MakeGraph

By: D. Austin Henderson, Jr.

Supported by: Nick Briggs (Briggs.pa@Xerox.com)

INTRODUCTION

MakeGraph is a module which sits on top of Grapher and helps one create graphs depicting a data structure by walking through it. The central idea is that each point in the walk (and node in the graph) is characterized by a datum/state pair and motion is defined by a graph specification in the form of state transition function. This function is specified by a collection of state specifications, each of which indicates how to display (label and font) the datum when one is in that state and how to find the datum/state pairs which are the sons of that node. Also the state specification may specify additional roots for the walk. The generation of a branch of the graph ceases when either there are no sons of a node, or an already encountered node is revisited (identical datum and identical state). The module contains a function for creating such graphs and an example of its use: a function which graphs the graph specifications themselves. Comments are welcomed

FUNCTIONS

The main functions are:

(MAKE.GRAPH *WINDOW TITLE GRAPH.SPECIFICATION ROOTS CONTEXT LEFTBUTTONFN MIDDLEBUTTONFN TOPJUSTIFYFLG DEPTH*) [Function]

Creates a MAKEGRAPH window. If *WINDOW* is NIL, then a new one will be created and the user will be prompted to position it. Otherwise, the graph will be shown in *WINDOW*. The window will be titled with *TITLE*, will call *LEFTBUTTONFN* and *MIDDLEBUTTONFN* on nodes selected (or NIL if selection is made where no node is positioned), and will be justified as indicated by *TOPJUSTIFYFLG* (a la Grapher). The button functions are defaulted to MAKE.GRAPH.LEFTBUTTONFN (which scrolls the window so that the selected node is in the middle of the window, or if the left shift key is depressed, prints out information about it) and MAKE.GRAPH.MIDDLEBUTTONFN (which provides a menu of two choices: INSPECT - inspects the datum of the node selected, or if the left shift key is depressed, inspects the node itself; and SUB.GRAPH - which opens another MAKEGRAPH window with the same parameters as this one, but with graph starting at the selected node). The arguments to MAKE.GRAPH are added as properties to the window under their argument names. Selecting in the title invokes the functions which are the values of the window properties TITLE.LEFTBUTTONFN and TITLE.MIDDLEBUTTONFN (not in the calling sequence; set by the user if desired; called with a single argument - *WINDOW*; defaulted to a function which provides a menu of UPDATE and SHOW.GRAPH.SPEC (see functionality below)). The graph is created according to the *GRAPH.SPECIFICATION* (see below) to depth *DEPTH*, starting from *ROOTS* which are (DATUM . STATE) pairs. *CONTEXT* is an extra argument which is a passed along to all accessing expressions.

GRAPH.SPECIFICATION [Parameter]

A GRAPH.SPECIFICATION is a property list of STATE.SPECIFICATIONs where the properties are the state names.

STATE.SPECIFICAITON [Property list]

A STATE.SPECIFICATION is a property list whose properties and values are as follows (in this, EXPR means a LISP form which will be evaluated in an environment in which DATUM is bound to the node's datum, STATE to the node's state, and CONTEXT to context):

LABEL [Property]

an expression returning something which will be printed as the label of the node; if no LABEL property is provided, the string "???" will be used.

FONT [Property]

an expression returning the font to be used for this node; if no FONT property is provided, the default font for the grapher will be used.

SONS [Property]

a form indicating a list of (DATUM . STATE) pairs to be used in generating the sons of this node; the acceptable forms are any of the following:

(data-expression state-expression) [Property value]

where data-expression returns a list of datum's for the son nodes, and state-expression is evaluated in the context of each of these in turn to produce the corresponding state of each.

(LIST (data-expression state-expression) ...) [Property value]

a template of expressions which are evaluated individually to produce a list of sons of the same form, viz. (DATUM . STATE) pairs.

(EVAL expression) [Property value]

the expression returns a list of (DATUM . STATE) pairs of the sons.

(UNION sons-spec ...) [Property value]

where each sons-spec is any of these forms (recursively).

(TRACE sons-spec) [Property value]

a device for helping debug graph specifications; the value is the value of sons-specs; the user is given the chance to INSPECT them after they have been generated.

ROOTS [Property]

like SONS, except the resulting (DATUM . STATE) pairs are used as possibly additional roots of the graph.

(MAKE.GRAPH.CONSTRUCT *GRAPH.SPECIFICATION INITIAL.ROOTS CONTEXT DEPTH*)[Function]

This is the functional heart of MAKE.GRAPH broken out for those who wish to handle their own interactions with grapher and the window package. It produces a list of graphnodes with labels and sons as specified by *GRAPH.SPECIFICATION* (see MAKEGRAPH), starting from *INITIAL.ROOTS* which are (DATUM . STATE) pairs. *CONTEXT* is an extra argument which is a passed along to all accessing expressions. Returns the list of graphnodes.

(MAKE.GRAPH.FIND.ROOTS *GRAPH.SPECIFICATION INITIAL.ROOTS CONTEXT DEPTH*)[Function]

Finds the real roots from a set of initial roots, using the same processing as MAKEGRAPH uses. This is helpful when you want to hand a "correct" set of roots of a structure to MAKEGRAPH without having to explore the dependencies within that structure. As with MAKEGRAPH, the data structure is processed according to the *GRAPH.SPECIFICATION* (see MAKEGRAPH), starting from *INITIAL.ROOTS* which are (DATUM . STATE) pairs. *CONTEXT* is an extra argument which is a passed along to all accessing expressions. Returns the real roots as a list of (DATUM . STATE) pairs.

Supporting Functions

(MAKE.GRAPH.UPDATE.WINDOW *WINDOW*) [Function]

Uses the window properties (which may have been changed) to reinvoke MAKE.GRAPH on the window.

(MAKE.GRAPH.SHOW.SPEC *GRAPH.SPECIFICATION*) [Function]

Uses MAKE.GRAPH to produced a graph of a *GRAPH.SPECIFICATION*. It uses as the graph.specification for this layout the value of the variable MAKE.GRAPH.SPEC.SPEC which presents *GRAPH.SPECIFICATION* (reflectively) as a graph.specification. (MAKE.GRAPH.SPEC.SPEC can serve as a template for other graph.specifications. It is a fairly complex 9-state specification. For a simpler example see below under MAKE.GRAPH.SHOW.LIST.)

(MAKE.GRAPH.EXAMPLE.1) [Function]

Calls MAKE.GRAPH.SHOW.SPEC on MAKE.GRAPH.SPEC.SPEC.

(MAKE.GRAPH.SHOW.LIST *OBJECT*) [Function]

Uses MAKE.GRAPH to produced a graph of an arbitrary Lisp object. It uses as the graph.specification for this layout the value of the variable MAKE.GRAPH.LIST.SPEC which presents *OBJECT* as a tree whose nodes are LISTPs and whose leaves are non-LISTPs.

MAKE.GRAPH.LIST.SPEC [Variable]

MAKE.GRAPH.LIST.SPEC is the simple 1-state specification below, included here as an example of a graph.specification

```
(OBJECT (  DOC (ANY LISP OBJECT)              - some documentation
           LABEL (COND           ((LISTP DATUM) "( )")
                      (T DATUM))
           SONS ((COND           ((LISTP DATUM) DATUM)
                      (T NIL))
                 (QUOTE OBJECT))
        )
 )
```

For a more complex example see above under MAKE.GRAPH.SHOW.SPEC.

(MAKE.GRAPH.EXAMPLE.2) [Function]

Calls MAKE.GRAPH.SHOW.LIST on MAKE.GRAPH.LIST.SPEC; that is, produces a graph of this simple graph.specification as a list. Notice that selecting the title command UPDATE in this window will yield a different graph of the same structure, viz. as a GRAPH.SPECIFICATION.

Other useful functions

(MAKE.GRAPH.DATUM *NODE*) [Function]

Returns the DATUM associated with the graph node *NODE*.

(MAKE.GRAPH.STATE *NODE*) [Function]

Returns the STATE associated with the graph node *NODE*.

(MAKE.GRAPH.FATHER *NODE*) [Function]

Returns the graph node which is the father of the graph node *NODE*.