocuments* directory. This enables a user to set up several potential default nets with different content, usually the node, link and state types, but also nets with default initial nodes and links.

### 2.3.1 Default file management

Users can select which of the available default nets is to be copied when a new net is created by clicking in the *Menu* symbol ( $\equiv$ ) in the *RepNet* panel of the *Manager* window (Figure 16).

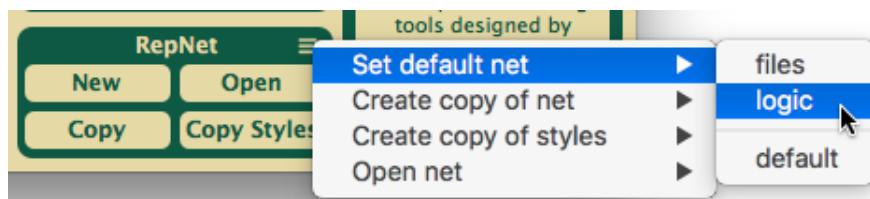


Figure 16: Setting the default net to be copied when creating a new net

The hierarchical popup menu shows the available nets in the on the right. The nets *files* and *logic* are user-developed to provide an appropriate node/link/state syntax for links to files or semantic networks respectively. The net *default* is system installed.

The main menu on the left provides three options:

**Set default net** Select which of the nets on the right is the default to be copied when a new net is created;

**Create copy of net** Create a copy of the selected net on the right—a useful alternative to changing the default net when only one new net of a different type is required;

**Create copy of styles** Create a copy of the selected net on the right containing only the styles—useful if some default nets contain nodes and links;

**Open net** Open the selected net for editing—making it simple to access the possible default net in order to make changes to them.

Users do not need to access these capabilities and can simply use the *New Net* command in the *File* menu or *Manager* window to create an empty net that is a copy of the system-installed default grid. However, entering node, link and state types for particular applications is time-consuming, and once this is done it is useful to store an empty net as a templet for future creation of nets of the same type.

This can also be done by copying an existing net file and deleting the net itself retaining only the types. Nets can also be saved as *master files* that when opened create a copy (§5.2.3).

### 3 Creating and editing nets

This section is concerned with creating and editing nets when a specific syntax of node and link types has already been specified. As already noted, RepNet keeps track of all the operations that change nets and supports multi-level undo through the *Undo* item at the top of the *Edit* menu (CMD-Z).

When the *New* button is clicked in the *Net* button in the *Manager* window a blank RepNet window opens as specified by the current default net.

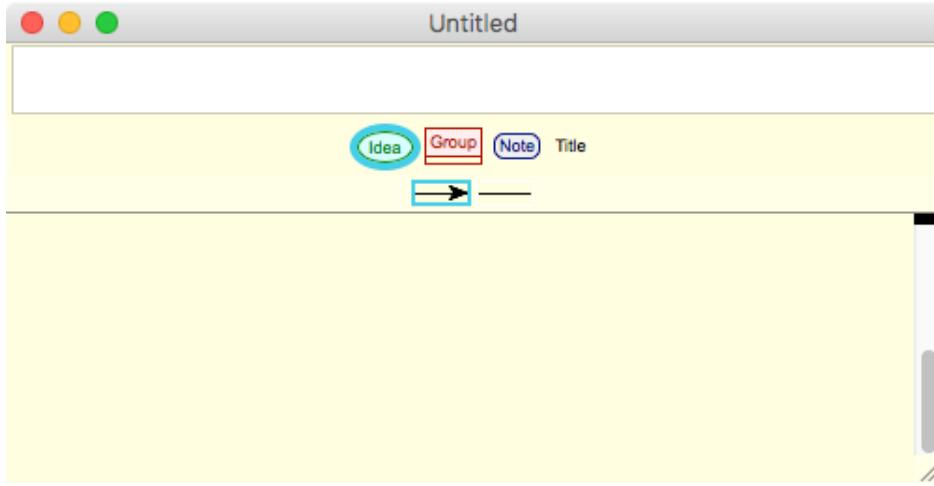


Figure 17: New net created

At the top is an editing area showing a text input field for the node label. Below this are icons for the node types initially available one of which may be selected by clicking on it. Below this are icons for the types of link available

When editing is complete the editing area may be covered by clicking in the dark rectangle at the top of the vertical scroll bar on the right of the net window and dragging it upwards. This drags the net up to cover the edit area, and it may be dragged down again at any time to expose the edit area. When a normal net file is opened from a file the edit area is covered, and when a new net is created the edit area is partially exposed as shown. The dark rectangle may be dragged further down to expose editing areas for node states and comments (§3.6).

#### 3.1 Entering new nodes

If one types a label in the text field at the top and keys *return*. then a new node is created of the type whose icon is selected. To support rapid entry, the text entered becomes selected so that further typing replaces it. Hence one can create a set of nodes of the same type by typing the label of each followed by *return*. They are automatically offset from one another (Figure 18).

If one double clicks an icon then it is selected and a new node is created just as if one had keyed *return*. Hence one can create a set of nodes of different types by typing the label of each followed by double-clicking the required type.

The location where the new node may be changed by clicking at the required location. The node will be centred on that location.

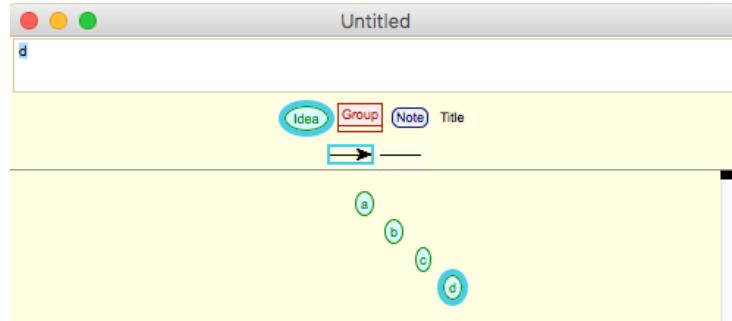


Figure 18: Entering a succession of nodes

Clicking in the node types area outside all the nodes deselects them all and any node created will have a null type. It will have a location and size which will be apparent when it is selected but will not display a label or surrounding structure. This node type is used primarily by other Rep Plus tools that generate graphic plots since a null node does display embedded graphics and can be useful if one wishes to display a bare image.

### 3.2 Wrapping the label text

The label text is wrapped automatically to be enclosed in a rectangle with its width about forty per cent more than its height, and the surrounding shape specified by the node type is fitted around this.

One can control the wrapping by inserting a SHIFT-return character where one wants to break the line. One can specify an unbroken line by putting a SHIFT-return character at the end of it.

### 3.3 Editing a node

One can edit any node by double-clicking on it which opens the top editing area (if it is not already open) and shows the node label and type. The node displays a mauve surround to show it is editable. As one mouses over it resizing arrows appear at each corner. Dragging these changes the size and aspect ratio, and the text wraps automatically unless it has return characters embedded in it (Figure 19). Any links between the node and others are automatically recomputed as the node resizes.

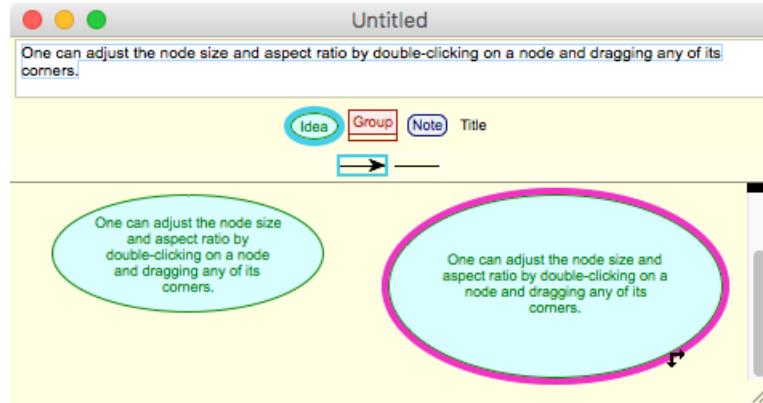


Figure 19: Resizing a node

The node can only be resized smaller if there remains room for the label text to be shown and dragging will become ineffective once the smallest possible size is reached. This make it easy to drag to the smallest possible size for text where embedded returns have been used to manage wrapping.

The label text and node type of an editable node may be changed by editing the text in the label field and/or clicking on a new node type. When one keys *return* the edited text replaces the existing node text and the node type is changed to that selected.

### 3.4 Selecting, positioning and deleting nodes

Nodes may be selected by clicking in them, and are then highlighted to show they have been selected. Multiple selections may be made by holding down the SHIFT key while clicking. All nodes may be selected through *Select All* in the *Edit* menu or keying CMD-A. A set of contiguous nodes may be selected by clicking outside them and dragging to draw a temporary thick black *selection rectangle* around them (Figure 20 left). When the mouse is released the enclosed nodes are selected as indicated by the thick blue surround drawn around them (Figure 20 right).

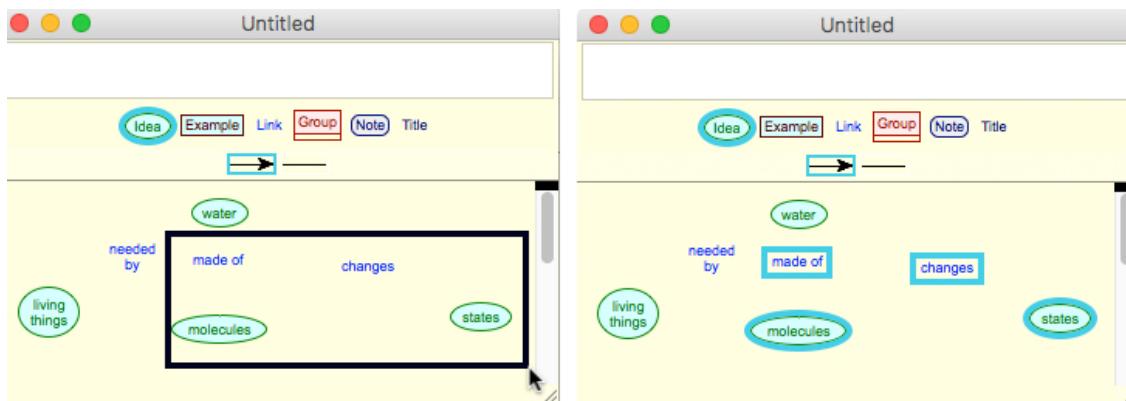


Figure 20: Selection rectangle dragged around nodes to select them

Clicking in an unselected node when others are selected deselects those other nodes. SHIFT-clicking in the unselected node adds it to the selection. SHIFT-clicking in a selected node removes it from the selection.

Selected nodes may be repositioned by using the four “arrow” keys on the keyboard. A normal click moves the nodes by 1 pixel in the direction of the arrow, and a SHIFT-click by 10 pixels. Selected nodes may also be repositioned by clicking in any selected node and dragging it to a new position.

Selected nodes may be deleted by keying *delete*.

### 3.5 Entering and deleting links

As the mouse moves across a node the cursor changes shape to indicate what action may be performed by clicking and dragging (Figure 21).

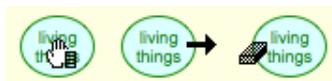


Figure 21: Hand, link and eraser cursors with the mouse at different positions over a node

In the centre it becomes a hand to indicate that the node can be grabbed and repositioned by dragging. At the right edge of a node it becomes an arrow to indicate that a link may be drawn from that node to another. At the left edge of a node it becomes an eraser to indicate that the same action will erase any link already drawn between the nodes.

Figure 22 shows some links being entered. The arrow link type has been selected. On the left the mouse has been moved to the right hand side of a node where the cursor changes to an arrow symbol and clicked and dragged with cursor remaining an arrow until, on the right, it arrives over another node and changes to a cross to indicate that a link may be dropped in place between the two nodes.

The link may be removed by the same sequence of actions but starting at the left hand edge of the initial node where the cursor becomes an eraser symbol. Whilst for directed links the path must be from the initial node to the final one, erasure may be carried out in either direction.

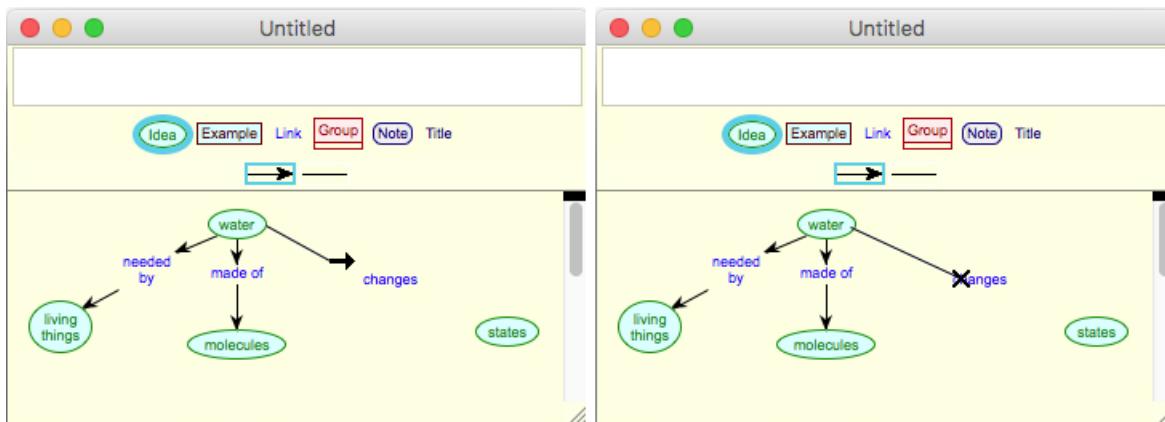


Figure 22: Link being entered between two nodes

Note that *changes* is essentially a link label for the link between the idea of *water* and that of *states*. RepNet supports the labeling of links between concepts by treating the link label as a node in its own right. This is the most general and flexible way of supporting link labels (and bipartite graphs), allowing links to have types if appropriate, and several links to share the same label if appropriate. For example, in Figure 4 *can be* and *as in* are used as shared labels, avoiding unnecessary duplication and simplifying and clarifying the meaning of the net.

RepNet only supports a single link from one node to another and if a link is entered when one already exists the old link is replaced. If a link is entered in the opposite direction then both links are retained but drawn displaced so that they may be distinguished. Figure 23 shows some two-way links.

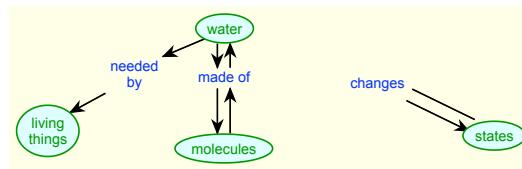


Figure 23: Two way links are automatically offset

The positioning of the points where links join nodes is automatic and nodes are dragged its links are recomputed automatically. When a node is deleted its links are automatically deleted.

Two options for link joins are available in the net style menu (§6): *join centres* where the links are computed as if they were joining the centres of the node but truncated at the edge of the node; *join 8 points* where the links are computed to join only at one of the corners, or the middle of a side, with the position chosen to minimize the length of the line and to ensure that the link is always in the direction of the relative positions of the node. It is simple to switch between the two modes and see which is better for a particular diagram.

For horizontal and vertical links there is no difference between the two options. For most nets, joining through the centres of the node is most effective, but for some nets the 8 points option results in fewer links crossing nodes. For example, Figure 24 is most tidily linked using 8 points, as illustrated in Figure 24.

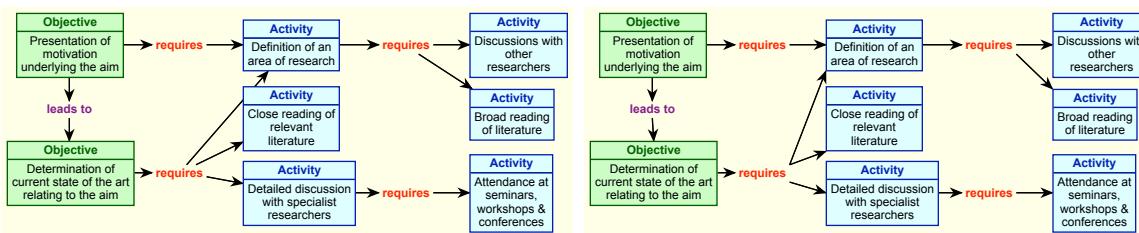


Figure 24: Left: joining to centres—Right: joining to 8 points

### 3.6 Editing the state of individual nodes

Each node also has an individual state indicated by a named style and eight binary flags that can be on or off as well as a hidden field for notes. These are normally set through scripts to indicate the outcome of some processing activity, but they may also be edited manually.

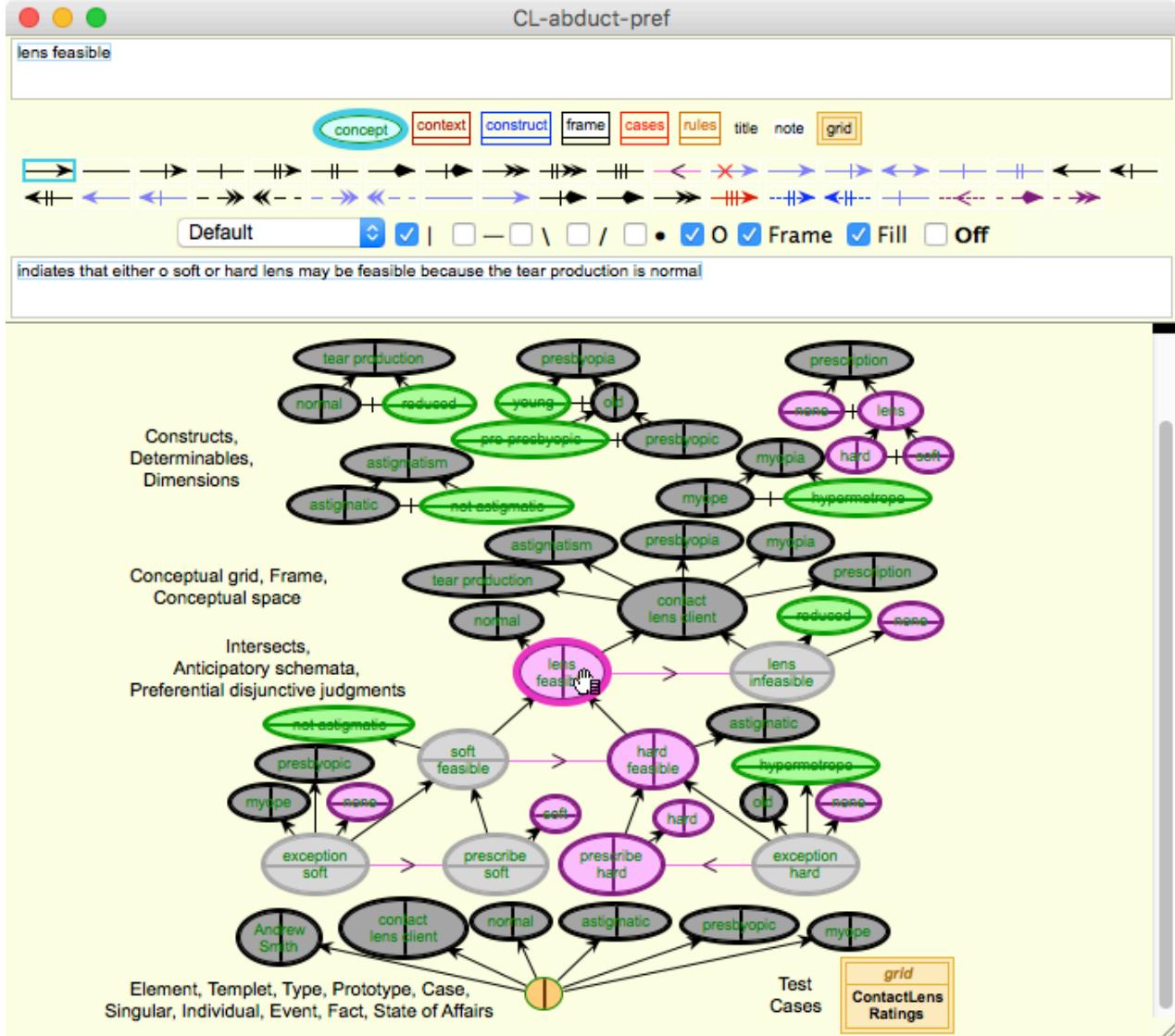


Figure 25: Showing and editing the states of nodes

For example, Figure 25 shows the results of running CNET, a logical inference script, on the expert systems net example of Figure 8. The inferential process has assigned a state to each node indicating its truth value and the type of inference that led to that value.

The black marker at the top of the scrollbar on the right has been dragged down to expose five editing panels: the label editor; the node types; the link types; the state types and values; and the note editor.

The three top panels have been discussed previously and provide richer examples of node and links types than previously shown. The state panel below the links panel provides a popup menu for the style, a check box for eight possible state marker, and a ninth check box that offers a convenient way to set all the flags to their off state.

The *lens feasible* node where the drag cursor is hovering in the centre has been double-clicked to open it for editing. Its state has been set to mark it with a vertical bar to indicate it is true for the case, and coloured mauve to indicate this was inferred through default reasoning.

The state panel shows that the state type has been set to *default* and the state itself has been set to be shown as a vertical bar, an annulus and frame colour specified by the style, and a fill colour also specified by the style.

Each state style specifies two colours, one for the *frame* and the other for the *fill*. The eight state marker check boxes specify how these colours are being used to decorate a specific node:

- | Horizontal line across node in frame colour;
- Vertical line across node in frame colour;
- \ Backward diagonal across node in frame colour;
- / Forward diagonal across node in frame colour;
- Bullet at centre of node in frame colour that resizes with node;
- Annulus around node in frame colour;

**Frame** Colour node frame in frame colour (replacing frame colour specified in node style);

**Fill** Fill node with fill colour (replacing fill colour specified in node style).

The state of the node opened for editing may be changed by selecting another node type and/or clicking in the check boxes, and keying “return.” The label and comment text may be also be edited as well as the node type.

In most applications the node states are set by scripts rather than edited manually, but they can also be useful to highlight particular nodes for a presentation. The hidden comment field is also used by scripts to hold node-specific data that may viewed by users or used as data by other scripts. It can also be entered manually as annotation of a particular node.

### 3.7 Enclosure nodes

Most of the node types available in RepNet are shapes that act as containers for the label text and or images. Instances of *enclosure* node types, however, act as containers for nodes and have additional features beyond those of the other node types. An enclosure node is created in the normal way, but it takes account of any selected nodes and positions itself to contain them.

When an enclosure is dragged to a different position it also repositions the enclosed nodes. The label of an enclosure note is displayed at the top of the node. Links may be drawn between an enclosure node, or nodes within it, and other nodes in the normal way. Since it often needs to be resized the resizing cursors appear at its corners even if has not been opened for editing and allow its size to be adjusted by dragging.

The *Group* node type in the default syntax for nets exemplifies the enclosure type. Figure 26 illustrates its use to group nodes in a concept map. A resize cursor is shown at its bottom right corner.

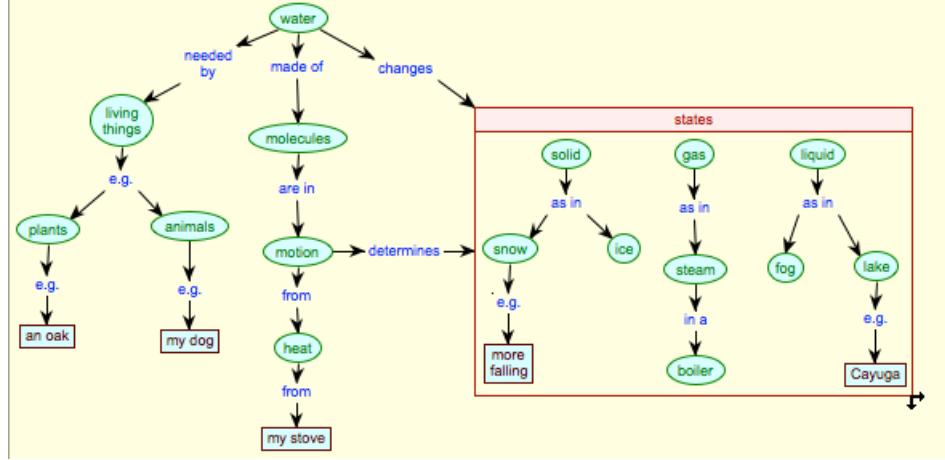


Figure 26: Enclosure node used to group concepts

Figure 27 shows enclosure types being used as logical elements in a semantic network that is equivalent to that of Figure 25. The enclosure node of type *construct* is interpreted by CNet as a logical macro representing the essential structure of a Kellian construct. The label of a *construct* enclosure node specifies the range of convenience of the construct whose poles are the nodes within the enclosure and are to be linked by mutual opposition links so long as this does not conflict with any ordination links. This results in same construct representation as before but in a more perspicuous notation.

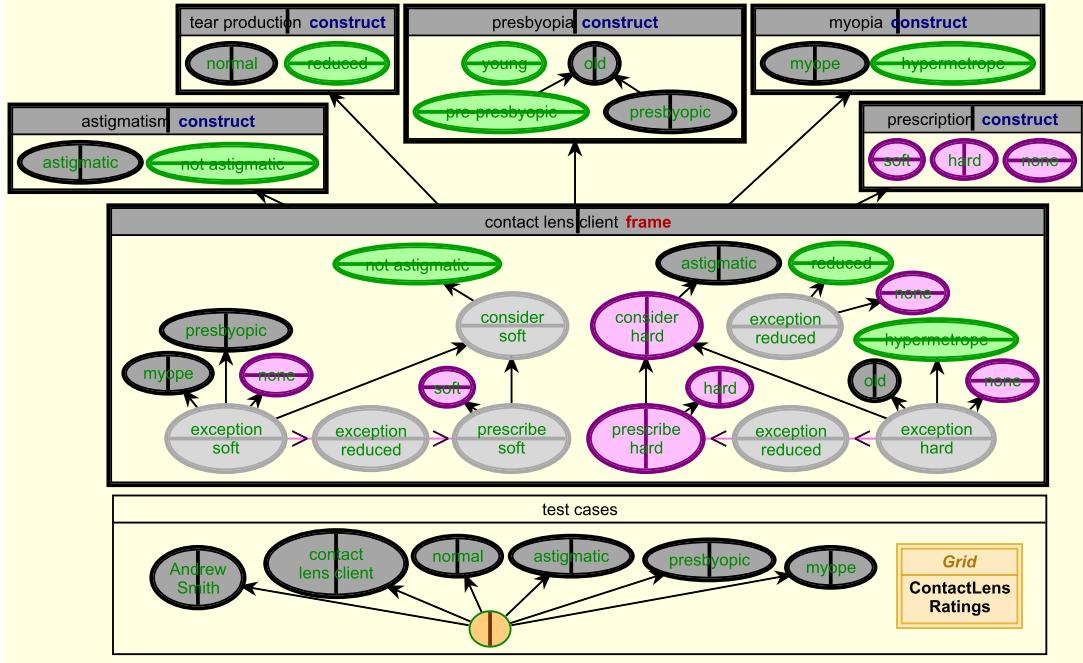


Figure 27: Enclosure nodes being used as logical macros

The enclosure node of type *frame* is interpreted by CNet as a logical macro representing the essence of a conceptual grid having the seven bottom nodes (those having no incoming links) are

*ideal elements* representing Kelly's *anticipatory intersects* that are subordinate to the abductive frame *contact lens client*.

The net is shown in the same state as that of Figure 25 where the CNET script has been used for abductive inference to anticipate the prescription applicable to the test case specified at the bottom left. The test cases themselves have been grouped in an enclosure node for convenience. It has no logical connotations but just serves to make the presentation more perspicuous to a person.

For purposes of selecting and dragging, an enclosure node is best conceptualized as the long rectangular region at the top that shows the label and, possibly, the type. The hand cursor only appears when the mouse is in this top rectangle, indicating that is where the mouse must be clicked to select the node or drag it to a new position, or double-clicked to open the node for editing.

The enclosed nodes on the lower rectangle behave normally and may be selected, dragged, opened for editing, linked, and so on, without any interaction with the enclosing node.

When the enclosure node is selected so are also all the enclosed nodes. When it is dragged all the enclosed nodes are dragged with it.

Sometimes it is useful to select only the enclosure node and this can be achieved by holding the *option* key down whilst clicking in the top rectangle. One can then, for example, delete the enclosing rectangle without also deleting all the enclosed nodes.

The eraser and arrow cursors for deleting and adding links are not restricted to the top rectangle but appear at the sides of the entire enclosure node, changing to the resizing cursors at the corners. Similarly the links themselves are computed relative to the entire enclosure node not just the top rectangle.

### 3.8 Undo in the Edit menu

When net data is changed the state of the grid before the change is recorded, the *Undo* (CMD-Z) option in the *Edit* menu becomes active, and selecting it or keying CMD-Z restores the state of the net before change. The last fifty state changes since the net was opened are recorded supporting multiple undoing. This information is discarded when the net is closed.

## 4 Copying and duplicating nets or subnets

RepNet supports the usual copy/paste and drag/drop techniques for copying items from one net to other or duplicating items within a net.

### 4.1 Copy and paste

Selected nodes (or the entire net if nothing is selected) and the links between them can be copied into the clipboard through the *Copy* command in the *Edit* menu or by keying CMD-C.

RepNet puts both the net data and an image in the clip in such a way that it is able to paste the data back into a net whereas other programs, such as word processors, paste the image in their documents.

The bitmap image in the clipboard is at screen resolution. To support publication of high-quality graphics the capability to export nets in SVG or as upscaled PNG or JPEG is also available (§5.2.1).

Net data in the clipboard may be pasted into another net through the *Paste* command in the *Edit* menu or by keying CMD-V. The pasted net is centred at the location last clicked in the net. Hence, to set the location for pasting, click in the net before pasting.

Nodes pasted may change appearance if the node type with the same name in the receiving net has a different graphic specification. If a node type with the same name does not exist in the receiving net then the node type data is also copied to the receiving net (making it possible to copy node types between nets).

Text data in the clipboard may be pasted into a net. Each line of text creates a new node of the type selected in the popup menu, and places the text in both the label and the note fields. This enables the user to edit the label, for example to shorten a journal citation, while retaining the full text in the note field. Tab characters in the data (typically from spreadsheet columns) are replaced by return characters in the label field, and hence generate new lines when the node label layout is calculated.

If the text data consists of a single line containing the substring “//:” then it is recognized as a URL and treated as if it were a file having the extension “html” (§4.3). A node is created having the text data in its note field that may be used to open the web page referenced by the URL. The node type will be that having the extension “html” if such a type is defined, otherwise the selected type.

### 4.2 Drag and drop

Drag and drop of net data behaves much the same as copy and paste except that RepNet puts the net data in the drag item for its own use, and the text of the node labels in for use by other applications. When data is dragged into a net it is centred on the point where it was dropped.

To distinguish between repositioning selected nodes and dragging copies of them, one must hold the *alt* or *option* key down when dragging in a node to indicate that it should become a drag item. When this is done the drag item is indicated by a grey rectangle initially surrounding the selected nodes.

Dragging a selected node, or nodes, and dropping them into the same net is a simple way of duplicating nodes or sub-nets.

### 4.3 Dragging files to a net

Files may also be dragged to a net. They are converted them to a text item containing a URL commencing with *file://* that provides a link to the file together with additional identification data. The name of the file is placed in the label field of a new node, and the URL in the note field.

If a node type has been specified for the file extension (§6.2) then a node of that type will be created. If not, and there is a node type specified for the extension \* then that a node of that type will be created. If not the selected node type will be created.

RepNet's default behaviour when a net is locked (§5.2.4) and a node is clicked is to try to open the file having the URL stored in the note field of the node. Hence, dragging files to a net may be used to set up a graphic structure providing access to those files (§8.2).

## 5 Contextual menus

Much of the functionality of RepNet is accessed through two context-sensitive popup menus, one available when the mouse is in the the net but outside all nodes (cursor symbol an arrow with a menu icon, ↗), and the other when the mouse is in the central region of a node (cursor symbol a hand with a menu icon, ⌂).

The menu icon indicates that a popup menu is available if one holds the *control* key down and clicks, right-clicks with a two-button mouse, or clicks and holds the mouse down without moving it. The menu options available are detailed in the following sections.

### 5.1 Contextual menu when cursor is over a node

Figure 28 shows the contextual menu that appears when the mouse is clicked while the cursor is over a node.

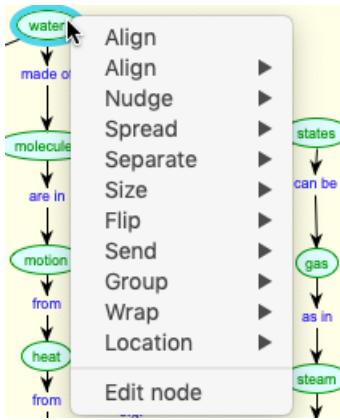


Figure 28: Contextual menu over node

Eleven options are available (where ► indicates specification in submenu):

**Align** Align the centres of the selected nodes (§5.1.1);

**Align** ► Align the sides, tops or bottoms of the selected nodes as specified (§5.1.1);

**Nudge** ► Move the selected nodes as specified (§5.1.2);

**Spread** ► Spread the selected nodes out as specified (§5.1.3);

**Separate** ► Separate the selected nodes by a specified amount as specified (§5.1.4);

**Size** ► Make the selected nodes the same size, or the minimum size, as specified (§5.1.6);

**Flip** ► Flip the positions of the selected nodes as specified (§5.1.5);

**Send** ► Send the selected nodes to a different layer or ranking in a layer as specified (§5.1.7);

**Group** ► Group or ungroup the selected nodes as specified (§5.1.8);

**Wrap** ► Change the wrap of the selected nodes as specified (§5.1.9);

**Location** ► Show and set the node location (§5.1.10);

**Edit node** Open the selected node for editing showing all the fields (not just the top three opened by double-clicking).

### 5.1.1 Aligning nodes

Selecting the top menu item, *Align*, aligns any selected nodes with the node under the cursor. It aligns the node centres either horizontally or vertically according to which requires the least movement. This usually corresponds to the user's intentions. Lines between the nodes become horizontal or vertical lines which gives the net a neater appearance.

It is often useful to align particular edges of the selected nodes rather than their centres. The submenu off the second *Align* menu provides the options to align the left, right, top, or bottom of the selected nodes with the node under the cursor, and to specify whether the centres are aligned horizontally or vertically. Figure 28 shows the contextual menu for specifying alignment and the results of selecting *Align Left*.

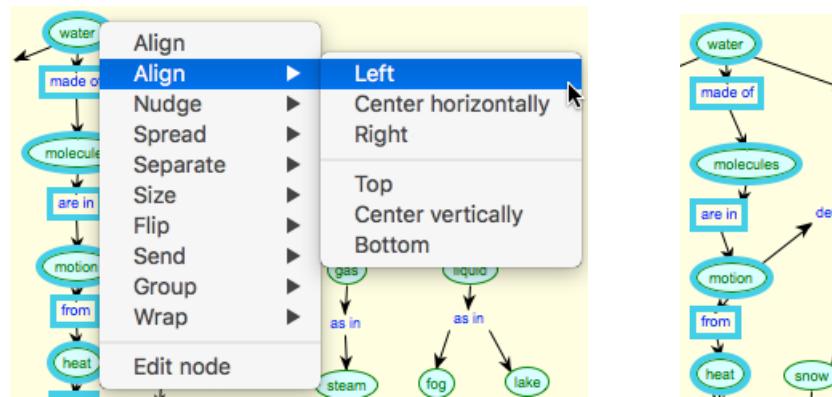


Figure 29: Alignment options (left) and outcome (right)

### 5.1.2 Nudging nodes

Dragging nodes or collections of selected nodes is the usual way of laying them out but it is not easy to make precise movements or to keep them only horizontal or vertical. The *Nudge* submenu enables such precise movements to be specified.

The options for precise movements of a small specified number of pixels are usually carried out using the arrow keys. The most useful nudge options are those to move horizontally or vertically by an amount specified by a signed number in the label field. Figure 30 shows the contextual menu for specifying nudges and the results of selecting *Nudge Horizontally* with -10 in the label field.

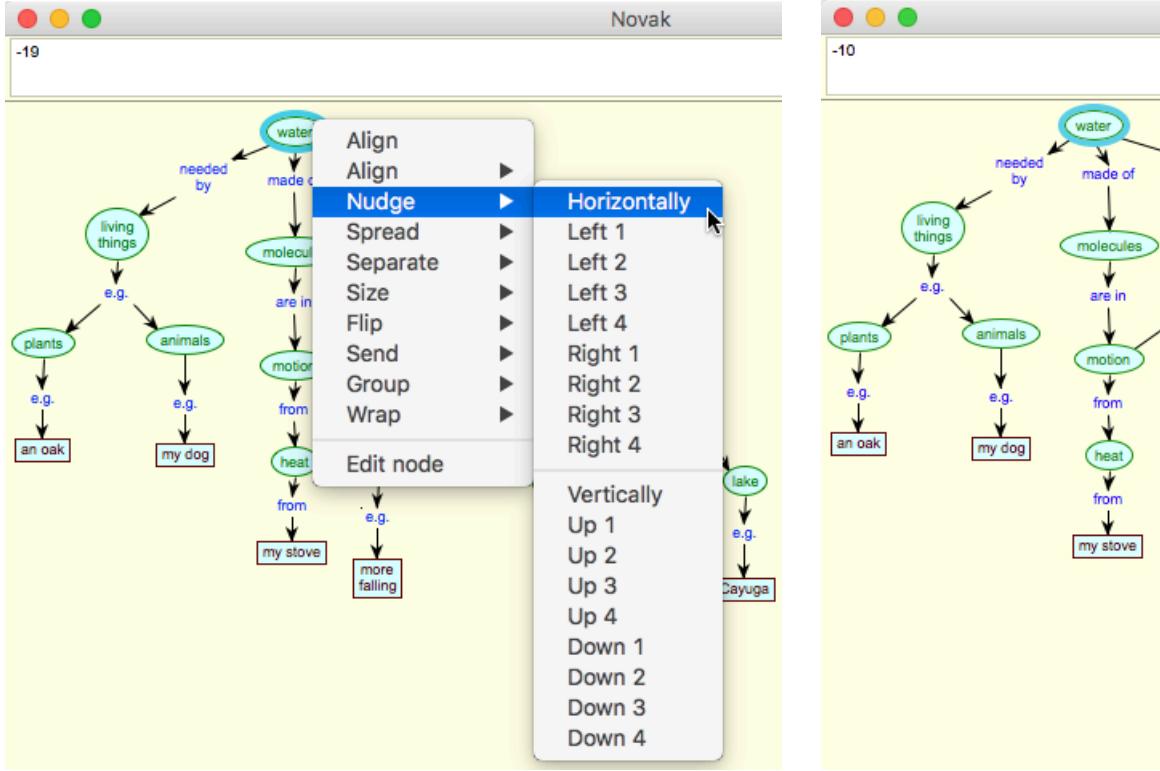


Figure 30: Nudge options (left) and outcome (right)

### 5.1.3 Spreading nodes

One often wants to create a tidy layout by spreading a collection of nodes to be evenly spaced horizontally or vertically. When the nodes have been roughly laid out manually the *Spread* submenu may be used to space the selected nodes evenly between the two end nodes of the selection. The spacing criterion may be that the centres of the nodes are evenly spaced or that the gaps between the nodes are equal. Figure 31 shows the contextual menu for specifying spreading and the results of selecting *Spread Gaps Vertically*. Note that the bottom and top selected nodes have not moved.

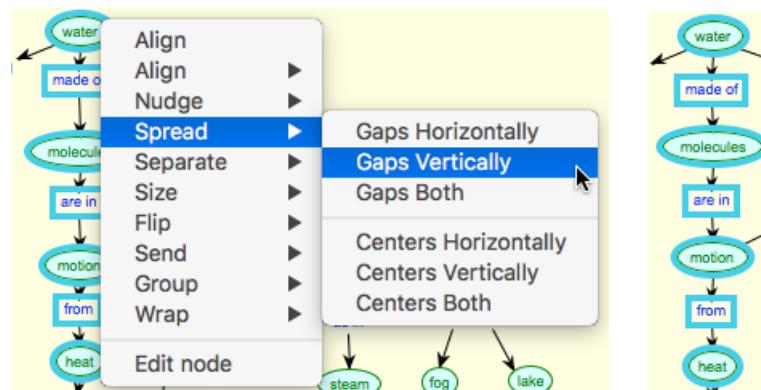


Figure 31: Spread options (left) and outcome (right)

### 5.1.4 Separating nodes

Sometimes one wishes to specify the separation between node edges or centres rather than spreading node between two end points. The *Separate* submenu may be used to space evenly the selected nodes on either side of the node clicked by the spacing specified in the label field. Figure 32 shows the contextual menu for specifying spreading and the results of selecting *Separate Gaps Vertically*. Note that the node clicked has not moved.

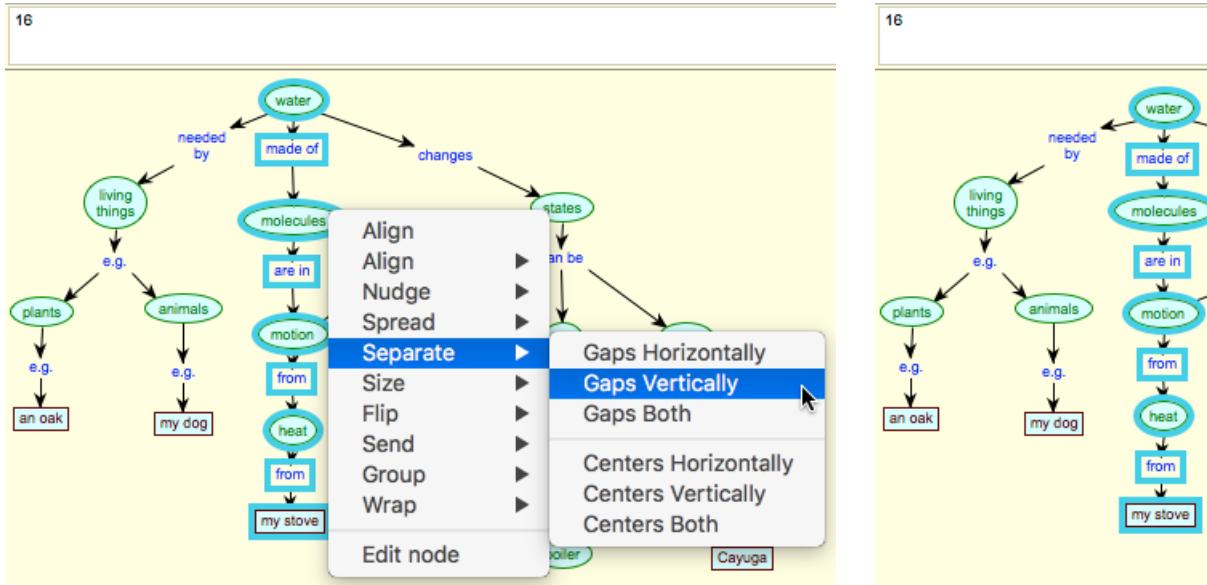


Figure 32: Separation options (left) and outcome (right)

### 5.1.5 Sizing nodes

It is useful to be able to adjust the size of related nodes to be the same to emphasize their commonality, and to minimize the size of a node to fit the nodes more closely. The *Size* submenu allows the selected nodes to be resized to be the same as the largest, horizontally, vertically or both, and to be adjusted to the minimum size that fits the text without changing the wrap (Figure 33).

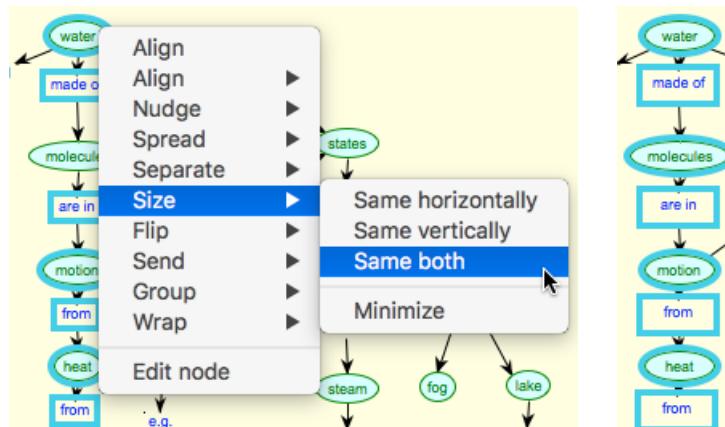


Figure 33: Size options (left) and outcome (right)

### 5.1.6 Flipping nodes

Sometimes one wishes to flip a collection of nodes horizontally or vertically so that the node at one extreme exchanges position with the node at the other extreme and intermediate nodes are also change so that the node spacing remains unchanged. Figure 34 shows the contextual menu for specifying flipping and the results of selecting *Flip Vertically*.

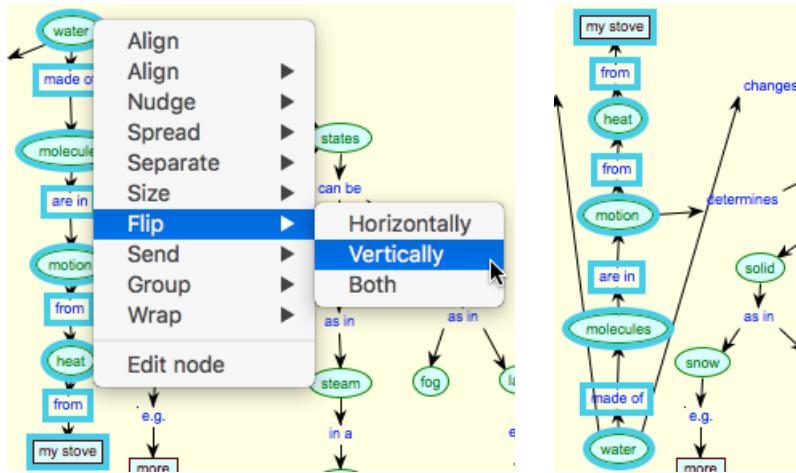


Figure 34: Flip options (left) and outcome (right)

### 5.1.7 Sending nodes to a different layer or location within a layer

Nodes do not overlap in most applications of RepNet but in some situations such as annotating a net and highlighting significant features it is useful to be able to create nodes that are located behind other nodes and their links. Nets have a layer structure which facilitates this, and the *Send* submenu may be used to manage the location of nodes within the layers.

Nets have four graphic layers: the front nodes; the links; the back nodes; and the background (Figure 35). The background layer may be used to provide an overall coloured background and is drawn first. Back nodes may be used to draw behind the links and front nodes and are drawn second. Links are drawn third and front nodes last.

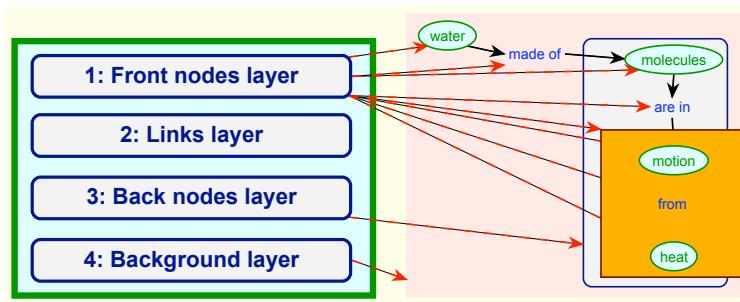


Figure 35: Layer structure of nets (left) and example of layered net (right)

The drawing of a sample net on the right illustrates this. The labelled nodes constitute the linked chain from *water* to *heat* from the net used in previous examples and is in the front nodes layer. The

rounded corner rectangle around part of the chain is in the back nodes layer and the nodes and links in the chain are drawn on top of it. The square corner rectangle is at the back of the front layer, and the nodes in the chain are drawn on top of it but their links are obscured because they are drawn below it (as are the dashed metalinks showing the relationship of the net graphics to the layers).

Figure 36 shows on the left the contextual menu for specifying sending when the mouse was clicked over a rectangular node at the top of the front layer that obscures part of the net.

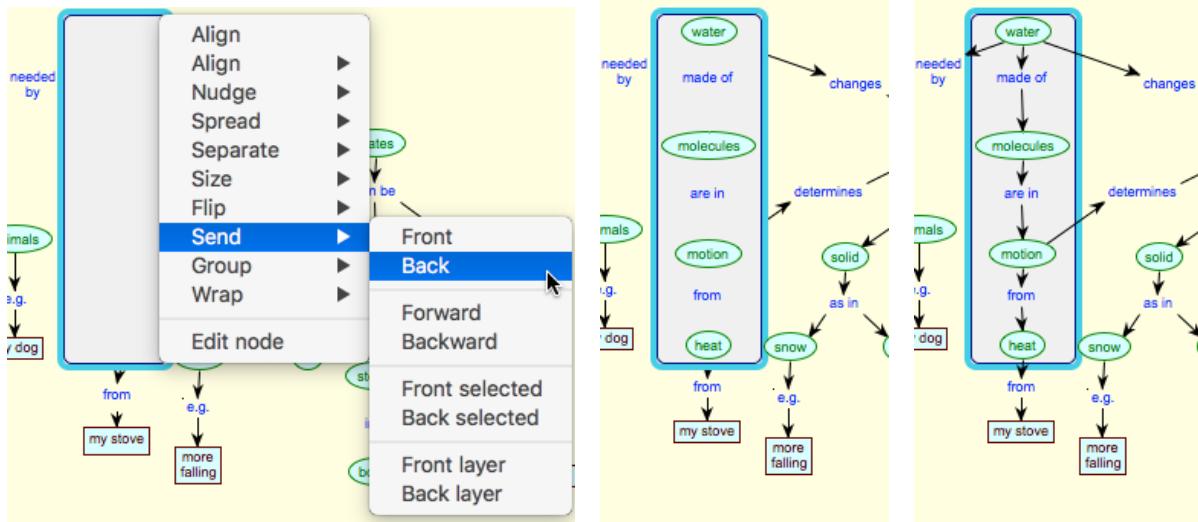


Figure 36: Send options (left), *Send Back* (centre), *Send Back layer* (right)

The centre section shows the result of selecting *Send Back* (to the back of the front nodes layer) and the right that of selecting *Send Back layer* (to the front of the back nodes layer). It is apparent that the role of the back layer is to allow nodes to be located behind the links layer. Figure 37 shows RepGrid analyses annotated using overlapping nodes to colour the background.

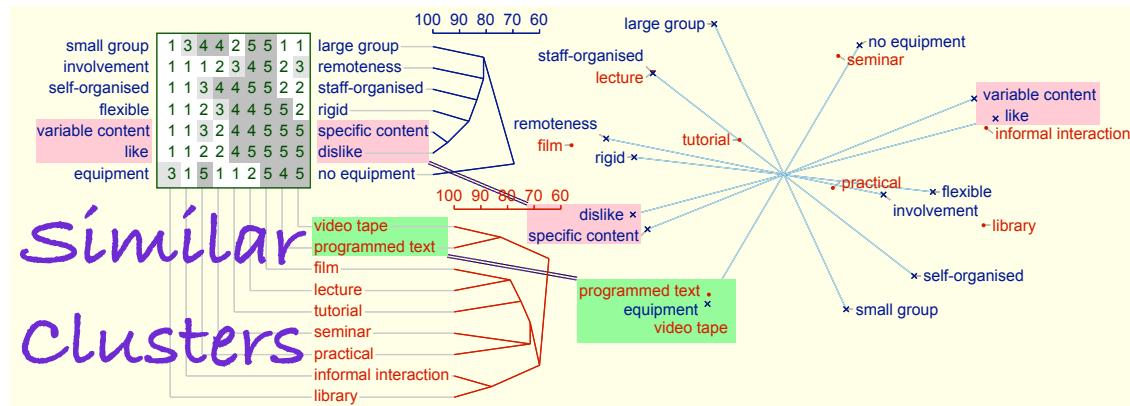


Figure 37: Annotating analyses using layers

The *Forward* and *Backward* options allow a node to be stepped past other nodes in its layer one at a time. The *Front selected* and *Back selected* options allow a node to be placed immediately in front of or immediately behind a collection of one or more nodes.

Thus, the *Send* submenu allows nodes to be moved between the front and back node layers, and the drawing order of nodes within a layer to be changed. These options usually suffice for the normal range of applications of RepNet. More complex requirements may be addressed by exporting the net in SVG format which preserves its structure and makes the visual representation of the editable in the general-purpose graphics tools that support SVG.

### 5.1.8 Grouping nodes

Collections of nodes may be grouped and a group behaves in many respects as if it was a single composite node. In major part this is because selecting any node in a group selects them all so that nodes within a group cannot be dragged or nudged into different locations relative to one another. The *Group* submenu offers options to group, or ungroup, selected nodes (Figure 38).

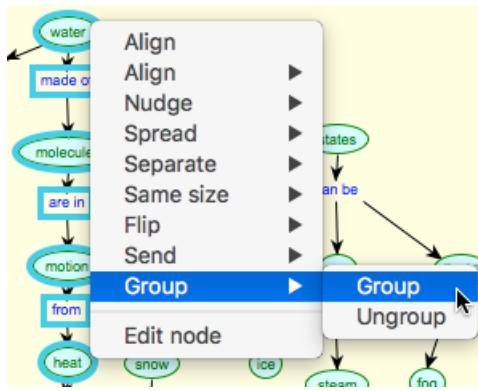


Figure 38: Grouping options

Groups can be arbitrary collections of nodes although they are very often spatially contiguous. Groups can be nested to any depth by grouping groups with other nodes and groups. When nested groups are ungrouped the nesting is reversed one step at a time. Enclosure nodes can be grouped with other nodes including those that they contain.

Links within a group, and from nodes within to nodes outside the group, behave as if the nodes remained independent entities. The cursor changes to an arrow or eraser for each node within the group and can then be dragged to any other node to create or delete a link. The align, spread and separate operations treat groups as if they were rectangle nodes having a size that is the minimum enclosing rectangle of the grouped nodes.

### 5.1.9 Wrapping nodes

The soft wrapping of the label text that occurs automatically as a node is resized is a convenient way of laying out nodes in a net. It is complemented by the hard wrap that can be achieved by inserting returns in the label. However, when nets are moved to another computer with fonts having different metrics from those where it was created, the soft wrapping points may be changed in a way that makes the layout very untidy.

For nets that are intended to be portable it may be appropriate to change all of some of the soft wraps to hard ones. Change in font metrics will then generally result only in slight changes in node

sizes which may be simply accommodated if the net is not too tightly spaced, or readily adjusted if necessary. The *Wrap* submenu provides options to change the wrapping of the selected nodes from soft to hard, or *vice versa* (Figure 39). It does not change the size of a node, and has no effect on enclosure nodes or on nodes where more than one consecutive return characters have been used to insert multiple line-spacing.

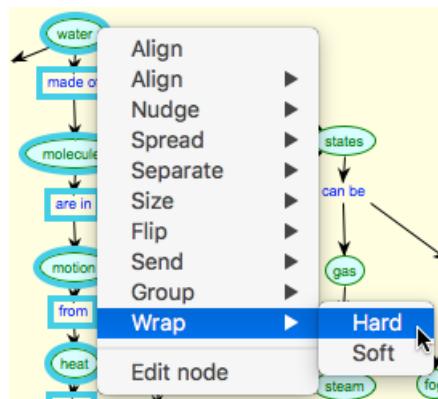


Figure 39: Wrapping options

### 5.1.10 Showing and setting node locations

Sometimes when one is layout diagrams very precisely it is occasionally useful to be able to view the location and size of the selected nodes, and/or adjust their locations. The *Location* submenu allows the location of the selected node(s) to be shown or set. Figure 40 shows the *Show* option used to place the locations and labels of the selected nodes in the top text field.

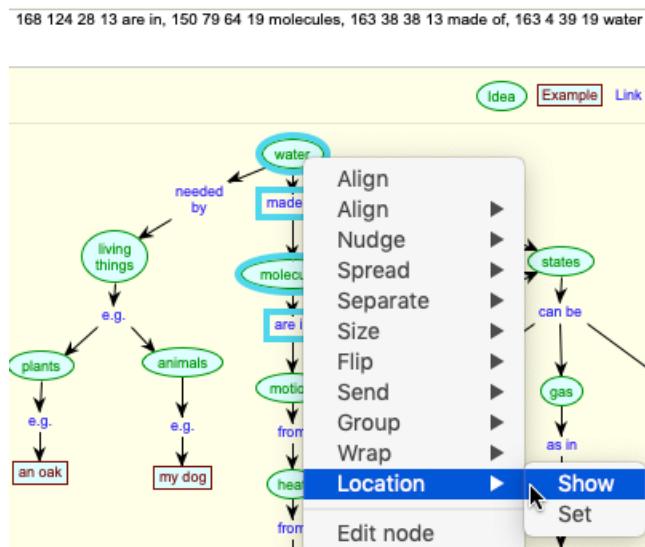


Figure 40: Showing and setting node locations

The positions of the selected nodes may be changed by editing the first two numbers, left and top, of the first node listed and selecting the "Set" option (with the same nodes selected). The first node listed will be moved to the new location and the remaining selected nodes will be moved by the same amount so that their relatives locations are unchanged. The sizes and labels of the nodes moved are unchanged.

## 5.2 Contextual menu when cursor is not over a node

Figure 41 shows the contextual menu that pops up when the mouse is clicked in the net outside a node.

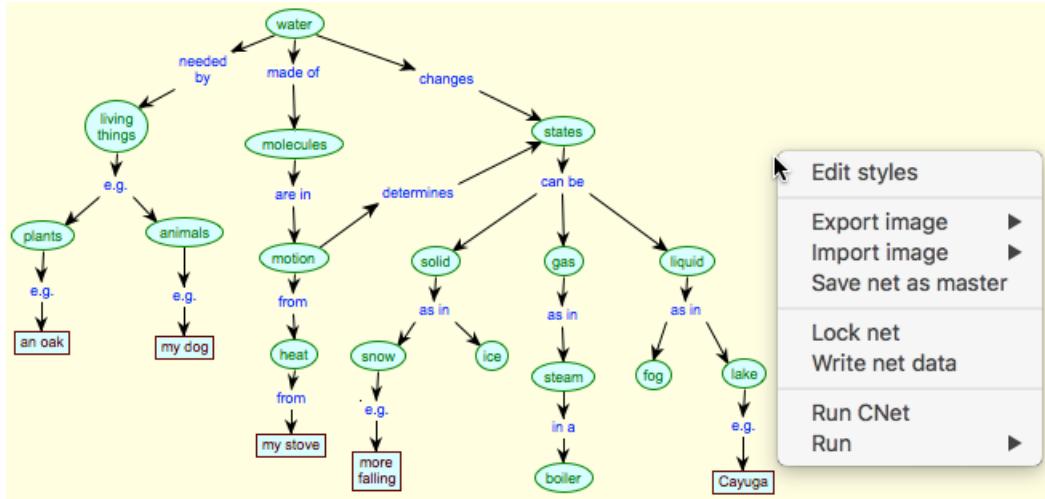


Figure 41: Contextual menu outside nodes

Eight options are available (where ► indicates specification in submenu):

- Edit styles** Open a styles floating dialogue to create and edit net, node, link and state styles (§6);
- Export image** ► Export the net, or part of it, in the selected image format (§5.2.1);
- Import image** ► Import a PNG or JPEG image to the net as part of a node (§5.2.2);
- Save net as master** Save the net in a format that automatically creates a copy when opened (§5.2.3);
- Lock net** Lock the net so that it cannot be edited (§5.2.4);
- Write net data** Write the node positions, size and names in a text window (§5.2.5);
- Run CNet** Run the last script that was previous run (in this case, CNet) (§8.1);
- Run** ► Run the specified script (§8.1).

### 5.2.1 Export the net as an SVG, PNG or JPEG image

The **Export image** submenu allows a selected part of the net, or the entire net if none is selected, to be saved to a file in SVG and scaled PNG or JPEG formats (Figure 42).

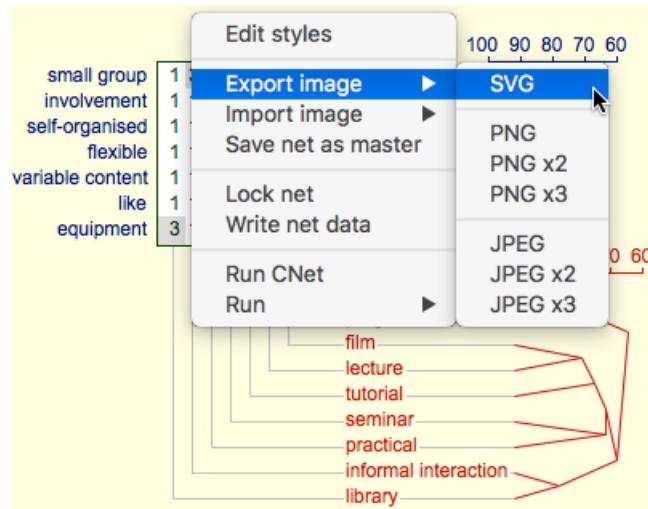


Figure 42: Saving the entire, or selected part, of a net as a SVG, PNG or JPEG file

SVG provides a precise, high-resolution representation of the net in a vector graphic format supported by web browsers and graphics editing packages such as Illustrator, EazyDraw and InkScape. It scales and prints to any size without distorting the image, and can be transformed to PDF vector graphic format.

PNG provides a bitmap representation of the net corresponding to the screen image that will scale but not so well as SVG.

JPEG encodes the image as a picture and can be used to compress it, but RepNet creates it at JPEG's highest quality level and this provides similar performance to PNG.

SVG is, as yet, supported by only a few document processors but is readily converted to PDF vector graphics suitable for use in Latex and is a compact format that can be embedded in HTML web pages.

PNG and JPEG are both supported by most document processors but do not print very well if created at screen resolution. Hence, RepNet provides the option to create them as two or three times screen resolution which, when scaled back in the document processor, will print at much higher quality.

Saving the image as SVG and opening it in a vector graphics editor allows its graphic content to be edited and annotated in greater detail than that supported in RepNet. The SVG created is structured around the modules that the Rep Plus analyses created and these appear as groupings in the graphics editors. This enables the meaningful parts of the image to be moved, scaled, recoloured, and so on, very naturally. If the image is successively ungrouped into its ultimate graphic primitives then every part of becomes editable.

Figure 43 shows a net completely ungrouped in EazyDraw with all its graphic components selected.

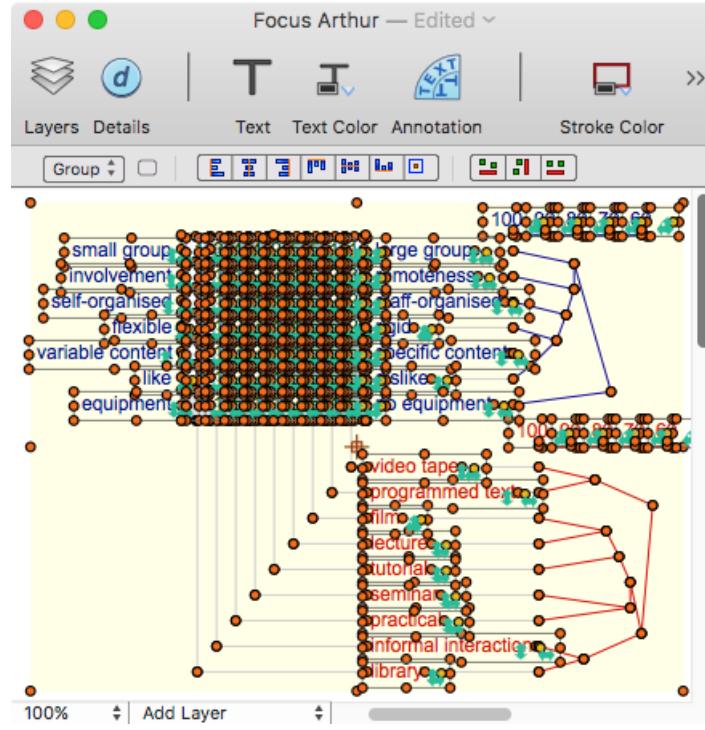


Figure 43: SVG file opened and ungrouped in EazyDraw

### 5.2.2 Import a PNG or JPEG picture to the net as part of a node

It is often useful to add images to a net, for example, screen dumps, photos of elements in a grid, and so on. The *Import image* submenu allows the image in a PNG or JPEG file to be shown as within a node in the net (Figure 44).

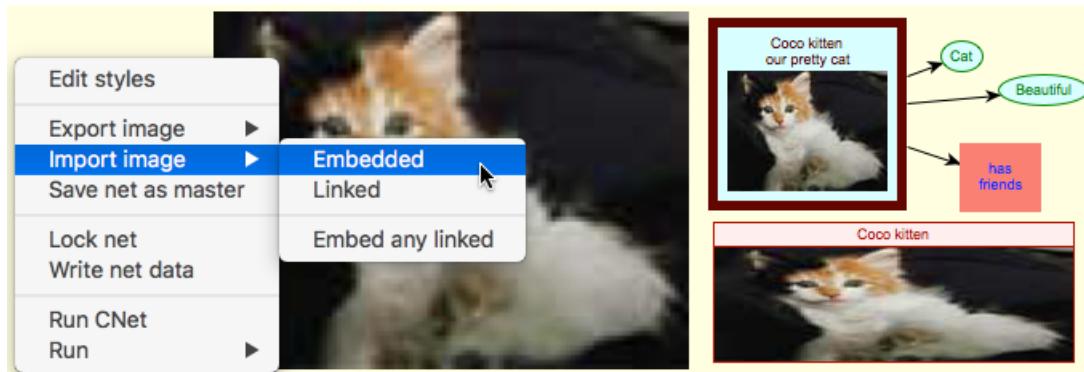


Figure 44: Importing an image as a node

The image data may be stored as part of the net by selecting the option *Embedded* or a link to the file may be stored selecting the option *Linked*. The latter option is useful if the image files are very large and it is better not to duplicate the image data in the net. The link stored includes both the absolute path to the image and one relative to the net (if it has already been saved), so the data and links are portable if stored in the same relative locations.

If some, or all, of the image data has been stored in a net through links then it is possible to make the net independent of the image locations by using the option *Embed* any links that uses the links to access the image data and embeds it in the net.

When an image is imported it is embedded in the selected node type (which can be the null type when no node type is selected). The node type options specify where the node will be located relative to the label (if there is one), whether the image resizes as the node is resized and, if so, whether the aspect ratio should be maintained (§6.2). The node behaves in other respects as a normal node and can have links and a state.

For non-enclosure nodes the image is shown in the body of the node separated from the label by a specifiable amount. For enclosure nodes the image is shown in the enclosing area. For example, Figure 44 shows the same image imported to: a null type node where only the image itself is displayed but is resizable with aspect ratio maintained (left); a rectangular node with a light green background, a thick brown frame and a label, not resizable (top right); an enclosure node, resizable, aspect ratio not maintained (bottom right);

### 5.2.3 Saving the net as a master file

It is sometimes convenient to save a net as a master file that cannot be edited but, when opened, creates a copy of itself. For example, the master may be stored on a shared computer in a teaching laboratory where the net is intended to be used and edited by students to create personalized versions. The *Save net as master* option provides this capability.

Note that master nets themselves cannot be further edited, but this can be achieved in practice by saving the copy under the original name (the operating system file protection capabilities may be used to prevent this happening accidentally).

### 5.2.4 Locking and unlocking the net

The *Lock net* option may be used to put the net into a state where it cannot be edited. This is usually done by scripts supporting interactivity since mouse actions in a lock net are sent to the current script for interpretation (§8.1.1). However, it may sometimes be useful to lock the net to avoid accidental editing or to use the default script that shows the cursor as a button when it is over the node and interprets a click as a request to open a file or web page whose URL is stored in the note field of the node (§8.2).

It is up to scripts to provide the capability to unlock a net, for example, the default script unlocks the net if there is a double-click in the net outside the nodes.

### 5.2.5 Writing the net data

The *Write net data* option opens a text window and writes data about the selected nodes (or all the nodes if none are selected). The data includes the positions and sizes of the nodes which may be useful, for example, in adjusting the layout of a net. It also includes the note fields which can only otherwise be accessed node by node and where it may be useful to view those of several nodes through one action.

# 6 Creating/editing types and styles for nodes, links and states

Each net is a conceptual structure expressed in a visual language whose possible statements are specified through the definition of a collection of node, link and state types. Once the language is defined its node, link and state types become available through the editing area that can be accessed at the top of a net (§3), but a separate capability is needed to create and edit the types. This is accessed through the *Edit styles* option at the top of the contextual menu available when the mouse is in the net but not over a node (§5.2).

Selecting *Edit styles* in the net whose styles are shown in Figure 25 brings up the dialogue shown in Figure 45. This window is in the *floating* layer above normal windows and is only visible if the associated net window is at the front. Thus, several nets can have their styles editing dialogues open at the same time but only that for the top net window will be visible.

The styles dialogue has four panes accessed through the tabs at the top for: overall *net* options; *node* styles; *link* styles; and *state* styles. It opens with the *net* options showing (Figure 45). The following sections detail the options available in each pane.

## 6.1 Net, node, link and state styles

The creation of a family of styles defining the visual language available for creating a net can take significant effort and, once it is completed, is likely to be used in many nets. This is easy to achieve for new nets by saving a blank net as a *master* net (§5.2.3), through the *Copy Styles* button in the *Rep-Net* panel of the manager window which creates a copy of a net having only its styles (§2.2.3), or by copying a net file and deleting the net.

It is also often useful to be able to apply the new styles to some existing nets. One can transfer a style from one net to another by dragging a node with that style from the first net to the second and then deleting the node, or copying and pasting the node. However, if the node has a style with the same name as one already in the net then the node is restyled in that of the net to which it is dragged or copied.

The *Copy styles from a net* panel at the top of the net styles pane offers the option of copying all the styles from one net to another and having the copied styles preempt any styles having the same name so that any existing nodes in that style are restyled with the new style.

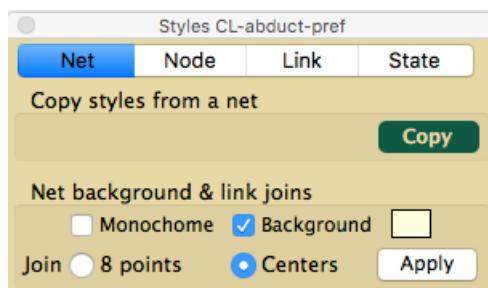


Figure 45: Net style options

Clicking on the *Copy* button brings up an *Open File* dialog to select the file with the styles to be copied, or the file may be dragged to the button. The new styles replace any with the same name but do not delete old styles not in the copied net. Hence, if one wishes to only replace some styles it is possible to do so by creating a copy of a file with the desired styles, deleting those not required, and dragging that edited copy to the *Copy* button.

The *Net background and links joining* panel at the bottom of the net styles pane offers the options of: displaying the net in monochrome with all the colours shown as shades of grey; displaying the background layer; specifying the background colour; and specifying whether links join nodes only at their corner and side centres or on lines projected through the node centres (§24).

The *Apply* button applies any changed parameters to the net so that the effect is immediately visible and can be undone or adjusted if desired.

### 6.1.1 Using the colour swatches to select colours

Each of the panes uses one or more colour swatches to set any colours required in a style. Clicking in the left half of a swatch brings up the standard CSS named colours in a selection menu; right half brings up the native colour selection widget for the platform (Figures 46 & 47).

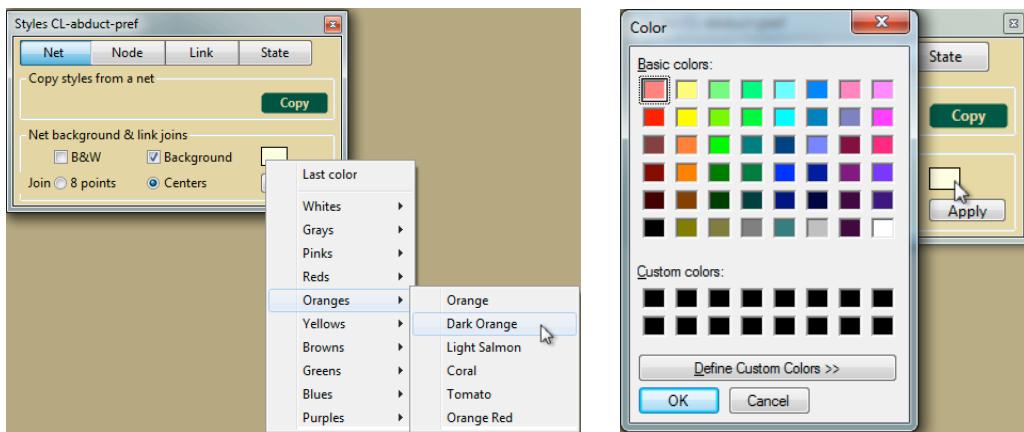


Figure 46: Colour selection under Windows—mouse clicked in left of swatch, in right of swatch

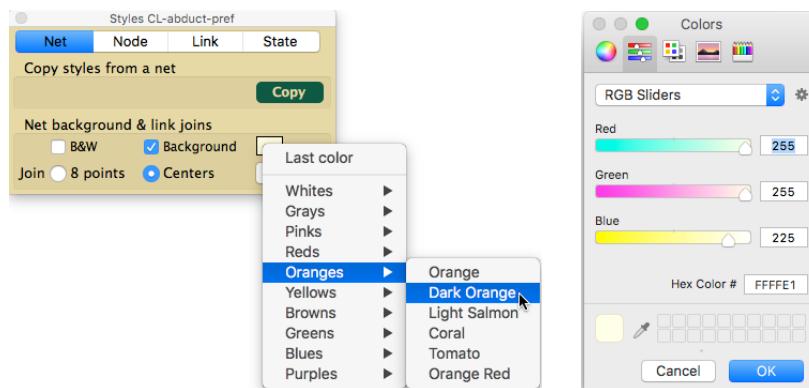


Figure 47: Colour selection under OS X—mouse clicked in left of swatch, in right of swatch

Rep Plus keeps track of the colour of the swatch last clicked in any dialogue and makes this an available option at the top of the named colours menu. This can be used for rapid setting of the same colour in several swatches, not only when the same colour is required but also, for example, when the same hue is to be used with differing degrees of saturation and/or brightness.

## 6.2 Node types and styles

Clicking on the *Node* tab switched to the node styles pane which provides facilities to define the node types and their styles (Figure 48).

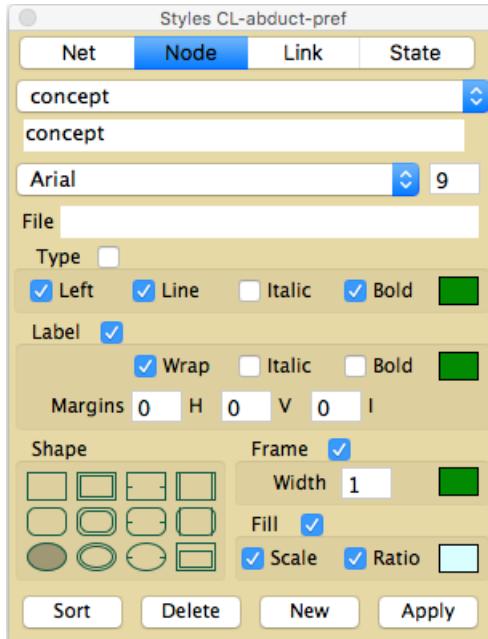


Figure 48: Node style options

Nodes have: a *type name* that may be shown at the top of the node; a *label* that may be shown in the body of the node; a *frame* that may be shown as a surround to the node; and a *fill* that may be shown as the background to the body of the node. They may also have a *note* that is not shown and does not require styling, and an associated image that is shown beneath label and whose scaling parameters are part of the body section.

The top line is a popup menu for selecting the node type to be edited (Figure 49), and the line below a text field to edit the name of the type or enter the name of a new type. Selecting a particular node type causes its current parameters to be shown.

The third line shows the font name and font size to be used for any text displayed as part of the node.

On the fourth line is a text field that may be used to enter a comma-separated list of any file extensions that might be associated with this the node. This is used to associate the node style with the node generated when a file is dragged to the net window and dropped (§8.2). For example, a

node type intended to represent document files, such as notes in a variety of formats, might have the following list of extensions, "pdf,odt,doc,docx,pages,txt,html". Note that the period character commencing the extension is omitted. If there is an asterisk, "\*", in the list then the last node type having one is selected if no more specific match has been found.

The *Type* panel is a group of controls styling the presentation of the node type name if it is to be shown at the top of the node. The check box at the top specifies whether the type name is to be shown. The four check boxes below specify: whether, if the node shape is an enclosure, the type name is to be shown on the left or right of the label; whether a line is to be drawn underneath the type name to separate it from the body of the node; and whether the text should be styled bold and/or italic. The colour swatch on the right specifies what colour should be used to display the node type.

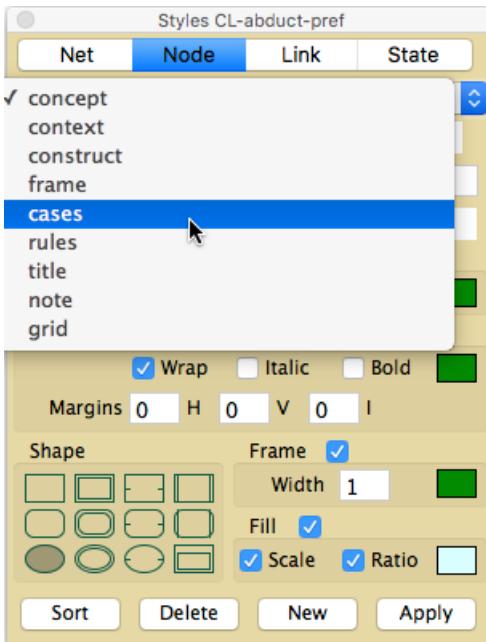


Figure 49: Node type selection

The *Label* panel is a group of controls styling the presentation of the node label if it is to be shown in the body of the node. The check box at the top specifies whether the label is to be shown. The three check boxes below specify: whether the label text should be wrapped; and whether the text should be styled bold and/or italic. The colour swatch on the right specifies what colour should be used to display the node label. The three text boxes at the bottom specify incremental adjustments (which can be negative) to the default margins around the text horizontally and vertically, and between the image, if any, and the label. Usually the defaults are adequate but occasionally some fine tuning is appropriate.

The *Shape* panel allows the shape and decoration of the node frame to be selected from the twelve possibilities illustrated. Clicking on a node shape greys it out to show it has been selected. The node shape on the bottom right is that of an *enclosure node* that behaves in a different way from that of the other nodes (§3.7).

The *Frame* panel contains two controls styling the presentation of the node frame if it is to be shown around the node. The check box at the top specifies whether the frame is to be shown. The *Width* text box specifies the width of the outer shape of the frame in pixels. The colour swatch specifies what colour should be used to display the frame,

The *Fill* panel contains three controls specifying the scaling of images and the colour of the fill. The check box at the top specifies whether the fill is to be shown. The *Scale* check box specifies that any associated image should be scaled to fit the node and resize as the node is resized. The *ratio* check box specifies that the original aspect ratio of the image should be maintained as it is scaled to fit the node. The colour swatch specifies what colour should be used to fill the node.

When nodes are incrementally created they appear in the editing area at the top of the net in the order in which they were created. Sometimes there is an alternative order which would group the nodes in a way that is more meaningful to someone creating a net. The *Sort* button at the bottom left brings up a dialogue box listing the node types such that they can be dragged into the preferred order. When this has been done the *Apply* button adjusts the node types in the net editing area accordingly.

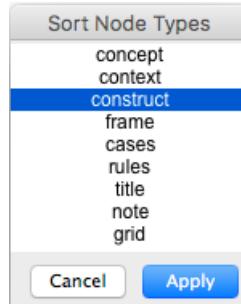


Figure 50: Node type sorting

The *Delete* button at the bottom deletes the selected node type. If there are nodes of this type in the net they are assigned the first node type on the list of types.

The *New* button at the bottom creates a new node type having the name and parameters specified. Node types must have different names and an error message is shown if the specified name is already in use.

The *Apply* button at the bottom changes the parameters of the selected node type to those specified. If the name has been edited then this is also changed although an error message will be shown if the specified name is already in use.

### 6.3 Link types and styles

Clicking on the *Link* tab switched to the link styles pane which provides facilities to define the ink types and their styles (Figure 51).

Links are represented as directed, decorated lines. They have: a *type name*; a *line style*; a *centre decoration style*; a *head decoration style*; and a *tail decoration style*.

The top line is a popup menu for selecting the link type to be edited (Figure 52), and the line below a text field to edit the name of the type or enter the name of a new type. Selecting a particular link type causes its current parameters to be shown.

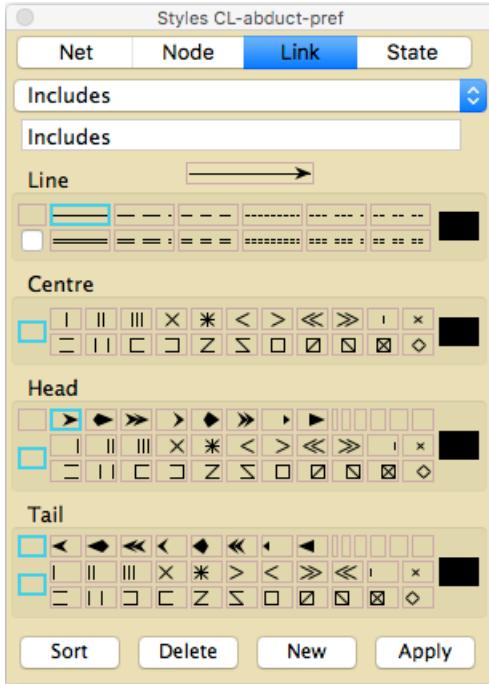


Figure 51: Link style options

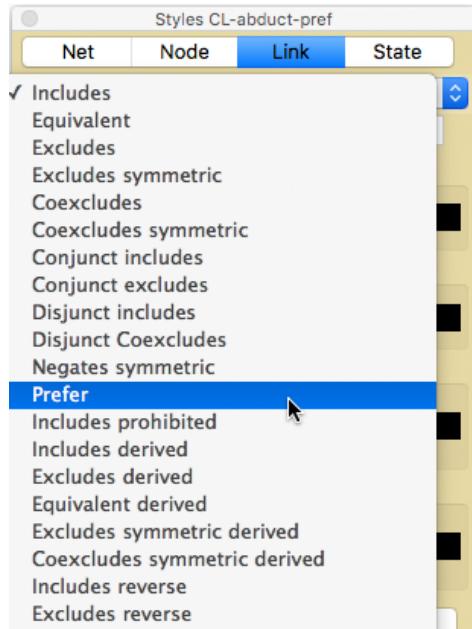


Figure 52: Link type selection

Centred under these is a box showing the link graphic that has been specified. This changes dynamically as the specification is changed so that the resulting graphic is always visible.

The *Line* panel offers a palette of different line styles, single or double, continuous or dashed in various patterns that are selected by clicking on one. The blank box at the top left specifies that no line should be drawn. The check box beneath it specifies the direction of drawing which makes no difference to continuous lines but can affect the way patterned lines merge with the decorations at their heads or tails. The colour swatch on the right specifies what colour should be used to display the line.

The *Centre* panel offers a palette of different decorations for the centre of the line that are selected by clicking on one. The blank box at the top left specifies that no centre decoration should be drawn. The colour swatch on the right specifies what colour should be used to display the centre decoration.

The *Head* panel offers, on the top row, a palette of different decorations for the head of the line and, on the following two rows, additional decorations that are placed just behind them. Decorations from palettes are selected by clicking on them. The blank box at the left of each palette specifies no decoration from that palette should be drawn. The colour swatch on the right specifies what colour should be used to display the head decoration.

The *Tail* panel offers, on the top row, a palette of different decorations for the tail of the line and, on the following two rows, additional decorations that are placed just behind them. Decorations from palettes are selected by clicking on them. The blank box at the left of each palette specifies no decoration from that palette should be drawn. The colour swatch on the right specifies what colour should be used to display the tail decoration.

The many line styles, decorations and colours makes it possible to create some florid link styles, such as  but the varied options are useful in emulating the symbolism of different disciplines, for example, in representing a wide range of logical and metalogical relations in a coherent fashion, as exemplified in (Gaines, 2015) and Figure 25.

The *Sort* button at the bottom left brings up a dialogue box listing the link types such that they can be dragged into a preferred order. When this has been done the *Apply* button adjusts the link types in the net editing area accordingly.

The *Delete* button at the bottom deletes the selected link type. If there are links of this type in the net they are assigned the first link type on the list of types.

The *New* button at the bottom creates a new link type having the name and parameters specified. Link types must have different names and an error message is shown if the specified name is already in use.

The *Apply* button at the bottom changes the parameters of the selected link type to those specified. If the name has been edited then this is also changed although an error message will be shown if the specified name is already in use.

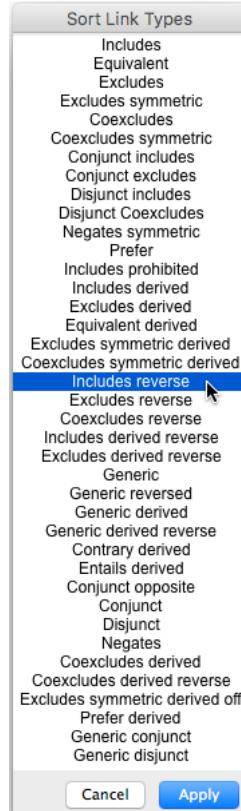


Figure 53: Link type sorting

## 6.4 State types and styles

Clicking on the *State* tab switched to the states types and styles pane which provides facilities to define the state types and their styles (Figure 54).

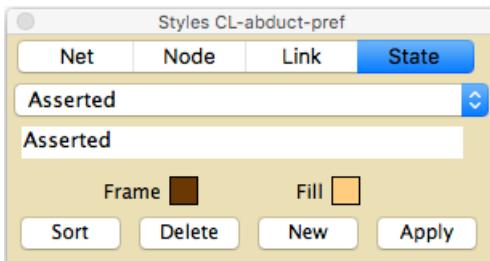


Figure 54: State style options

State styles have two parameters, the colour of the frame and that of the fill (§3.6). The use of these to indicate the state of a node is specific to the particular node and not part of the type specification.

The top line is a popup menu for selecting the state type to be edited (Figure 55), and the line below a text field to edit the name of the type or enter the name of a new type. Selecting a particular link type causes its current parameters to be shown.

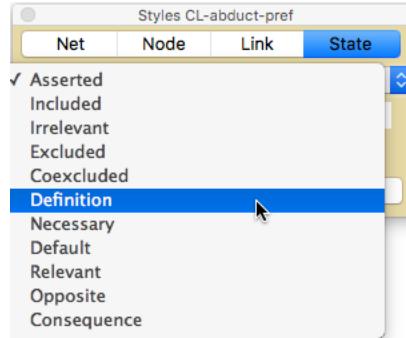


Figure 55: State type selection

The two swatches may be used to select the colour of the frame and file of the state decoration.

The *Sort* button at the bottom left brings up a dialogue box listing the states types such that they can be dragged into a preferred order. When this has been done the *Apply* button adjusts the state types in the net editing area accordingly.

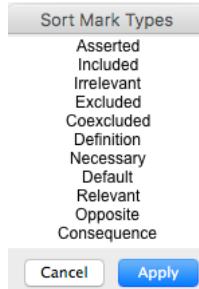


Figure 56: State type sorting

The *Delete* button at the bottom deletes the selected state type. If there are states of this type in the net they are assigned the first state type on the list of types.

The *New* button at the bottom creates a new state type having the name and parameters specified. State types must have different names and an error message is shown if the specified name is already in use.

The *Apply* button at the bottom changes the parameters of the selected state type to those specified. If the name has been edited then this is also changed although an error message will be shown if the specified name is already in use.

## 7 Graphic structures from other applications in RepNet

As well as providing a tool for creating graphic plots having a node/link structure, such as concept maps and state-transition diagrams, RepNet provides editing and printing capabilities for the graphic output produced by other Rep Plus tools such as RepGrid, RepGrids and RepScript.

Some of this output, such as that from *PrinGrid* and *CrossPlot*, comprises node/link structures that can be fully edited in RepNet, but some of it, such as that from *Display*, *Focus*, *Compare*, *Synopsis* and *Scree*, comprises vector graphic plots that behave as nodes having incorporating unitary graphic items that can be only partially edited in RepNet. Imported PNG and JPEG images also behave in this way.

The PNG and JPEG images may be resized, and the RepGrid and RepGrids plots can have their colour schemas changed since these are represented as RepNet node and link styles.

RepNet graphics can be exported to PNG, JPEG or SVG, and the SVG form preserves the full vector graphic structure of the RepGrid and RepGrids plots so that they may be edited at a lower level of detail if required.

The following structures detail the graphic structure of various RepGrid and RepGrids plots.

### 7.1 Standard styles of RepGrid plots

The styling of all RepGrid analysis plots is specified through the RepGrid Style dialogue (Figure 57 left), and the relevant styles information for any particular plot is encoded in the RepNet representation as a set of five node styles, and four link styles (Figure 57 centre).

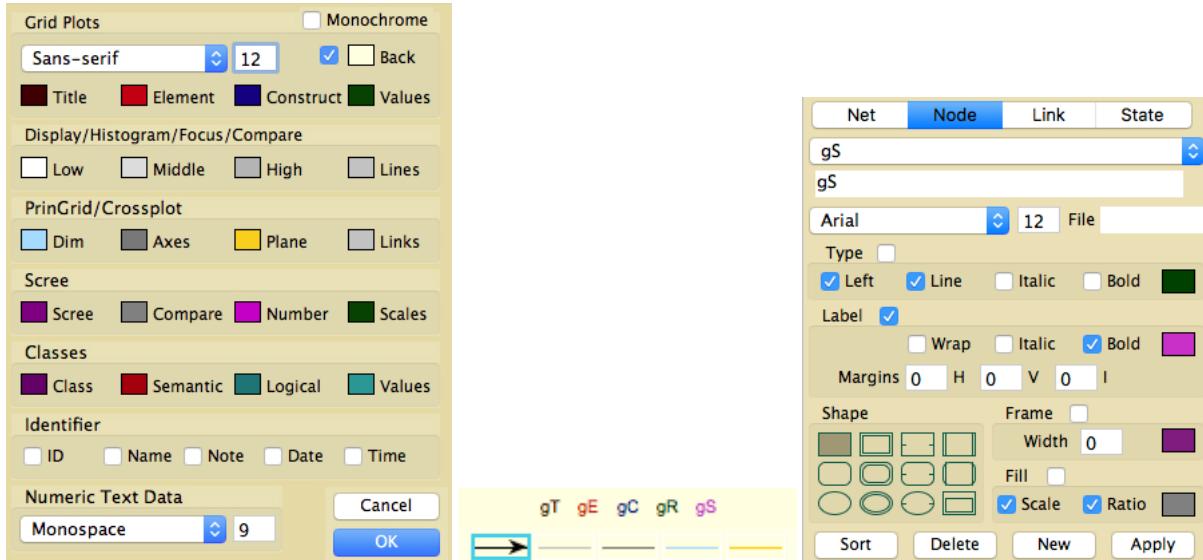


Figure 57: RepGrid styles (left), RepNet equivalent styles (centre), Scree style gS (right)

The first four node styles, gT (title), gE (element), gC (construct) and gR (rating) are always present, and the fifth, gS (scree) is specific to Scree plots. The arrow link type is present by default, and the four line link types are specific to PrinGrid and CrossPlot plots.

Figure 57 right shows the RepNet style dialogue for the gS node style, and it can be seen that the four colours specified for the Scree plot in the RepGrid dialogue have been encoded in the gS node style: *Scree* as frame colour; *Compare* as fill colour; *Number* as label colour; *Scales* as type colour. No node in this style is created but the colours are used in the scree plot any may be changed in RepNet by editing the gS node type.

Style	Type	Label	Frame	Fill
gT		Title	Axes	Back
gE	Voronoi	Element (text)	Element (tree, dot)	Lines (element)
gC	Dim	Construct (text)	Construct (tree, cross)	Lines (construct)
gR	Low	Values	High	Middle
gS	Scales	Number	Scree	Compare

Table 1: Translation of RepGrid style colours to RepNet node style colours

Note that the element and construct colours specified in RepGrid are entered twice in the RepNet styles, once for the text and once for the trees (in Focus) or the dots and crosses (in Crossplot and PrinGrid). This enables the text and tree/dot/cross colours to be changed independently if desired. Similarly the connecting lines are entered twice, once for the element tree and once for the construct tree.

The font and font size specified in RepGrid are entered in every node and can be varied independently. However, this is usually inappropriate for the element and construct labels since any change in font, size or style changes the font metric and would result in an untidy layout. If a different font specification is required it needs to be entered in RepGrid and the analysis layout recomputed. Changing the font data for title can always be accommodated and may also be effective for the rating values.

Figure 58 shows a Focus plot where the element and construct tree and line colours have been changed, as have the ratings style and background shadings, and the title font and colour.

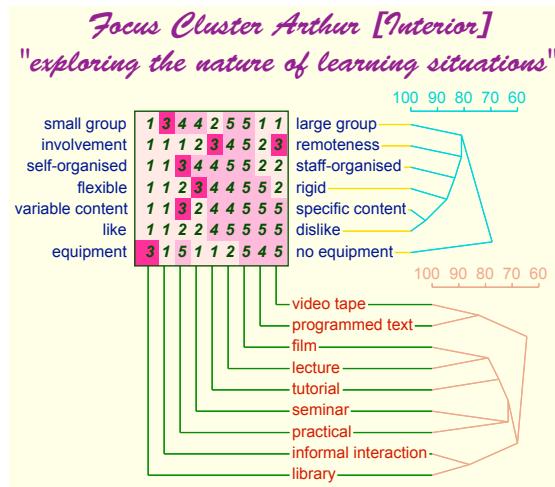


Figure 58: A Focus plot restyled in RepNet

In general, editing RepGrid plots in RepNet is only useful if they have been combined or annotated for a presentation and some slight tweaking is required that can be done more rapidly in RepNet than by running an analysis again.

## 7.2 Histogram plot styles

The *Synopsis* tool in RepGrid and the *Histo* tool in RepGrids both generate histogram plots. Figure 59 shows examples where *Select All* has been used to select all the nodes so that the node structure is apparent. The plot on the left has 10 nodes in total, the title and three groups of 3 for the histograms, That on the left has 15 nodes, the title and two groups of 7 for the histograms.

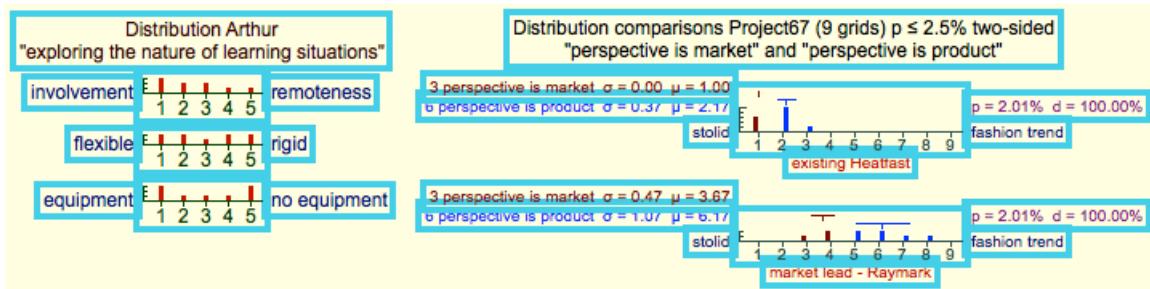


Figure 59: Node structure of histogram plots

Figure 60 top left shows one of the histogram plots ungrouped and the histogram graphic extracted as a node. The node has the null style but, as shown on the right, the style can be changed and a label added if required.

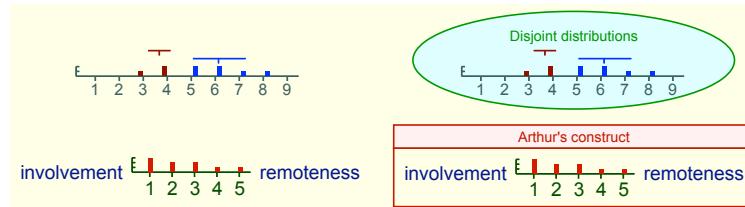


Figure 60: Styling and enclosing histogram graphics

Similarly, the nodes in the histogram in the histogram at the bottom left can be enclosed in an enclosure node with a label showing a comment on it.

The colour scheme of the histogram plots is defined by an associated collection of node types. Figure 61 left shows the five node types common to all RepGrid plots, and right those common to all RepGrids distribution comparisons.



Figure 61: Node types for RepGrid histograms (left) and RepGrids histograms (right)

Whilst the histogram graphic plot is a unitary entity that cannot be edited in RepNet, its colour scheme is represented by the styles of the node types. The histogram bars in the RepGrid synopses represent element counts and are shown in the gE element label colour, and the scales and ratings are shown in the gR rating label colour. The histogram bars in RepGrids distribution comparison plots represent distribution counts and are shown in the Distribution and Distribution+ label colours.

### 7.3 Scree plot styles

The node structure of the scree plots generated in RepGrid are shown in Figure 62). The title and component estimate numbers are normal textual nodes but the scree graph itself is a unitary entity and only its colours may be edited in RepNet.

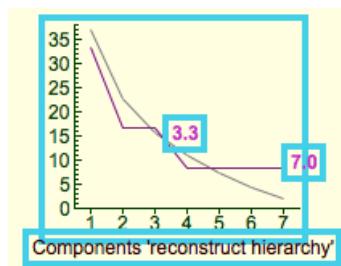


Figure 62: Scree plot node structure

As usual, the title is a node of type gT. The four colours of the *scales*, *scree* graph, *comparison* graph and estimated *number* of components are stored in the gS node type as specified in Figure 57 and Table 1, and can be changed in RepNet by editing the style of this type.

### 7.4 Display and Focus plot styles

The node structure of the Display and Focus plots generated in RepGrid are shown in Figure 63). The plots are unitary entities and only their colours, and possibly the ratings font data, may be edited in RepNet as exemplified in Figure 58.

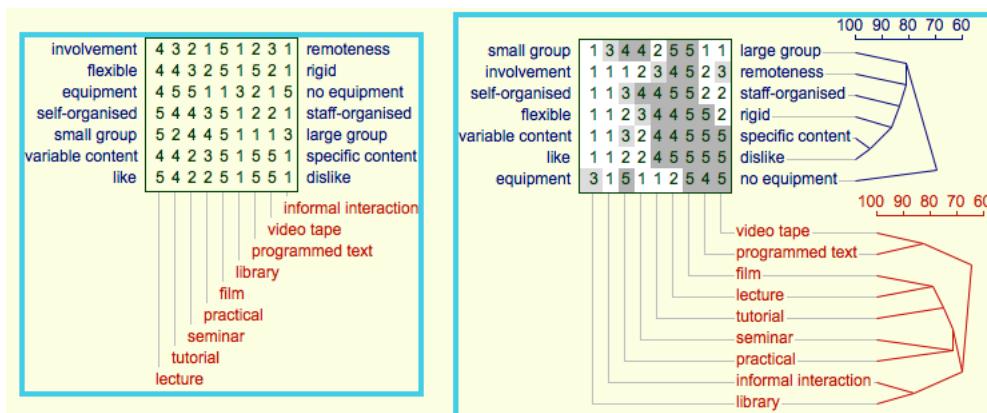


Figure 63: Display and Focus plot nodes

## 7.5 Crossplot and PrinGrid plot styles

The node structure of the Crossplot and PrinGrid plots generated in RepGrid are shown in Figure 64). The plots are node/link structures and may be freely edited as such.

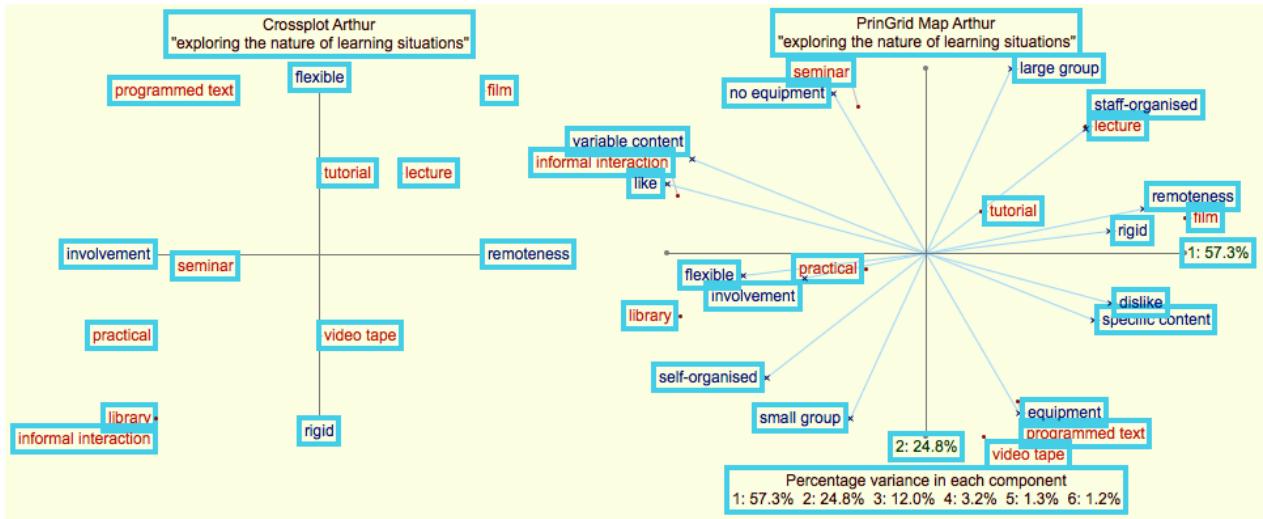


Figure 64: Crossplot and and PrinGrid plot nodes

Three types of *special* node styles are used in the plot and used as anchors for various lines such as the dimensions, axes and links from displaced element and construct labels and their coordinates. They are a dot •, a cross ✕, and an invisible node ■ that only appears when selected, and are used to indicate the element, construct and axis coordinates, respectively. The colour of the dot is that of the gE frame, and of the cross that of the gC frame (see Table 1).

The RepNet editor makes no provision for entering these special nodes but those generated in RepGrid plot may be copied and pasted to another net. Their types may also be changed and they will be coloured in the frame colour of the new type. They do have labels, nodes and states but these have no effect on their appearance.

Four link types are also generated (Figure 57 centre) that provide styles for the line between: an element or construct label and its coordinate (gL); the axis end points (gA); the dimension end points (gD); and the lines delimiting the X-Y plane in 3D plots (gP). The styles of these may all be changed.

## 7.6 Socionets plot styles, link weights and and interactive features

The RepGrids socionets plots (Figure 65) have extensions to the node and link type to support link weights and the interactive control of the threshold and display of the weight threshold above which links are displayed. The weights cannot be entered or edited in RepNet, and the weighted links are in pairs which makes link editing somewhat confusing. The node types for the link label and the cut off must be first and second, respectively, as play a special role in the interaction.

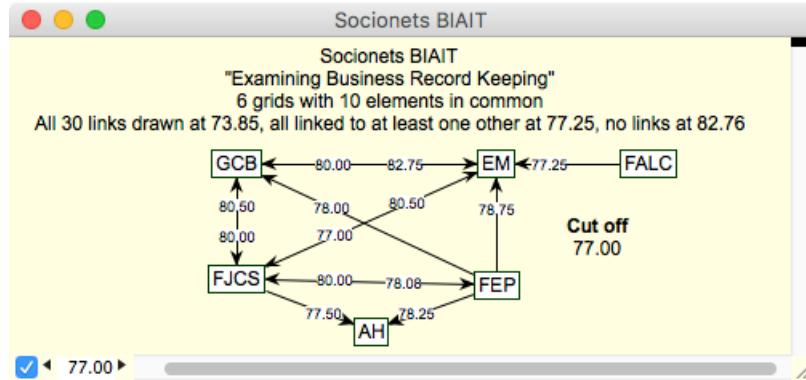


Figure 65: Socionets interactive net with link weights

However, in other respects the socionets nets behave as a normal net. Nodes and links may be added and deleted, and node states may be changed, for example for annotation and highlighting. Node and link styles may also be modified. The type of the cut off node is normally displayed and can be changed, for example to *Threshold*, without affecting its dynamic updating. It can also be deleted if not required.

## 7.7 Techniques for creating graphic structures in RepNet

Whilst RepNet is not a general-purpose graphics creation and editing tool, the node/link structures it supports can be used to create a wide range of structures than one might expect. This is particularly important in programming graphic plots in scripts where there are often requirements that go beyond those of the conventional *concept map* style of plot.

For example, invisible nodes may be used as anchors for line segments enabling structures such as line histograms and graphs requiring curved lines to be plotted. This might be tedious to do manually but can be simply achieved in RepScript. For example,

For example, Figure 66 shows a logical net structure representing a genus-differentia structure on the left together a Leibniz line diagram providing intensional and extensional semantics for the net on the right. The line diagram was generated automatically by a program in RepScript that is part of the CNet logical representation and inference tool.

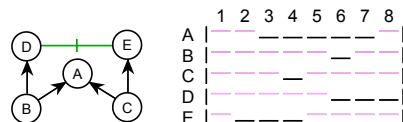


Figure 66: Leibniz line diagram for a logical net generated in RepScript by CNet

# 8 Scripting RepNet

All of the data in a net is accessible to RepScript as are the capabilities to create nets and to interact with them through the mouse or keyboard. Hence it is possible to extend the graphic functionality of Rep Plus through scripts issued with Rep Plus, generated by users, or shared through the user community.

RepScript is a dynamically compiled, object oriented modern programming language extended to have access to the data structures of RepGrid, RepGrids, RepServe and RepNet. The language and its application programming interface are detailed in the RepScript manual. The following sections exemplify various ways in which RepScript can be used to create modify and interact with nets in RepNet.

## 8.1 Running a script and interacting with it

The bottom section of the popup menu that appears when the mouse is clicked not over a node (§5.2) provides options to run the last script previously run (or set up by the application that created the net) or to select and run any available script (Figure 8.1). In addition the last script may also be run by double-clicking in the net outside the nodes.

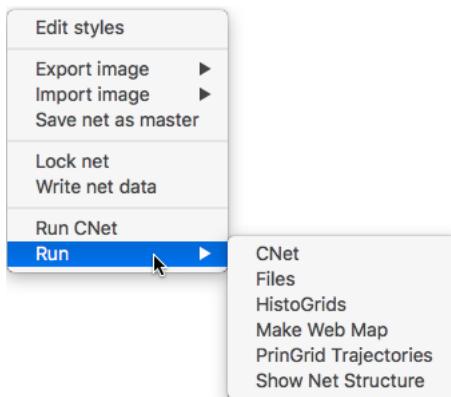


Figure 67: Running a script from the popup menu when the mouse is clicked outside the nodes

If the script locks the net then any mouse or keyboard actions in the net are sent as messages to the script rather than to the net editing processes. This enables interaction with the net to be programmed through the script. For example, node states may be changed, popup menus may be generated to offer the user context-sensitive options, or the *Escape* key may be used to terminate the script.

### 8.1.1 Messages sent to the script of an interactive (locked) net

Messages are sent to the script by running it with an argument available through the function *Script-State* that indicates the type of action and further arguments in the vector with an empty name, accessible as vGet(0), vGet(1) and so on. The node number of the node under the cursor is normally

sent in the first argument and the second argument may be "Double" for a double-click or the key depressed for a keyboard action. Further tool-specific arguments may be supplied by Rep Plus tools that generates nets that can interact with the tool such as the RepGrids *Histo* tool.

The action arguments are:

**Run** Sent when the script is initially run to run the script and allow interactive scripts to lock the net;

**Click** Sent when the mouse is clicked in a locked net;

**Key** Sent when a key is depressed whilst a locked net is in the front window;

**Cursor** Sent repeatedly to a locked net to allow the cursor to be set according to the context.

## 8.2 Files: opening files from a net

*Files* is a RepNet script for using the link in the note field of a node created from a file that has been dragged to a net to open that file when the node is clicked. When the script is run the net is locked and as the mouse is moved over the net it is shown as a star outside the nodes and a button when it is over a node.

Clicking when the cursor is a button opens the file referenced in the note field of the node. Double-clicking when it a star, or keying *Escape*, halts the script.

### 8.2.1 *Files script*

```
// Last updated 15-Dec-2017

// Use URL stored in note field to open a file

// Main Program

dim node As Variant=vGetI(0)
dim arg As String=vGet(1)

select case scriptState
  case "Cursor"
    if node=-1 then
      vSet(12,0) // Star -- outside nodes
    else
      vSet(10,0) // Button -- inside a node
    end if
  case "Key"
    if arg=ESC then nSet("Lock","false") // escape key terminates script
  case "Click"
    if node=-1 then
      if arg="Double" then nSet("Lock","false") // double-click outside nodes
    else
      dim s As String=nGet("Node",node,"")
      s=hGet("Note")
      if s.Instr("//")>0 then Call wOpenURL(s)
    end if
  case "Run" // net run -- must lock
    nSet("Lock","true") // lock the net
end select
```

## 8.3 PrinComp: creating a plot from an analysis of a grid

*PrinComp* is a RepGrid script that computes a principal components analysis of a grid and outputs it as text and a graphic plot in RepNet. It replicates some of the functionality of the built-in *PrinGrid* tool and is intended to illustrate how grid data can be analyzed and plotted through scripts, for example, to extend RepGrid with multi-dimensional scaling tools. Figure 68 shows the output from *PrinComp* and §8.3.1 lists the script. This illustrates how a net window may be created and populated with nodes and links graphing the principal components analysis.

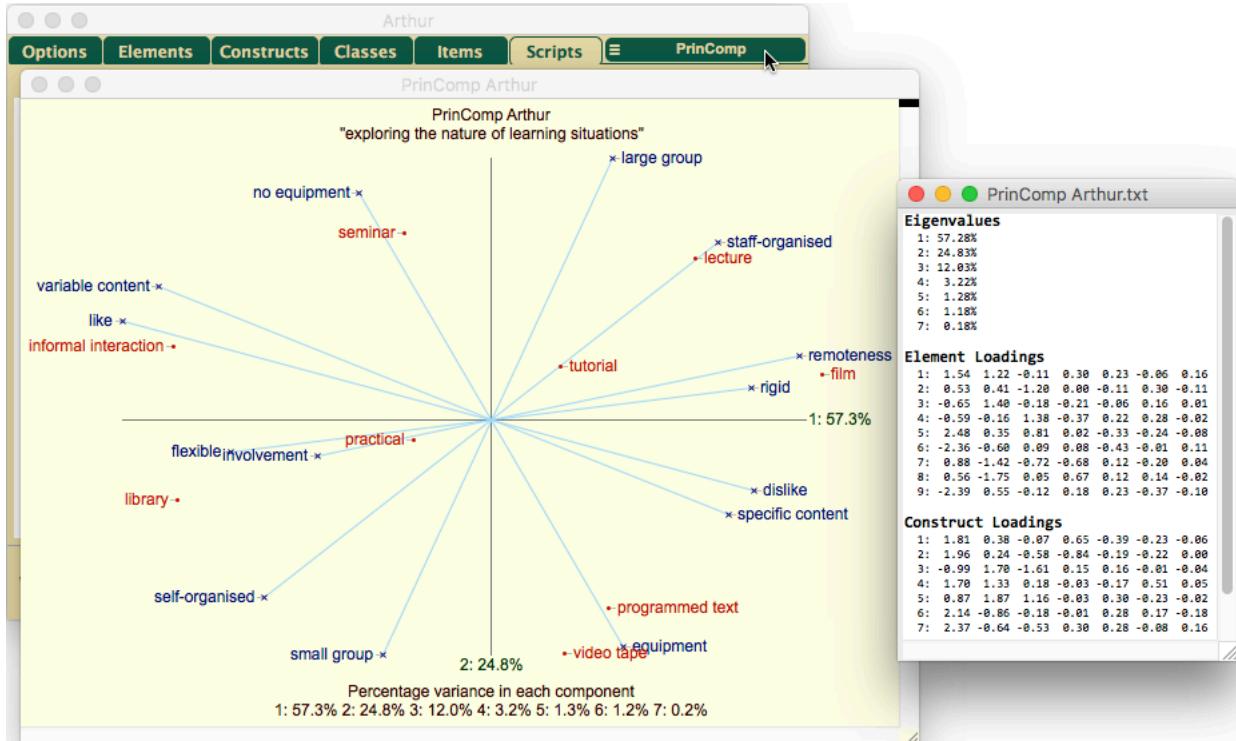


Figure 68: *PrinComp* script run from a grid in RepGrid

### 8.3.1 *PrinComp* script

```
// Principal components analysis of a grid -- output plot and text forms

const NodeSpecNone=4
const NodeSpecCross=8
const NodeSpecDot=12

sub WriteMatrix(M,fmt) As Double, fmt As String)
    const fmti="###"
    dim rb As Integer=UBound(M,1), cb As Integer=UBound(M,2), i,j As Integer
    for i=0 to cb
    next
    for i=0 to rb
        Write(sFormat(i+1,fmti)+":")
        for j=0 to cb
            Write(" "+sFormat(M(i,j),fmt))
        next
    next
end sub
```

```

        next
        Writeln
    next
end sub

sub WriteVector(V() As Double, fmt As String)
    const fmti="###"
    dim bnd As Integer=V.UBound, i As Integer
    for i=0 to bnd
        Writeln(sFormat(i+1,fmti)+": "+sFormat(V(i),fmt)+"%")
    next
end sub

Sub TextOutput()
    wOpenWindow("PrinComp "+gGet("Name")+".txt")
    wSetFont("monospace",9,"",&c000000)
    write("Eigenvalues"+EOL,12,"B")
    WriteVector(V,"-0.00")
    Writeln
    Write("Element Loadings"+EOL,12,"B")
    WriteMatrix(LE,"-0.00")
    Writeln
    Write("Construct Loadings"+EOL,12,"B")
    WriteMatrix(LC,"-0.00")
    Writeln
    wFlushln
    wSetChanged(false)
End Sub

Sub GraphOutput(hAxis As Integer, vaxis As Integer, scale As Integer, hrev As Boolean, vrev As Boolean, mean As Boolean, num As Boolean)
    nNew(6)
    SetupCoordinates(hAxis,vAxis,scale,hrev,vrev,mean)
    PlotAxes(hAxis,vAxis)
    PlotConstructs(num)
    PlotElements(num)
    PlotTitle
    PlotVariance
    Call nGet("Window","PrinComp "+gGet("Identifier"))
End Sub

Sub PlotVariance()
    dim s,t() As String, i As Integer, bnd As Integer=V.UBound
    for i=0 to bnd
        t.Append str(i+1)+": "+Format(V(i),"0.0")+"%"
    next
    s="Percentage variance in each component"+CR+Join(t)
    hEmpty
    hSet(0,"Left")
    hSet(vmax,"Top")
    hSet("Ratings","Type")
    hSet(s,"Label")
    hSet("1 0 0 20","Place")
    Call nGet("NodeNew","");
End Sub

Sub PlotTitle()
    dim s As String="PrinComp "+gGet("Identifier")+CR+++++gGet("Context")+++++
    hEmpty
    hSet(0,"Left")
    hSet(vmin,"Top")
    hSet("gT","Type")
    hSet(s,"Label")
    hSet("1 2 0 10","Place")
    Call nGet("NodeNew","");

```

```

End Sub

Function SetupQuadrants(h As Integer, v As Integer) As String
    dim n As String=nGet("NodeNew","")
    if h<0 then
        if v<0 then
            TL.Append n
        else
            BL.Append n
        end if
    else
        if v<0 then
            TR.Append n
        else
            BR.Append n
        end if
    end if
    return n
End Function

Sub PlotElements(num As Boolean)
    dim j As Integer, s,n1,n2 As String, left As Boolean
    for j=0 to bnde
        hEmpty
        left=E(j,0)<0
        hSet("gE","Type")
        hSet(NodeSpecDot,"Code")
        hSet(E(j,0),"Left")
        hSet(E(j,1),"Top")
        n1=nGet("NodeNew","");
        s=gGet("E",str(j))
        if num then
            if left then s=s+" "+str(j+1) else s=str(j+1)+" "+s
        end if
        hSet(s,"Label")
        hSet(0,"Code")
        if left then hSet("2 1 5","Place") else hSet("0 1 5","Place")
        n2=SetupQuadrants(E(j,0),E(j,1))
        nSet("Line",n1,n2,"gL")
    next
End Sub

Sub PlotConstructs(num As Boolean)
    dim i As Integer, s,lhp,rhp,n1,n2,n3,n4 As String, left As Boolean
    for i=0 to bdnc
        left=L(i,0)<0
        s=gGet("C",str(i),"")
        lhp=hGet("L")
        rhp=hGet("R")
        hEmpty
        hSet("gC","Type")
        hSet(NodeSpecCross,"Code")
        hSet(L(i,0),"Left")
        hSet(L(i,1),"Top")
        n1=nGet("NodeNew","");
        s=lhp
        if num then
            if left then s=s+" "+str(i+1) else s=str(i+1)+" "+s
        end if
        hSet(s,"Label")
        hSet(0,"Code")
        if left then hSet("2 1 5","Place") else hSet("0 1 5","Place")
        n2=SetupQuadrants(L(i,0),L(i,1))
        nSet("Line",n1,n2,"gL")
    next
End Sub

```

```

hEmpty
left=R(i,0)<0
s=gGet("C",str(i))
hSet("gC","Type")
hSet(NodeSpecCross,"Code")
hSet(R(i,0),"Left")
hSet(R(i,1),"Top")
n3=nGet("NodeNew","");
s=rhp
if num then
  if left then s=s+" "+str(i+1) else s=str(i+1)+" "+s
end if
hSet(s,"Label")
hSet(0,"Code")
if left then hSet("2 1 5","Place") else hSet("0 1 5","Place")
n4=SetupQuadrants(R(i,0),R(i,1))
nSet("Line",n3,n4,"gL")
nSet("Line",n1,n3,"gD")
next
End Sub

Sub PlotAxes(hAxis As Integer, vaxis As Integer)
  dim i,j As Integer, n1,n2 As String

  hEmpty
  hSet("gR","Type")
  hSet(NodeSpecNone,"Code")
  hSet(hmin,"Left")
  hSet(0,"Top")
  n1=nGet("NodeNew","");
  hSet(hmax,"Left")
  n2=nGet("NodeNew","");
  nSet("Line",n1,n2,"gA")

  hSet("0","Left")
  hSet(vmin,"Top")
  n1=nGet("NodeNew","");
  hSet(vmax,"Top")
  n2=nGet("NodeNew","");
  nSet("Line",n1,n2,"gA")

  hSet(0,"Code")
  hSet(hmax,"Left")
  hSet(0,"Top")
  hSet(Str(hAxis+1)+": "+sFormat(V(hAxis),"0.0")+"%","Label")
  hSet("0 1","Place")
  n1=nGet("NodeNew","")

  hSet("0","Left")
  hSet(vmax,"Top")
  hSet(Str(vAxis+1)+": "+sFormat(V(vAxis),"0.0")+"%","Label")
  hSet("1 0","Place")
  n1=nGet("NodeNew","")

End Sub

Sub SetupCoordinates(hAxis As Integer, vaxis As Integer, scale As Double, hrev As Boolean, vrev
  As Boolean, mean As Boolean)
  dim i,j As Integer, m,xmult,ymult,h,v As Double
  if hrev then xmult=-scale else xmult=scale
  if vrev then ymult=scale else ymult=-scale

  redim L(bndc,1)
  redim R(bndc,1)
  for i=0 to bndc

```

```

if mean then m=M(i)
h=LC(i,hAxis)*xmult
v=LC(i,vAxis)*ymult
L(i,0)=(-1.0-m)*h
L(i,1)=(-1.0-m)*v
R(i,0)=(1.0-m)*h
R(i,1)=(1.0-m)*v
hmin=Min(hmin,Min(L(i,0),R(i,0)))
hmax=Max(hmax,Max(L(i,0),R(i,0)))
vmin=Min(vmin,Min(L(i,1),R(i,1)))
vmax=Max(vmax,Max(L(i,1),R(i,1)))
next

redim E(bnde,1)
for j=0 to bnde
    E(j,0)=LE(j,hAxis)*xmult
    E(j,1)=LE(j,vAxis)*ymult
    hmin=Min(hmin,h)
    hmax=Max(hmax,h)
    vmin=Min(vmin,v)
    vmax=Max(vmax,v)
next
End Sub

Sub Main(hAxis As Integer, vaxis As Integer, scale As Integer, hrev As Boolean, vrev As Boolean,
mean As Boolean, num As Boolean)
// this demonstrates how to get data from grid and compute eigenvectors
// G=gGetA2D this function could be used to extract the normalized rating data from the grid
// for other analyses
V=gGetEigen(false,false,1.0,M,LE,LC) // for principal components the built-in function may be
used
GraphOutput(hAxis,vAxis,scale,hrev,vrev,mean,num)
TextOutput
End Sub

// Globals
dim ne As Integer=gGetI("NE")
dim bnde As Integer=ne-1
dim nc As Integer=gGetI("NC")
dim bndc As Integer=nc-1
dim TL(),TR(),BL(),BR() As String
dim E(-1,-1),L(-1,-1),R(-1,-1),M(-1),V(-1),LE(-1,-1),LC(-1,-1),hmin,hmax,vmin,vmax As Double

// horiz comp, vert comp, scale, h rev, v rev, mean, num
Main(0,1,100,false,false,true,false)

```

#### 8.4 PrinGrid Trajectories: annotating an analysis of a composite grid

*PrinGrid Trajectories* is a RepNet script for annotating a PrinGrid analysis of a composite grid from grids with both common elements and constructs to show trajectories of change. Figure 69 shows the output from PrinGrid Trajectories and §8.4.1 lists the script. This illustrates how data in a net may be accessed and modified in a way that may be undone.

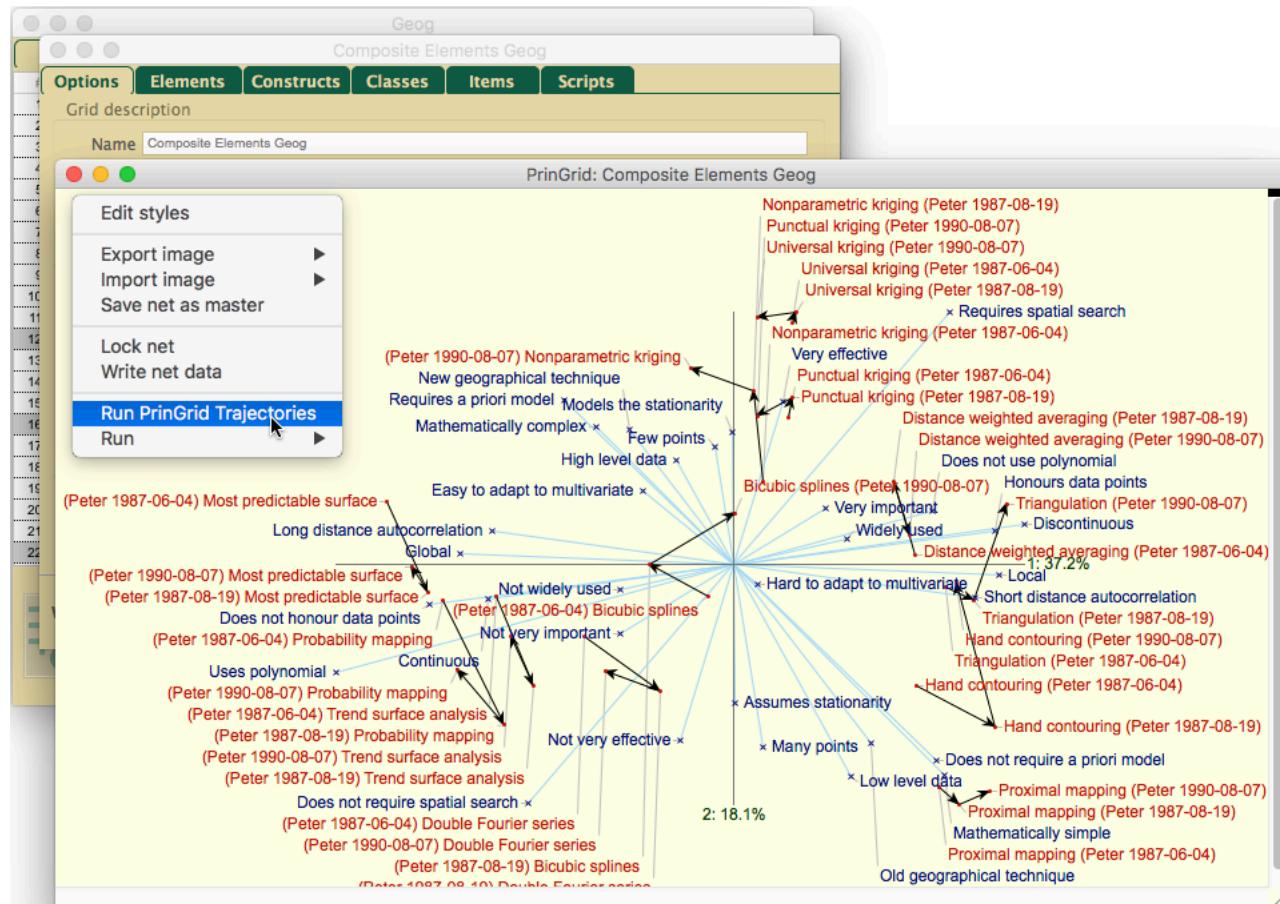


Figure 69: *PrinGrid Trajectories* script run on a PrinGrid plot of a composite grid in RepNet

#### 8.4.1 PrinGrid Trajectories script

```

// Uses the annotation of the location nodes (dots and cross) in a composite grid
// net to plot trajectories of change in a PrinGrid analysis of a composite of
// grids with both common E and common C

// Last changed 6-Feb-2016

const etyp=3073 // MapSpecDot+1
const ctyp=2050 // MapSpecCross+2

dim i, j, bnd, typ(), en(), cn() As Integer, s, nam(), e(), c() As String

```

```

typ()=nGetAI("Types")
nam()=nGetA("Nodes")

bnd=typ.UBound
for i=0 to bnd
  if typ(i)=etyp then
    e.append nam(i)
    en.append i
  elseif typ(i)=ctyp then
    c.append nam(i)
    cn.append i
  end if
next

nSet("UndoSave","PrintGrid Trajectories")

bnd=en.UBound
for i=0 to bnd-1
  s=e(i)
  j=e.IndexOf(s,i+1)
  if j<>-1 then nSet("Line",str(en(i)),str(en(j)),"Arrow")
next

bnd=cn.UBound
for i=0 to bnd-1
  s=c(i)
  j=c.IndexOf(s,i+1)
  if j<>-1 then nSet("Line",str(cn(i)),str(cn(j)),"Arrow")
next

Redraw

```

## 8.5 HistoGrids: interacting through a net with the RepGrids Histo tool

*HistoGrids* is a RepNet script for interacting with histogram plots from the RepGrids Histo tool. Figure 69 illustrates the interaction supported by *HistoGrids* and §8.5.1 lists the script. It illustrates how RepScript can use messages from RepNet to provide the user with access to the RepGrids data from which the plot was generated.

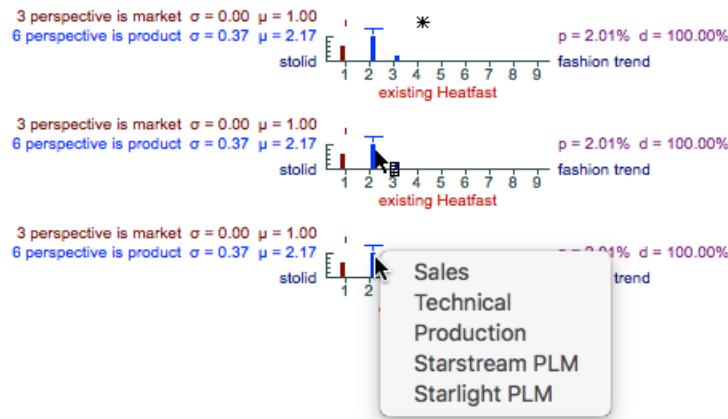


Figure 70: *HistoGrids* script run on a histogram plot from the RepGrids Histo tool

The script is called whenever the mouse is moved or clicked and received a message that indicates which part of which histogram the mouse is over. The script changes the cursor from a star to

a pointer with a popup menu symbol when the mouse is over a histogram bar. If the mouse is then clicked the script generates a popup menu showing the grids that contributed to the count indicated by that bar. If a grid is selected from the menu the script opens it in RepGrid.

### 8.5.1 *HistoGrids script*

```
// Last updated 2-Jul-2017

// Interaction with RepGrids Histograms

// Main Program

// message - node number TAB label TAB note TAB Flag ("" "G" or "GC") TAB C TAB E TAB T TAB B
dim message As string=vGet(0)
dim doub As String=vGet(1)
dim node As integer=sGetI(message,1)

select case scriptState
  case "Cursor"
    if node=-1 then
      vSet(12,0) // CStarPM -- outside nodes
    else
      select case sGet(message,4)
        case "GC"
          vSet(1,0) // CNormalPM
        else
          vSet(12,0) // CStarPM
        end select
      end if
    case "Click"
      if node=-1 then
        DoClickOutside
      else
        select case sGet(message,4)
          case "GC"
            DoHistoClick(sGetI(message,5),sGetI(message,6),sGetI(message,7),sGetI(message,8))
          else
            DoClickOutside
          end select
        end if
      case "Run" // net run -- must lock
        nSet("Lock","true") // lock the net
        case "Locked" // user has locked net
      end select
    sub DoClickOutside()
      if doub="true" then nSet("Lock","false")
    end sub

    sub DoHistoOpen(fid As String)
      'print fid
    end sub

    sub DoHistoClick(C As Variant, E As Variant, T As Variant, B As Variant)
      dim s As String=Menu(Join(iGetA("GridsMenu",C,E,T,B),""))
      'dim s As String=Menu("Open#+"+Join(iGetA("GridsMenu",C,E,T,B),"#"))
      if s<>"" then iGetX("GridOpen",s)
    end sub
  end select
```

## 8.6 CNet: interacting with a logical inference engine programmed in scripts

**CNet** is a suite of RepNet scripts that implements inference and semantic modelling for first order logical systems. It includes a translator for semantic networks and was originally developed to serve as a principled inference engine for semantic network representations of expert systems (Rappaport and Gaines, 1990; Gaines, 1991, 2009). Since logical axioms may also be represented graphically as semantic nets (Gaines, 1992, 1993) it also became used to study the foundations of, and relations between, different logical systems (Gaines, 2015). Since Kelly (1955) presents the foundations of personal construct psychology in axiomatic form (Shaw and Gaines, 1992b,a, 1993) it can be used to study the processes of anticipation within Kelly's framework (Shaw and Gaines, 2005; Gaines and Shaw, 2012).

CNet is a complex set of scripts that serves to illustrate how RepScript may be used to add new applications to Rep Plus that integrate with its existing functionality and also create major extensions of that functionality. It is a complex application with its own manual.

## 9 Bibliography

- Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (2003). *The Description Logic Handbook*. Cambridge University Press, Cambridge.
- Bird, A. (2007). Inference to the only explanation. *Philosophy and Phenomenological Research*, 74(2):424–432.
- Blundell, A. J. (1982). *Bond Graphs for Modelling Engineering Systems*. Ellis Horwood, Chichester, UK.
- Bryson, J. M., Ackermann, F., Eden, C., and Finn, C. B. (2004). *Visible Thinking: Unlocking Causal Mapping for Practical Business Results*. Wiley, Chichester, UK.
- Gaines, B. (1991). An interactive visual language for term subsumption visual languages. In *IJCAI'91: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 817–823. Morgan Kaufmann, San Mateo, CA.
- Gaines, B. (1993). A class library implementation of a principled open architecture knowledge representation server with plug-in data types. In *IJCAI'93: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 504–509. Morgan Kaufmann, San Mateo, CA.
- Gaines, B. R. (1992). Managing tractability in an open-architecture knowledge representation server. In *Proceedings of the AAAI Workshop on Tractable Reasoning*, pages 51–56. AAAI, Palo Alto, California.
- Gaines, B. R. (2003). Organizational knowledge acquisition. In Holsapple, C., editor, *Handbook on Knowledge Management: 1*, pages 317–347. Springer, Berlin.
- Gaines, B. R. (2009). Designing visual languages for description logics. *Journal of Logic, Language and Information*, 18(2):217–250.
- Gaines, B. R. (2015). Universal logic as a science of patterns. In Koslow, A. and Buchsbaum, A., editors, *The Road to Universal Logic: Festschrift for 50th Birthday of Jean-Yves Béziau*, pages 145–189. Birkhäuser, Basel.
- Gaines, B. R. and Shaw, M. L. G. (1994). Using knowledge acquisition and representation tools to support scientific communities. In *AAAI'94: Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 707–714. AAAI Press/MIT Press, Menlo Park, California.
- Gaines, B. R. and Shaw, M. L. G. (1995a). Collaboration through concept maps. In Schnase, J. and Cunnias, E., editors, *Proceedings of CSCL95: Computer Support for Collaborative Learning*, pages 135–138. Lawrence Erlbaum, Mahwah, New Jersey.
- Gaines, B. R. and Shaw, M. L. G. (1995b). Concept maps as hypermedia components. *International Journal Human-Computer Studies*, 43(3):323–361.

- Gaines, B. R. and Shaw, M. L. G. (2012). Computer aided constructivism. In Caputi, P., Viney, L. L., Walker, B. M., and Crittenden, N., editors, *Constructivist Methods*, pages 183–222. Wiley, New York.
- Gaut, B. (2000). “Art” as a cluster concept. In Carroll, N., editor, *Theories of Art Today*, pages 25–44. University of Wisconsin Press, Madison, WI.
- Gaut, B. (2005). The cluster account of art defended. *British Journal Aesthetics*, 45(3):273–288.
- Kelly, G. A. (1955). *The Psychology of Personal Constructs*. Norton, New York.
- Lehmann, F. and Rodin, E. Y. (1992). *Semantic Networks in Artificial Intelligence*. Pergamon Press, New York.
- Lipton, P. (2004). *Inference to the Best Explanation*. International library of philosophy. London.
- Mintzes, J. J., Wandersee, J. H., and Novak, J. D. (1998). *Teaching Science for Understanding: A Human Constructivist View*. Academic Press, San Diego.
- Mintzes, J. J., Wandersee, J. H., and Novak, J. D. (2000). *Assessing Science Understanding: A Human Constructivist View*. Academic Press, San Diego.
- Novak, J. D. (1998). *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum, Mahwah.
- Novak, J. D. and Gowin, D. B. (1984). *Learning How To Learn*. Cambridge University Press, New York.
- Oliver, R. M. and Smith, J. Q. (1990). *Influence Diagrams, Belief Nets, and Decision Analysis*. Wiley, Chichester, UK.
- Pask, G. (1976). *Conversation Theory*. Elsevier, Amsterdam.
- Rappaport, A. T. and Gaines, B. R. (1990). Integrated knowledge base building environments. *Knowledge Acquisition*, 2(1):51–71.
- Reisig, W. (1985). *Petri Nets: An Introduction*. Springer, Berlin.
- Shaw, M. and Gaines, B. (1992a). On the relation between the repertory grid and term subsumption knowledge structures: theory, practice and tools. *Research and Development in Expert Systems IX. Proceedings of British Computer Society Expert Systems Conference*, pages 125–143.
- Shaw, M. and Gaines, B. (1993). Personal construct psychology foundations for knowledge acquisition and representation. In Aussénac, N., Boy, G., Gaines, B., Linster, M., Ganascia, J., and Kodratoff, Y., editors, *Proceedings of EKAW-93: Seventh European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 256–276.

- Shaw, M. L. G. and Gaines, B. (1992b). Kelly's 'Geometry of psychological space' and its significance for psychological modeling. *New Psychologist*, pages 23–31.
- Shaw, M. L. G. and Gaines, B. (1998). A research-based masters program in the workplace. *Proceedings of WCCCE'98: Western Canadian Conference on Computing Education*.
- Shaw, M. L. G. and Gaines, B. R. (2005). Expertise and expert systems: emulating psychological processes. In Fransella, F., editor, *The Essential Practitioner's Handbook of Personal Construct Psychology*, pages 87–94. Wiley, Chichester, UK.
- Sowa, J. F. (1991). *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. Morgan-Kaufman, San Mateo, California.
- Toulmin, S. (1958). *The Uses of Argument*. Cambridge University Press, Cambridge, UK.

**Most of our publications cited in the references are freely available on the web**