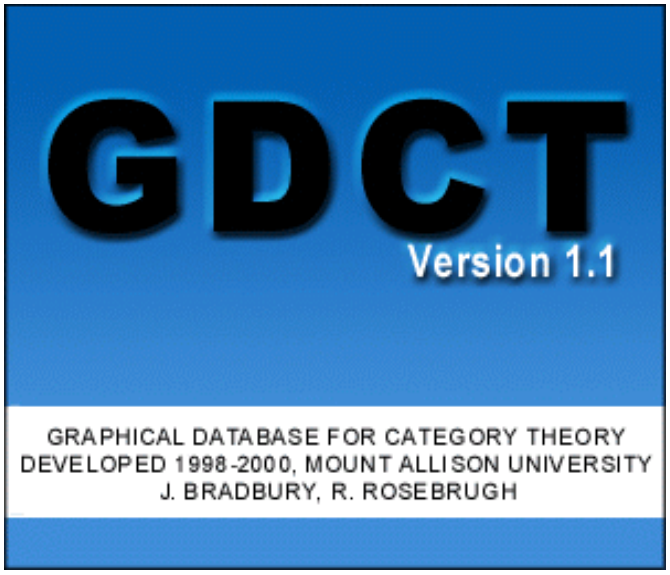


USER GUIDE for GRAPHICAL DATABASE
for CATEGORY THEORY

Written By: Jeremy Bradbury
Edited By: Robert Rosebrugh

June 20, 2000

TABLE OF CONTENTS

Additional Information about GDCT
can be found at
http://mathcs.mta.ca/research/rosebrugh/gdct/

USER GUIDE FOR GDCT

SECTION 1: INTRODUCTION

About GDCT

Graphical Database for Category Theory was developed in the summer of 1999 under an NSERC summer research grant. Some of the algorithms used in GDCT were originally developed in A Database of Categories, a C program written by Ryan Gunther and Michael Fleming under the supervision of Dr. Robert Rosebrugh, and Category Theory Database Tools, written by Jeremy Bradbury under the supervision of Dr. R. Rosebrugh.

This application allows for the creation, editing, and storage of finitely presented categories. Categories can be opened and saved from local files as well as loaded from a specified server. Once a category file is in memory it can be printed or tools for testing properties of objects and arrows can be applied to it. The tools available in this version of GDCT are Make Confluent, Equality of Composites, Make Dual, Initial Object, Sum, Terminal Object, and Product.

This program also allows the display of categories through the use of graph classes that were developed at Auburn University for Visualizing Graphs with Java (VGJ), a tool for graph drawing and graph layout. The original program was heavily modified for displaying categories. It can be found at http://www.eng.auburn.edu/department/cse/research/graphdrawing/.

Functors between finitely presented categories can also be created and stored. Functors in memory display their domain and codomain categories, and their action is shown by an animated display.

Some of the algorithms used in GDCT were originally developed in A Database of Categories, a C program written by Ryan Gunther and Michael Fleming with supervision by Robert Rosebrugh. This is available at http://mathcs.mta.ca/research/rosebrugh/dbc/. The GDCT application is also base on Category Theory Database Tools, a Java applet written by Jeremy Bradbury with supervision by Robert Rosebrugh. This program is available at http://mathcs.mta.ca/research/rosebrugh/ctdt/.

Development Team

Jeremy Bradbury

B.Sc. Honours in Computer Science & Mathematics, Mount Allison University

e-mail address: jsbrdbr@mta.ca

webpage address: http://cs.mta.ca/~jsbrdbr/

Dr. Robert Rosebrugh

Professor, Dept. Mathematics and Computer Science, Mount Allison University

e-mail address: rrosebrugh@mta.ca

webpage address: http://www.mta.ca/~rrosebrugh/

Installation

The most current version of Graphical Database for Category is available for download at http://mathcs.mta.ca/research/rosebrugh/gdct/download.htm. There are four installation options. The first two options are available only to Windows 95/98/NT/2000 users. Options three and four are available to users of all platforms.

Option One: InstallShield Setup [Windows Only]

Download the InstallShield setup program that will automatically install GDCT for you. This option is best for user who will require help with initial setup. This setup includes a Java Runtime Environment for Windows 95/98/NT/2000 only. The InstallShield version of the GDCT Setup is 3.42 MB in size.

Option Two: Non-InstallShield Setup [Windows Only]

Download the non-InstallShield version of the GDCT setup program. For this option the user should unzip the GDCT.zip file to the desire directory and then run the application from there. This setup includes a Java Runtime Environment for Windows 95/98/NT/2000 only. The non-InstallShield version of the GDCT setup program is 2.56 MB. For users who are concerned with download size this is the best version.

Option Three: Download Compiled Java Source Files and Java Runtime Environment

Under this option, the complete Java source files are already compiled into "class" files and stored in a java archive ("jar") file. The jar file can be downloaded as a *.zip file as a *.tar.gz file, or in uncompressed form. This download also includes all necessary help files as well as sample category and functor files. In order to interpret the compiled code, in the jar file, a Java Run-time Environment must be downloaded. JREs are used by an operating system to interpret Java class files so that they can be run on the local machine.

Option Four: Download Java Source Files and Java Developer's Kit

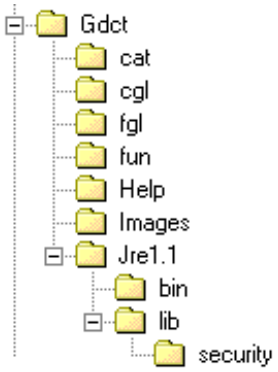
The complete Java source files are available for download and may be modified under the terms of the GNU General Public License, Version 2. This download also includes all necessary help files as well as sample category and functor files. It is available as a *zip file or as a *tar.gz file. In order to compile and interpret Java source files, the Java Developers Kit must be downloaded. JDKs are used by an operating system to compile Java source files and interpret Java class files so that they can be run on the local machine.

Directory Structure

Upon installation, the Graphical Database for Category Theory, unless changed by the user, will be installed in a directory called GDCT. Inside this directory are the subdirectories cat, cgl, fgl, fun, Help, Images, and Jre1.1. The following files are also located inside the main directory :

- GDCT.bat — This is the batch file that the user selects to interpret the java archive file (jar).
- GDCT.jar — This is the java archive file that is interpreted by the Java Runtime Environment. This file contains all of the compiled Java source files.
- GDCT.ico — This is an icon file that is used on any shortcuts to the GDCT application.
- GDCT.ini — This initialization file contains information regarding internal settings, category graphical settings, functor graphical settings, functor animation settings, server settings, as well as a list of the most recently opened files.

The subdirectories cat and cgl contain sample category files. The subdirectories fun and fgl contain sample functor files.



The subdirectory Help contains all of the help files while the subdirectory Images contains all of the images used by the Java application.

The Jre1.1 subdirectory contains the Java Runtime Environment Version 1.1 which is called by GDCT.bat to interpret GDCT.jar.

WARNING: The user should not modify any files in the subdirectories Help or Images.

SECTION 2: CATEGORIES

Introduction

A finitely presented category can be considered a “directed graph with relations between paths of its edges”.

Our notation for category data is:

Objects:
A, B, C, D, . . .

Maps/Arrows:
e:A →A, f:A →B, g:B →C, h:A →C. . .

Identity Maps/Arrows:
1:A →A, ... (one for each object)

Composition of Maps/Arrows defined by Relations:
e = 1, g*f = h, ... (path of arrows)

File Types, Categories

In GDCT, there are three different file types associated with categories:

***.cat**

These files are a text only representation of a category. These files can be both opened and saved. Since a graphical representation of the category is not saved one is generated randomly upon opening the *.cat file.

***.cgl**

These files contain both a text and graphical representation of a category and can be both opened and saved. They are a concatenation of a .cat file and a .gml file.

***.gml**

These files contain only a graphical representation of a category. These files can only be saved (and may be used by the VGJ applet), they can not be opened.

CAT File Format

The text representation of categories is stored in the following format. Any line that is a comment must start with the "#" character. On the line following the word "category" is the category name. Objects are listed between the words "objects" and "arrows". Arrows are listed between the words "arrows" and "relations". Relations are listed after the word "relations".

NOTE: The words "category", objects", "arrows", "relations", and "gml" are key words and cannot be used as the name of the category or as the name of an object or arrow within the category.

Objects and arrows can be a string of any length and can contain any characters excluding ":", "-", ">", "*", "#", ".", "=", ",", " ".

Relations are composed using the composition symbol "*".

An example of a typical CAT file can be found in Appendix A.

CGL File Format

A CGL file contains the information for both a text representation and a graphical representation.

The text representation is the same as the file format for CAT files. Any line that is a comment must start with the "#" character. On the line following the word "category" is the category name. Objects are listed between the words "objects" and "arrows". Arrows are listed between the words "arrows" and "relations". Relations are listed after the word "relations".

NOTE: The words "category", objects", "arrows", "relations", and "gml" are key words and cannot be used as the name of the category or as the name of an object or arrow within the category.

Objects and arrows can be a string of any length and can contain any characters excluding ":", "-", ">", "*", "#", ".", "=", ",", " ".

Relations are composed using the composition symbol "*".

The graphical representation uses a format known as gml which was developed at Auburn University for use with the Visualizing Graphs with Java (VGJ) project. It is incorporated into the cgl format by placing it at the end of the file after the keyword "gml". For more information consult the VGJ Documentation.

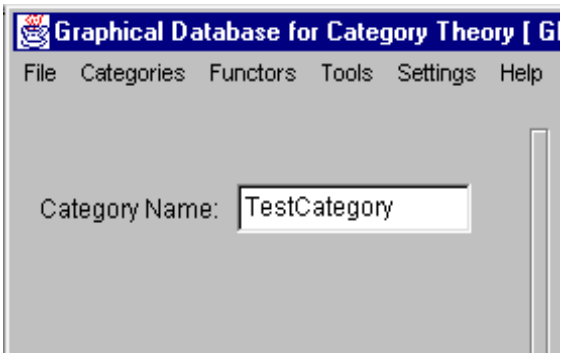
An example of a typical CGL file can be found in Appendix A.

Creating Categories

To create a Category in memory the user selects Categories | Create Category.

The names of objects and arrows can be strings of any length that is not a reserved word and can contain any characters excluding ":", "-", ">", "*", "#", ".", "=", ",", " ".

Enter a (descriptive!) name in the "Category name" box and type Enter.



Type the names of objects in the "Object Name" box which now appears, completing each with Enter. Press the "Objects Complete" button when done.

During creation of the category the objects and arrows are displayed in the Category Display Window. Objects may be moved about by mouse action at any time. Click on an object and drag it to a new position.

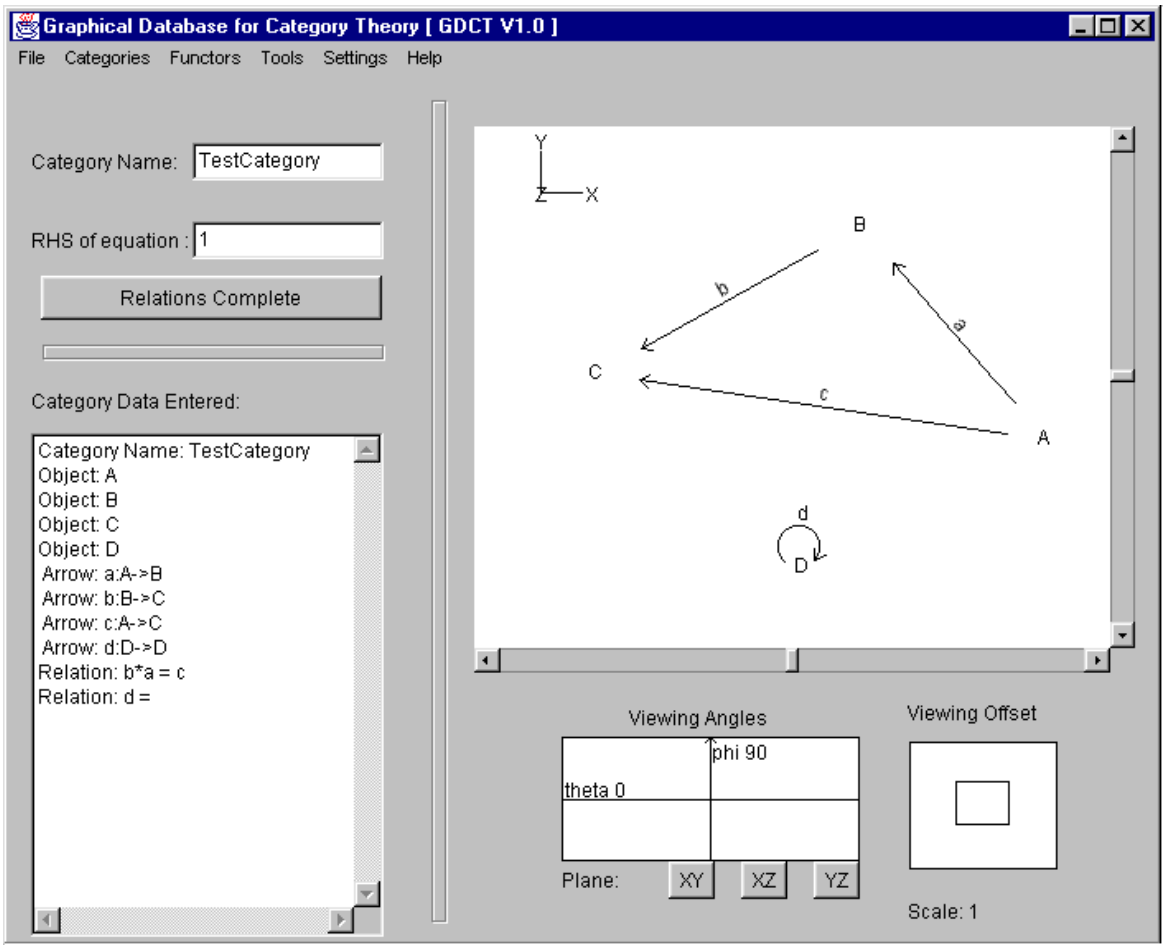
WARNING: No two objects may have the same name. If a duplicate object name is entered the program will display an error message and the new object will not be included in the category.

Entering arrows is very similar to entering objects. Type the names of arrows in the "Arrow Name" box which now appears, completing each with Enter. After the name of an arrow is entered, the user is prompted for the domain and then the codomain of the arrow. Press the "Arrows Complete" button when done.

As arrows are entered they are displayed in the Category Display window.

WARNINGS: No two arrows may have the same name. The domain and codomain of each arrow must be a declared object.
An arrow cannot be named "1" since this symbol reserved by the program for the identity arrow.

Next the user enters the equations among composites of the generating arrows. The "LHS of equation" box appears and the user must enter a composable path of arrows (separated by the composition symbol "*"). In the "RHS of equation" box another composable path is entered.
After the equations have been entered press the "Equations complete" button.



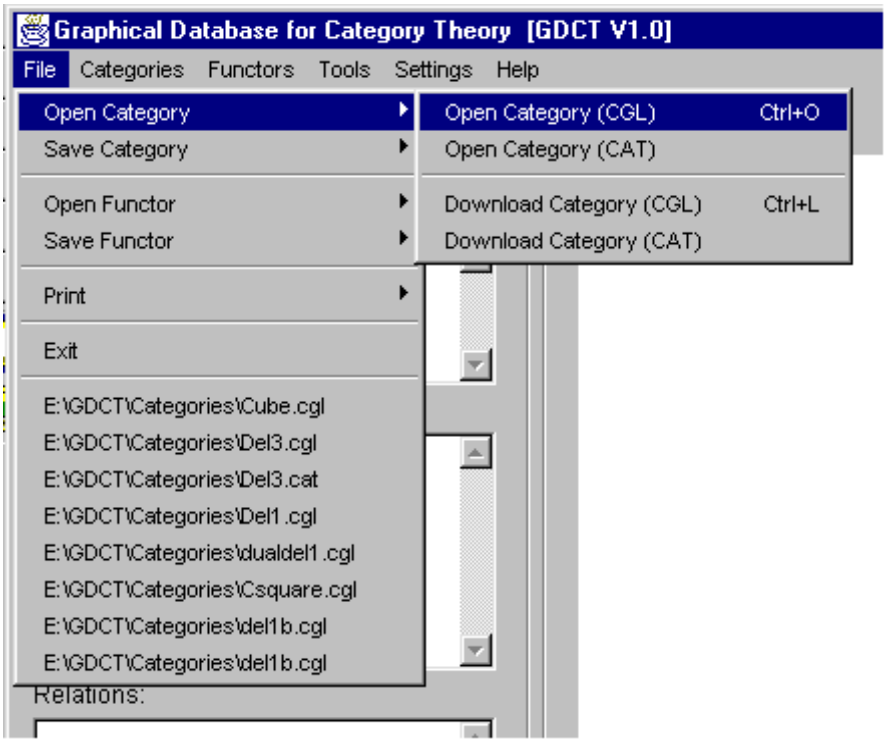
WARNING: Only valid equations are accepted. An equation is invalid if it contains arrows that are not in the category, or an undefined composition, or if the domain and codomain of the left side do not equal the domain and codomain of the right side.

This completes the creation of the new category.

WARNING: The newly created category is stored in memory. To save the category to disk select File | Save Category and select graphical (.cgl) or text (.cat) format. A save dialogue window opens.

Opening Categories

In order to open a text category file (.CAT) select File | Open Category | Open Category(CAT) or Categories | Open Category | Open Category(CAT). This opens a file selection window. Opening graphical category files (.CGL) is similar. An alternative method to open a .cgl category file is to press CTRL+O. To open a category simply double click on the category's name or click once on the category name and press the OK button.



If there is any error in the format of the selected file an error message is displayed and the open is aborted.

The most recent category files can also be opened by clicking on one of their names in the File menu.

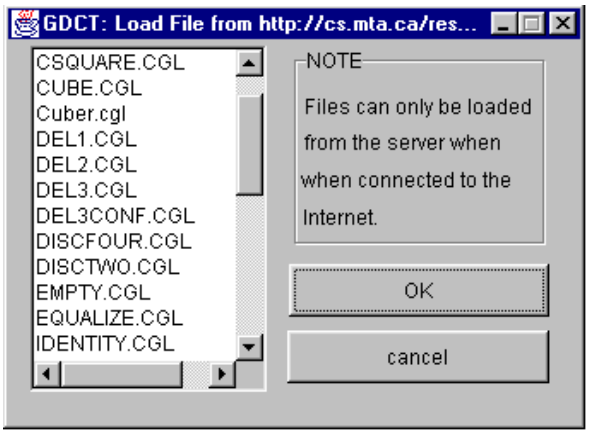
Downloading Categories

In order to download a a text category file (.cat) select File | Open Category | Download Category(CAT) or Categories | Open Category | Download Category(CAT).

Downloading a graphical category file (.cgl) is similar, select File | Open Category | Download Category(CGL) or Categories | Open Category | Download Category(CGL). An alternative method to download graphical category files is to press CTRL+D.

Choosing any of the above options will display a list of categories in a selection box in a new frame. The categories listed are located on the server specified in the frame's title bar. To select a category to download

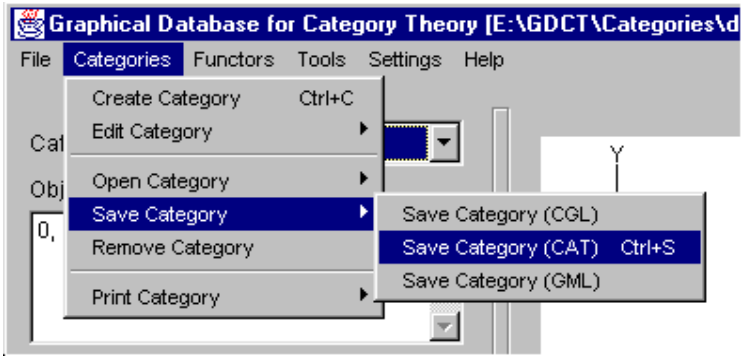
simply double click on the category's name or click once on the category name and press the OK button. Either of these methods will display the downloaded category. If the program cannot locate the files on the server an error message will be displayed. This could result from the user not being connected to the Internet or as a result of a server error.



To change the settings for downloading categories, select Settings | Set Server. This displays a dialog for changing the default server and directories from which categories and functors can be loaded using the appropriate load command.

Saving Categories

In order to save a category file (.cgl) select File | Save Category | Save Category(CGL) or Categories | Save Category | Save Category(CGL). This opens a file selection window. Saving text category files (.cat) or simply the VGJ graph file (.gml) is similar. A shortcut to save a .cgl category file is to press CTRL+S. To save a category to an existing file simply double click on the category file name. A new file is created by typing its name in the dialogue and clicking OK.



Removing Categories

Selecting Categories | Remove Category opens a list of the categories currently in memory. Select from the list and click OK to remove a category.

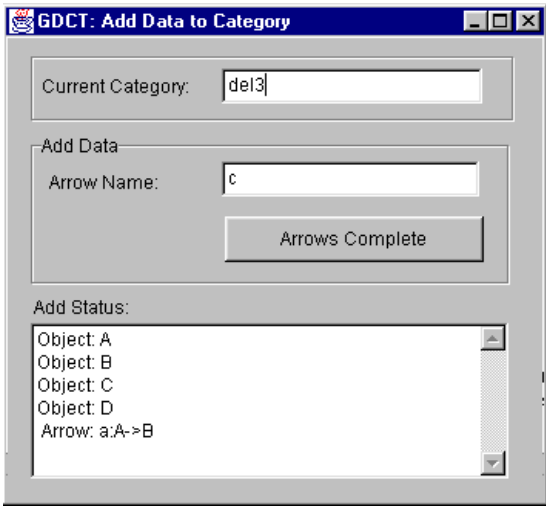
Printing Categories

To print the text version of the currently displayed category, select File | Print | Print Category (Text). To print the graphical version select File | Print | Print Category (Graphical).

To save the graphical version of the currently displayed category to a postscript file, select File | Print | Print Category (Postscript).

Adding Data

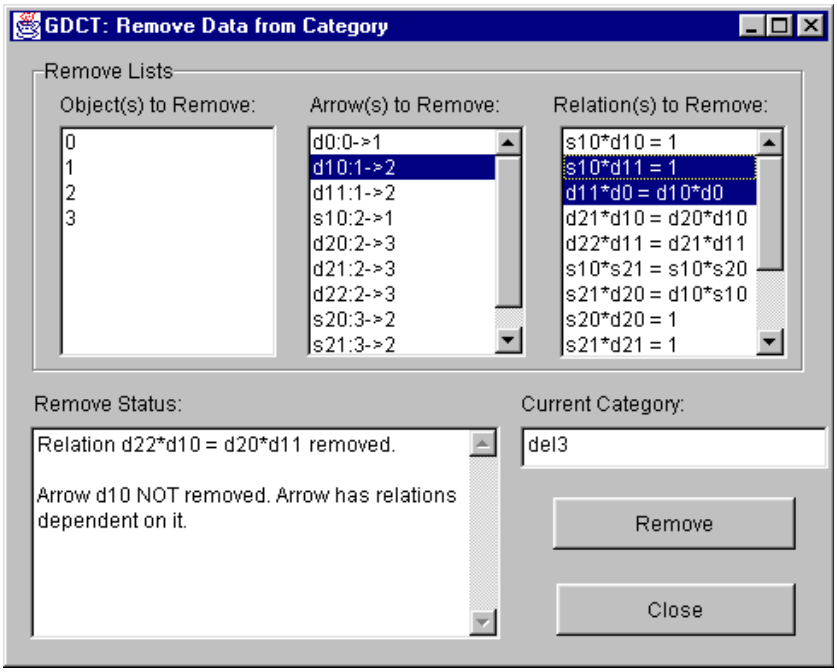
Selecting Categories | Edit Category | Add Data opens a dialogue which allows addition of objects, arrows and relations to the current category. To add objects type their names in the box. When finished click Objects Complete, then arrows may be added: type the name, then the domain and codomain; when finished click Arrows Complete. To add relations the left and right sides must be specified; when finished click Relations Complete.



WARNING: The domain and codomain of a new arrow must already exist, and the sides of a relation must be composable paths. An arrow can be removed only if it does not appear in any current relations. An object can be removed only if it is neither the domain nor codomain of a current arrow. Adding data only affects the version of a category in memory. To save any changes to disk use the Save Category dialogue.

Removing Data

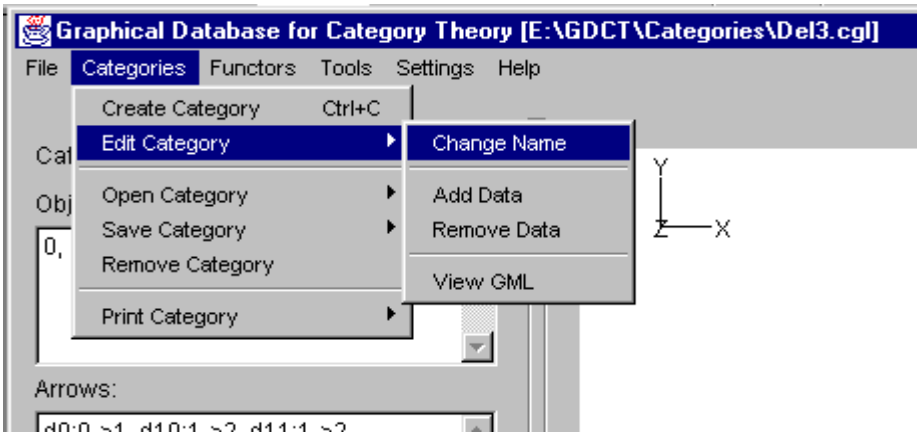
Selecting Categories | Edit Category | Remove data opens a dialogue with lists of the objects, arrows and relations of the current category. Select items from the lists and click Remove to delete the data from the category.



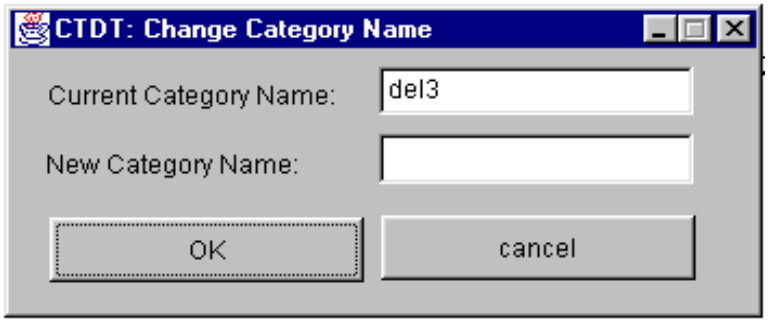
WARNING: An arrow can be removed only if it does not appear in any current relations. An object can be removed only if it is neither the domain nor codomain of a current arrow. Removing data only affects the version of a category in memory. To save any changes to disk use the Save Category dialogue.

Changing a Category Name

To change the name of the currently displayed category, the user selects Categories | Edit Category | Change Name.



Selecting this option will display a dialog in which the user can enter the new name.

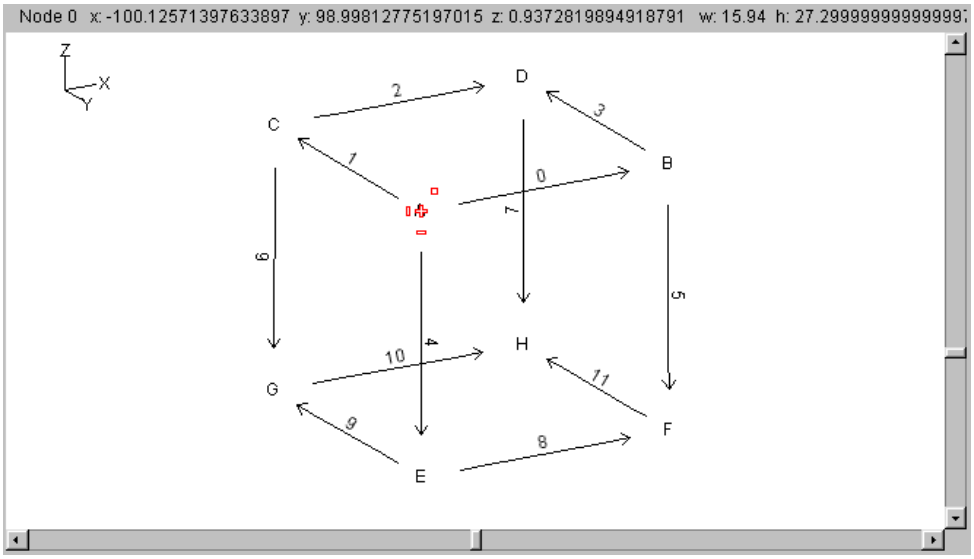


SECTION 3: GRAPHICAL DISPLAY OF CATEGORIES

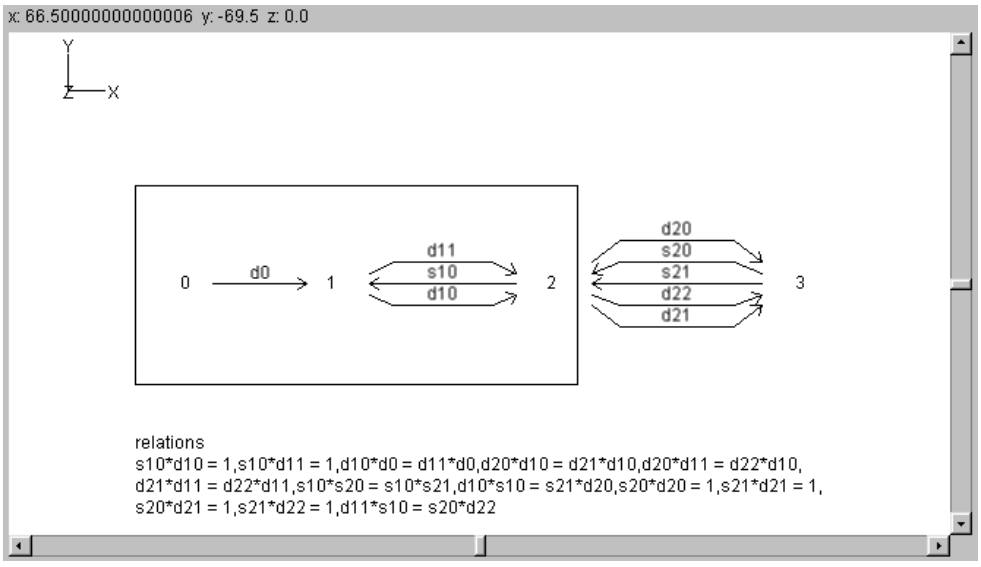
Introduction

The controls for the graphical display of categories are explained in the help files under the title "Visual Display Controls". This help file deals with manipulating the graphical display of a category.

There are two ways to select a object or arrow. The first is to click directly on the object or arrow so that it become highlighted red.



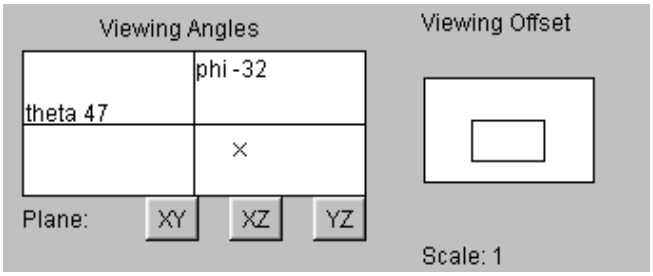
The second way is to click and hold the left mouse button down before dragging a square around the desired object(s) or arrow(s).



Once an object or arrow has been selected, it can be moved by holding down on it and pulling it to another position on the graphical display canvas.

Visual Display Controls

The controls for the graphical display of categories were developed at Auburn University as part of the a 3D package that is used by GDCT in the display of categories. The visual controls can be hidden from view by selecting Settings | Show VGJ Controls. The visual controls available are:



Viewing Angles

The viewing angle can be changed by using the control below the left side of the graphical display. Click and hold the "X" to rotate the view to any 3D angle. The current orientation of the category displayed can be determined by the XYZ axis displayed in the top left corner of the category display area.

Viewing Plane

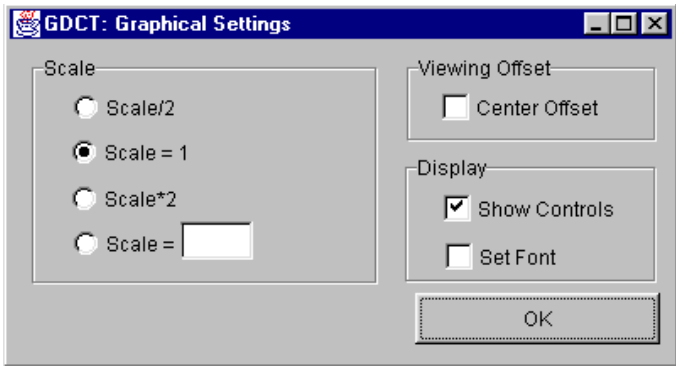
The plane that the category is being viewed in can be changed by pressing one of the three buttons below the "Viewing Angles" control window in the bottom right corner of the main screen. The user can select the XY plane, XZ plane, or the YZ plane.

Viewing Offset

The viewing offset is located in the bottom right hand corner of the main window. It is basically a virtual screen that allows the user to click and hold on the "box" to move the viewable canvas area without using the scroll bars. The viewing offset can be automatically centered by selecting Settings | Category Graphical Settings and clicking the check box by "Center Offset".

Scale

The scale that the category is being viewed at is displayed directly below the "Viewing Offset" controls. The scale can be changed by selecting Settings | Category Graphical Settings. The user can decrease the scale by a factor of two, increase the scale by a factor of two, set the scale to one, or set a user defined scale.



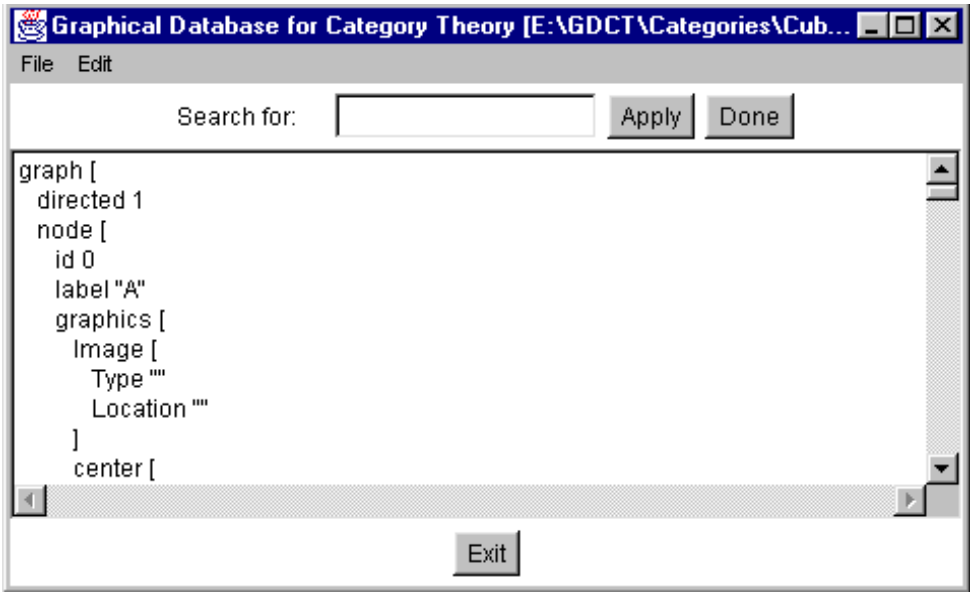
Set Font

The font size and face can be changed by selecting Settings | Category Graphical Settings and clicking the check box in front of "Set Font". This will display a dialog in which the user can type in the desired face and size for the font. Click "Apply" to save these changes or click "Cancel" to keep the current settings.

View GML

GML is the text representation of the graphical display of the current category. To view the GML code select Categories | Edit Category | View GML. Choosing this menu option will display a frame containing the GML code.

Each object is stored as a node. The relations are stored as a single node. The arrows are stored as edges between nodes. Multiple arrows between nodes are stored as one edge with commas separating the names of the individual arrows.

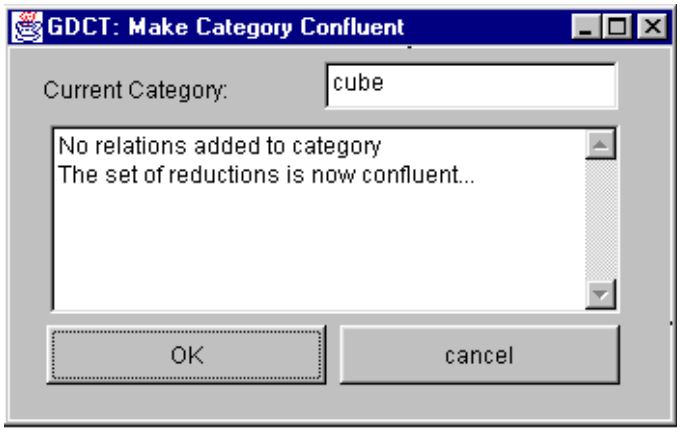


There are several tools to help viewing of the GML:

Edit | GoToTop
Edit | GoToBottom
Edit | GoToLine
Edit | Search

Make Confluent

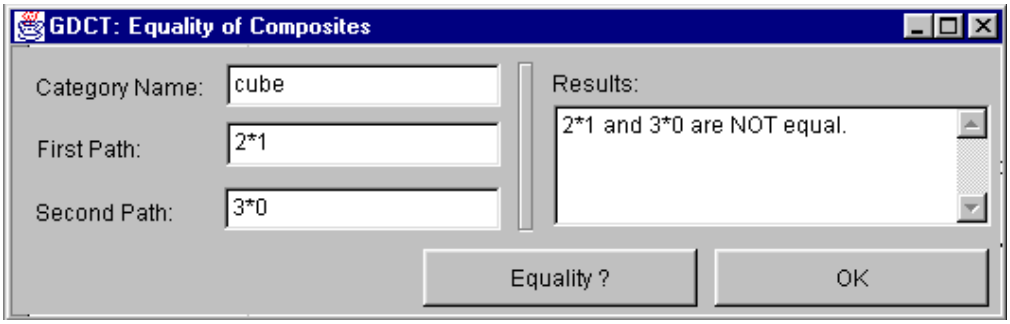
Select Tools | Make Confluent. This command applies the Knuth-Bendix algorithm to the currently displayed category with arrows ordered in their storage order.



Relations which are added to the current category are reported. To add these relations to the current category click "Ok", or click "Cancel" to keep the category in its current from.

Equality of Composites

To test two (possibly composite) arrows for equality in the current category select Tools | Equality of Composites. A dialogue allows entry of two paths.



WARNING: The category must be confluent for valid results. Domains and codomains of paths must match.

Make Dual Category

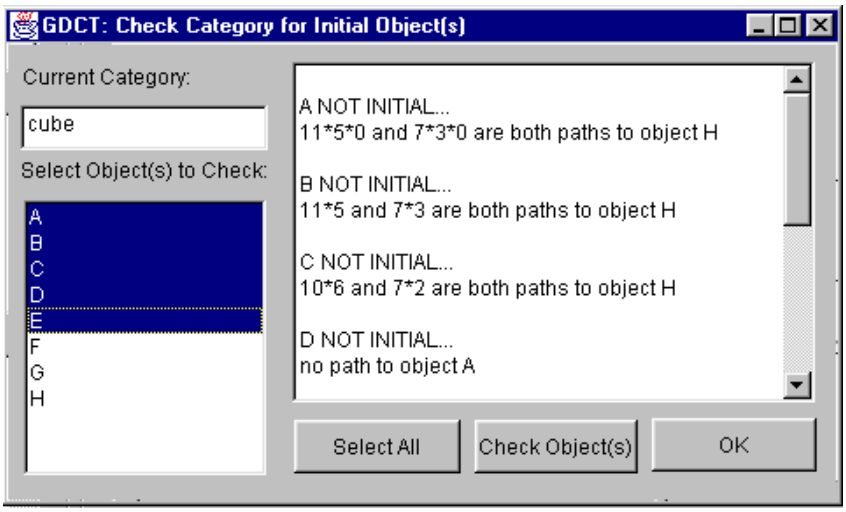
This tool will create the opposite category of the currently displayed category. Select Tools | Make Dual. A dialogue window opens prompting for the name of the dual. After the user provides the name the dual category is displayed.

WARNING: The opposite category is NOT SAVED automatically when it is created; use File | Save

Category if you wish to save the dual.

Initial Object ?

Select Tools | Initial Object?. This command tests objects of the current category (chosen by a dialogue) for initiality. For each selected object which is not initial a reason is given.



WARNING: This command will not operate correctly unless the category has a confluent set of relations. If in doubt use Make Confluent tool first.

Sum

Selecting Tools | Sum? opens a dialogue window allowing selection of a possible sum object from the current category from a drop-down and specification of (possibly composite) first and second injections, then testing the cospan for being a sum.

WARNING: The codomain of each injection must be the possible sum object. The test is only valid on a confluent category. (If in doubt use the Make Confluent tool).

NOTE: This feature has not yet been implemented.

Terminal Object

Select Tools | Terminal Object?. This command tests objects of the current category (chosen by a dialogue) for being terminal. For each selected object which is not terminal a reason is given.

WARNING: This command will not operate correctly unless the category has a confluent set of relations. If in doubt use Make Confluent Tool first.

Product

Selecting Tools | Product? opens a dialogue window allowing selection of a possible product object from the current category from a drop-down and specification of (possibly composite) first and second projections, then testing the span for being a product.

WARNING: The domain of each projection must be the possible product object. The test is only valid on a confluent category. (If in doubt use the Make Confluent tool).

NOTE: This feature has not yet been implemented.

SECTION 5: FUNCTORS

Introduction

A functor is a “structure preserving” interpretation of one finitely presented category into another.

A functor preserves the structure of a category because it sends:

Objects to Objects
Arrows to Arrows

and it preserves the composition that is provided by relations.

Functor File Types

In Category Theory Database Tools, there are two different file types that are associated with functors.

***.fun**

These files are a text only representation of a functor. These files can be both opened and saved. Since a graphical representation of the two categories involved is not saved, the graphical representations are generated randomly upon opening the *.FUN file.

***.fgl**

These files contain both a text and graphical representation of the categories involved in a given functor as well as the functor information itself. This type of file can be both opened and saved.

FUN File Format

The text representation of functors is stored in the following three part format.

Part one consists of a category stored in the CAT file format.

Part two consists of a second category stored in the CAT file format.

Part three consists of the functor information. In this part, any line that is a comment must start with the "#" character. On the line following the word "functor" is the functor name. The mapping of objects to objects is listed between the words "objects" and "arrows". The mapping of arrows to paths is listed after the word "arrows".

NOTE: The words "category", "functor", "objects", "arrows", and "gml" are key words and cannot be used as the name of the categories, functors between them or as the name of an object or arrow within categories.

An example of a typical FUN file can be found in Appendix A.

FGL File Format

The text representation of functors is stored in the following three part format.

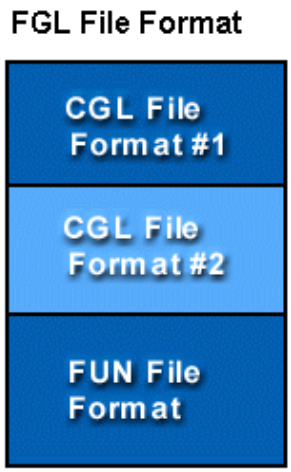
Part one consists of a category stored in the CGL file format.

Part two consists of a second category stored in the CGL file format.

Part three consists of the functor information. In this part, any line that is a comment must start with the "#" character. On the line following the word "functor" is the functor name. The mapping of objects to objects is listed between the words "objects" and "arrows". The mapping of arrows to paths is listed after the word "arrows".

NOTE: The words "category", "functor", "objects", "arrows", and "gml" are key words and cannot be used as the name of the categories, functors between them or as the name of an object or arrow within categories.

An example of a typical FGL file can be found in Appendix A.

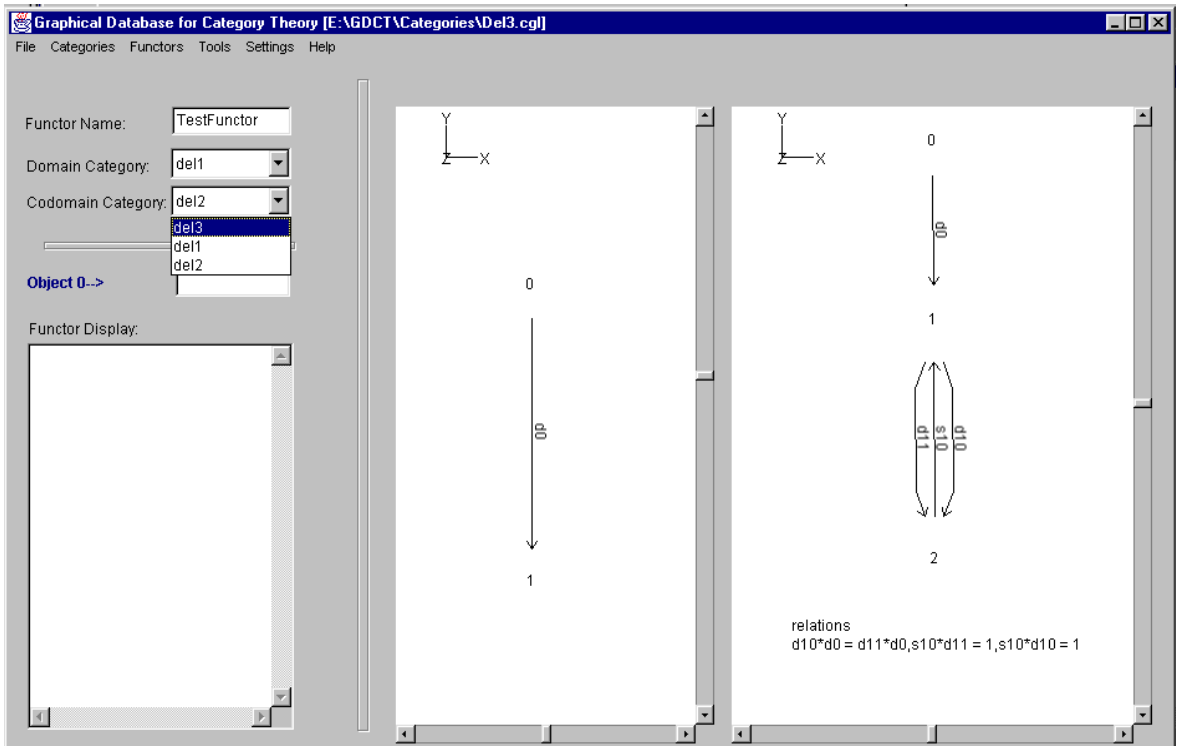


Creating Functors

To create a Functor in memory the user selects Functors | Create Functor.

First the domain and codomain categories must be selected from those in memory using the drop-down category lists. The categories will be displayed in two windows.

Enter a (descriptive!) name in the "Functor Name" box and type Enter.



For each object type the name of its image under the functor in the box which now appears and type Enter. For each arrow type the name of its image under the functor (which may be a composite arrow) in the box and type Enter.

During creation of a functor its action is displayed in the Functor display box.

WARNING: The image of an arrow must have domain and codomain which are the images of the domain and codomain of the arrow.

This completes the creation of the new functor. Following creation it is possible to animate the display of the functor using the buttons below the category display windows.

WARNING: The newly created functor is stored in memory. To save the functor to disk select File | Save Functor or Functors | Save Functor. The functor will be saved in the format selected using a save dialog.

Opening Functors

In order to open a text functor file (.fun) select File | Open Functor | Open Functor(FUN) or Functors | Open Functor | Open Functor(FUN). This opens a file selection window. Opening graphical functor files (.fgl) is similar. A shortcut to open a .fgl functor file is to press CTRL+F. To open a functor simply double click on the functor's name or click once on the functor name and press the OK button.

If there is any error in the format of the selected file an error message is displayed and the open is aborted.

NOTE: Opening FGL functor files is not yet implemented

Downloading Functors

In order to download a text functor file (.fun) select File | Open Functor | Download Functor(FUN) or Functors | Open Functor | Download Functor(FUN).

Downloading a graphical functor file (.fgl) is similar, select File | Open Functor | Download Functor(FGL) or Functors | Open Functor | Download Functor(FGL).

Choosing any of the above options will display a list of functors in a selection box in a new frame. The functors listed are located on the server specified in the frame's title bar. To select a functor to download simply double click on the functor's name or click once on the functor name and press the OK button. Either of these methods will display the downloaded functor. If the program cannot locate the files on the server an error message will be displayed. This could result from the user not being connected to the Internet or as a result of a server error.

To change the settings for downloading functors, select Settings | Set Server. This displays a dialog for changing the default server and directories from which categories and functors can be loaded using the appropriate load command.

NOTE: Downloading FGL functor files is not yet implemented.

Saving Functors

In order to save a functor in memory as a graphical file (.fgl) select File | Save Functor | Save Functor(FGL) or Functors | Save Functor | Save Functor(FGL). This opens a file selection window. To save a functor to an existing file simply double click on the functor file name. A new file is created by typing its name in the dialogue and clicking OK.

Saving functor files as text only (.fun) is similar.

Removing Functors

Selecting Functors | Remove Functor opens a list of the functors currently in memory. Select from the list and click OK to remove a functor.

Printing Functors

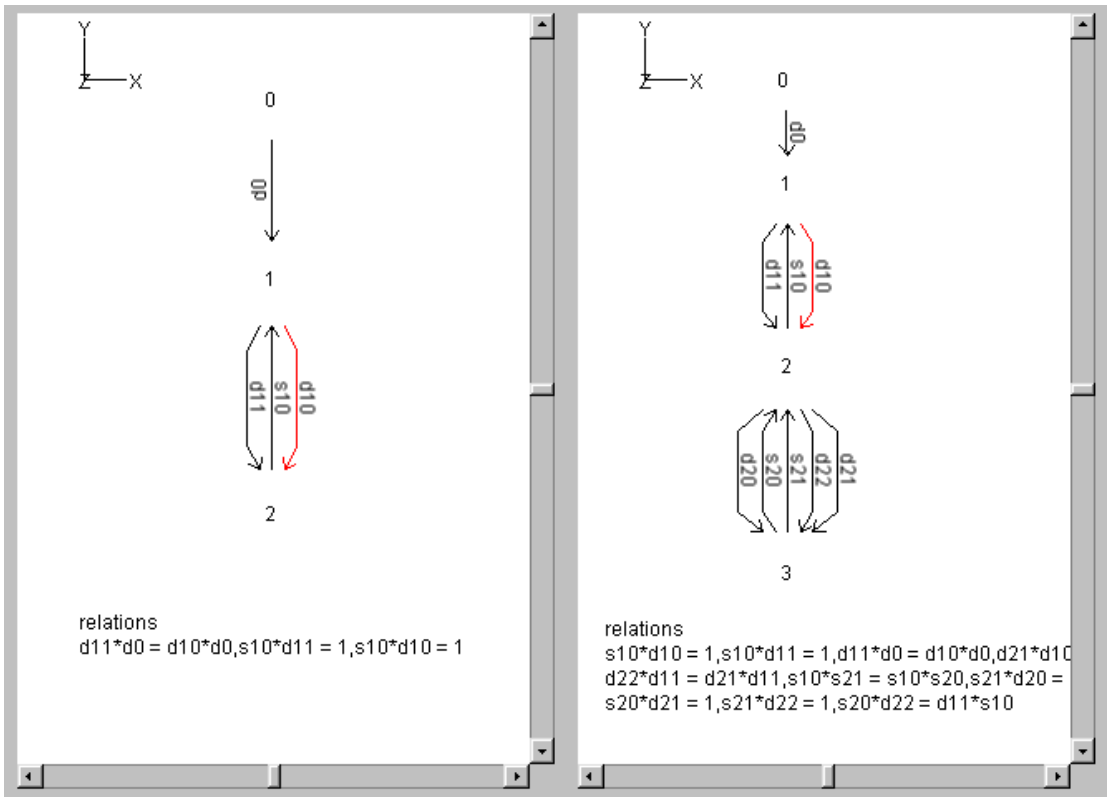
To print the text version of the currently displayed functor, select File | Print | Print Functor (Text).

Functor Animation

The mappings on objects and arrows defined by a functor can be highlighted through the use of animation. There are two ways to animate a functor.

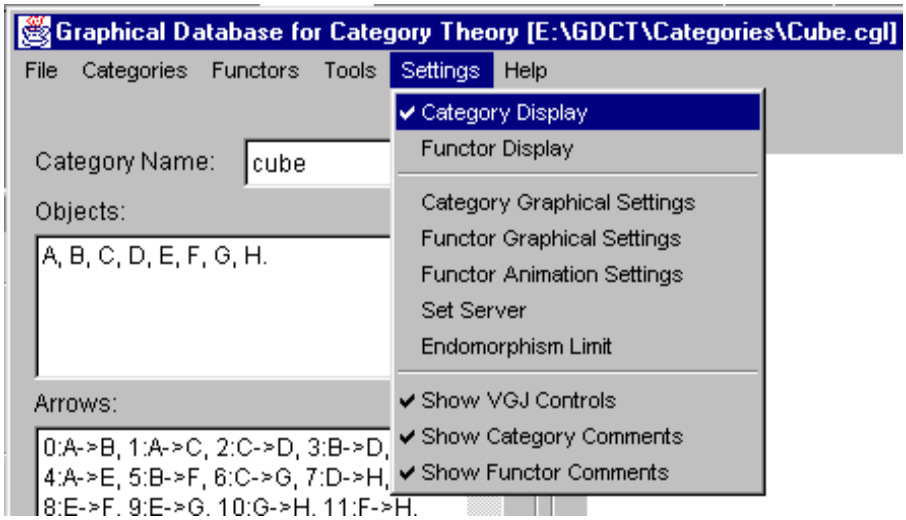
The first way is to use the Back and Forward buttons below the graphical display of the functor to step though all of the mappings. The animation can be restarted by pressing the Reset button.

The second way to animate a functor is to press the Auto Animate button which will animate all of the mappings at the animation speed indicated in Settings | Functor Animation Settings.



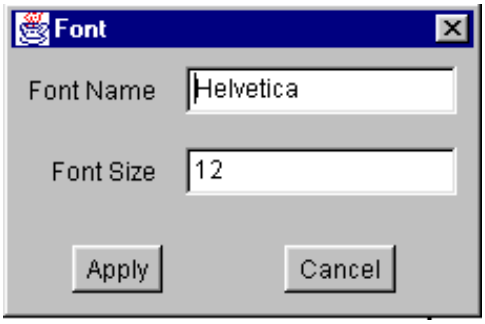
Switching Between Category and Functor Displays

Selecting either of Settings | Category Display or Settings | Functor Display allows the user to switch between the display of categories and the display of functors.



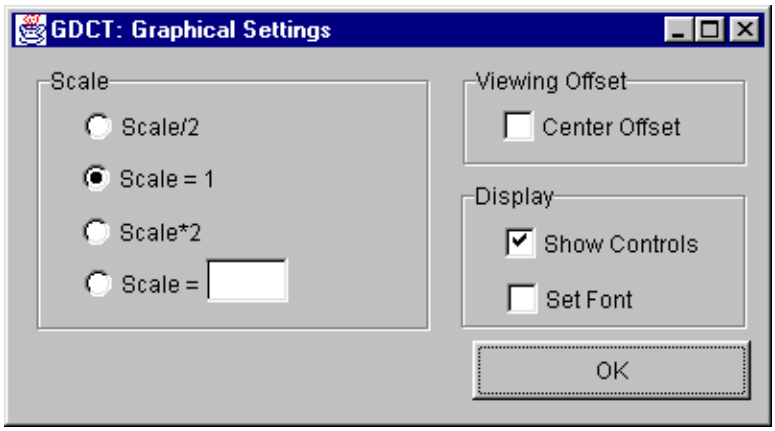
Font Settings

Selecting Settings | Category Settings and then checking the box in front of “Set Font”. This displays a dialog in which the user can change the font settings within the program.



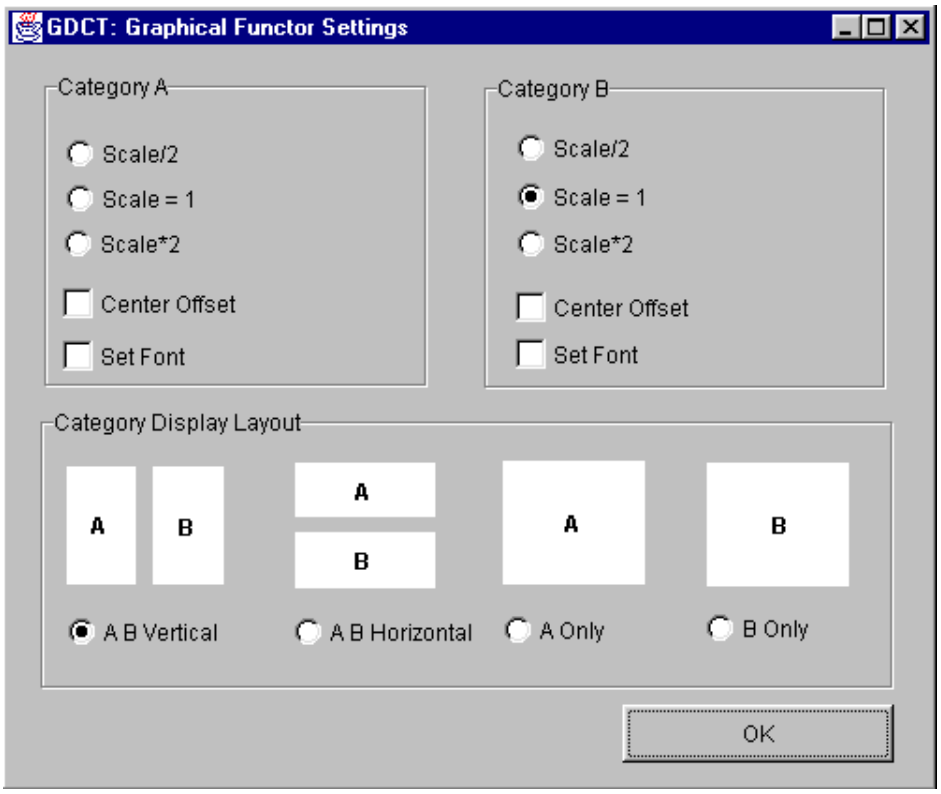
Category Graphical Settings

Selecting Settings | Category Graphical Settings displays a dialog for changing the graphical settings in the category display.



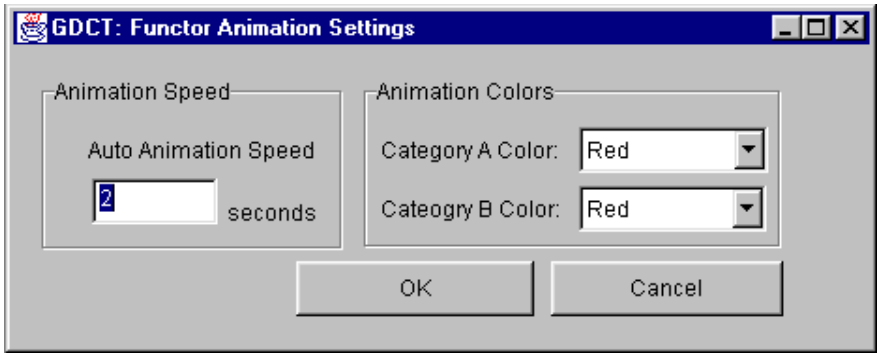
Functor Graphical Settings

Selecting Settings | Functor Graphical Settings displays a dialog for changing the graphical settings in the functor display.



Functor Animation Settings

Selecting Settings | Functor Animation Settings displays a dialog for changing the animation delay time (default 2 seconds) and the animation colors for the domain and codomain categories.

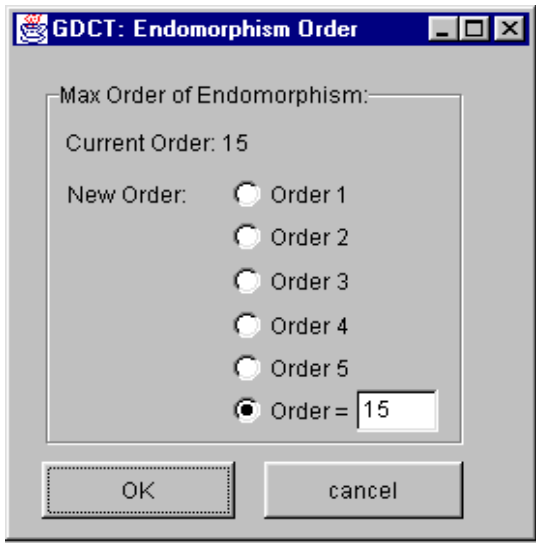


Server Settings

Selecting Settings | Set Server displays a dialog for changing the default server and directories from which categories and functors can be loaded using the appropriate load command.

Endomorphism Limit

Selecting Settings | Endomorphism Limit displays a dialog for changing the endomorphism limit.

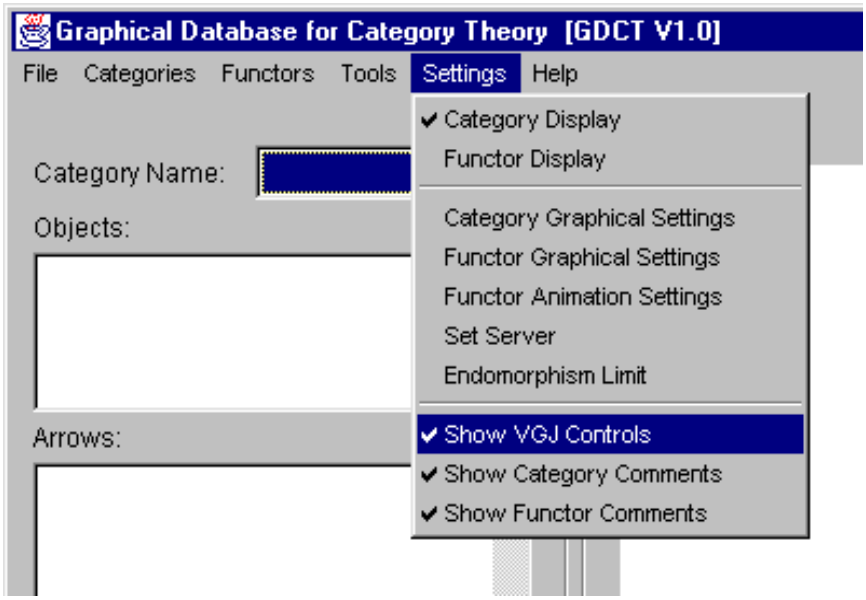


Displaying the VGJ Controls

If Settings | Show VGJ Controls is selected it displays the VGJ controls in the category display. If deselected, it hides them.

Displaying the Category Comments

If Settings | Show Category Comments is selected it displays the comments in the category display. If deselected, it hides them.



Displaying the Functor Comments

If Settings | Show Functor Comments is selected it displays the comments in the functor display. If deselected, it hides them.

APPENDICES

APPENDIX A: EXAMPLE FILES

Example CAT File

```
#This is an example category
category
  sumone
objects
  A, B, C, D.
arrows
  f:A->C, g:B->C, h:C->D, i:A->D,
  j:B->D.
relations
  h*f = i, h*g = j.
```

Example CGL File

```
# A commutative square.
category
 csquare
objects
 A, B, C, D.
arrows
 f:A->B, g:A->C, h:B->D, k:C->D.
relations
 h*f = k*g.
gml
 graph [
  directed 1
  node [
    id 0
    label "A"
    graphics [
      Image [
        Type ""
        Location ""
      ]
      center [
        x -48.30000000000001
        y 49.0
        z 0.0
      ]
      width 15.94
      height 27.299999999999997
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
```

```
node [
  id 1
  label "B"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x -48.0
      y -53.0
      z 0.0
    ]
    width 17.36
    height 27.299999999999997
    depth 20.0
  ]
  vgj [
    labelPosition "in"
    shape "Oval"
  ]
]
node [
  id 2
  label "C"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x 51.69999999999999
      y 49.0
      z 0.0
    ]
    width 18.78
    height 27.299999999999997
    depth 20.0
  ]
  vgj [
    labelPosition "in"
    shape "Oval"
  ]
]
node [
  id 3
  label "D"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x 50.59999999999995
      y -53.0
```

```
        z 0.0
      ]
      width 18.78
      height 27.299999999999997
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  node [
    id 4
    label "relations\nh*f = k*g"
    graphics [
      Image [
        Type ""
        Location ""
      ]
      center [
        x 169.0
        y -4.0
        z 0.0
      ]
      width 20.0
      height 20.0
      depth 20.0
    ]
    vgj [
      labelPosition "center"
      shape "Oval"
    ]
  ]
  edge [
    linestyle "solid"
    label "k"
    source 2
    target 3
  ]
  edge [
    linestyle "solid"
    label "h"
    source 1
    target 3
  ]
  edge [
    linestyle "solid"
    label "g"
    source 0
    target 2
  ]
  edge [
    linestyle "solid"
    label "f"
    source 0
```

```
    target 1
  ]
 ]
```

Example FUN File

```
#This is an example of an fun file
#This is the first category
category
  del2
objects
  0, 1, 2.
arrows
  d0:0->1, d10:1->2, d11:1->2, s10:2->1.
relations
  d11*d0 = d10*d0, s10*d11 = 1, s10*d10 = 1.

#This is the second category
category
  del3
objects
  0, 1, 2, 3.
arrows
  d0:0->1, d10:1->2, d11:1->2, s10:2->1,
  d20:2->3, d21:2->3, d22:2->3, s20:3->2,
  s21:3->2.
relations
  s10*d10 = 1, s10*d11 = 1, d11*d0 = d10*d0, d21*d10 = d20*d10,
  d22*d10 = d20*d11, d22*d11 = d21*d11, s10*s21 = s10*s20, s21*d20 = d10*s10,
  s20*d20 = 1, s21*d21 = 1, s20*d21 = 1, s21*d22 = 1,
  s20*d22 = d11*s10.

#This is the functor information
functor
  del23
objects
  0 |--> 0, 1 |--> 1, 2 |--> 2.
arrows
  d0 |--> d0, d10 |--> d10, d11 |--> d11, s10 |--> s10.
```

Example FGL File

```
#This is an example of an fgl file
#This is the first category
category
  del2
objects
  0, 1, 2.
arrows
  d0:0->1, d10:1->2, d11:1->2, s10:2->1.
relations
  d11*d0 = d10*d0, s10*d11 = 1, s10*d10 = 1.
gml
  graph [
```

```
directed 1
node [
  id 0
  label "0"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x 25.0
      y -120.0
      z 0.0
    ]
    width 15.94
    height 27.299999999999997
    depth 20.0
  ]
  vgj [
    labelPosition "in"
    shape "Oval"
  ]
]
node [
  id 1
  label "1"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x -60.0
      y 69.0
      z 0.0
    ]
    width 15.94
    height 27.299999999999997
    depth 20.0
  ]
  vgj [
    labelPosition "in"
    shape "Oval"
  ]
]
node [
  id 2
  label "2"
  graphics [
    Image [
      Type ""
      Location ""
    ]
    center [
      x -17.0
```

```
        y -68.0
        z 0.0
      ]
      width 15.94
      height 27.299999999999997
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  node [
    id 4
    label "relations\nd11*d0 = d10*d0,s10*d11 = 1,s10*d10 = 1"
    graphics [
      Image [
        Type ""
        Location ""
      ]
      center [
        x 0.0
        y -150.0
        z 0.0
      ]
      width 336.85999999999996
      height 48.599999999999994
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  edge [
    linestyle "solid"
    label "d11,d10"
    source 1
    target 2
  ]
  edge [
    linestyle "solid"
    label "d0"
    source 0
    target 1
  ]
  edge [
    linestyle "solid"
    label "s10"
    source 2
    target 1
  ]
 ]

#This is the second category
```

category
del3
objects
0, 1, 2, 3.
arrows
d0:0->1, d10:1->2, d11:1->2, s10:2->1,
d20:2->3, d21:2->3, d22:2->3, s20:3->2,
s21:3->2.
relations
s10*d10 = 1, s10*d11 = 1, d11*d0 = d10*d0, d21*d10 = d20*d10,
d22*d10 = d20*d11, d22*d11 = d21*d11, s10*s21 = s10*s20, s21*d20 = d10*s10,
s20*d20 = 1, s21*d21 = 1, s20*d21 = 1, s21*d22 = 1,
s20*d22 = d11*s10.
gml

```
graph [
 directed 1
 node [
   id 0
   label "0"
   graphics [
     Image [
       Type ""
       Location ""
     ]
     center [
       x 37.0
       y 131.0
       z 0.0
     ]
     width 15.94
     height 27.299999999999997
     depth 20.0
   ]
   vgj [
     labelPosition "in"
     shape "Oval"
   ]
 ]
 node [
   id 1
   label "1"
   graphics [
     Image [
       Type ""
       Location ""
     ]
     center [
       x 0.0
       y 73.0
       z 0.0
     ]
     width 15.94
     height 27.299999999999997
     depth 20.0
   ]
```

```
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  node [
    id 2
    label "2"
    graphics [
      Image [
        Type ""
        Location ""
      ]
      center [
        x 121.0
        y 139.0
        z 0.0
      ]
      width 15.94
      height 27.299999999999997
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  node [
    id 3
    label "3"
    graphics [
      Image [
        Type ""
        Location ""
      ]
      center [
        x -52.0
        y -21.0
        z 0.0
      ]
      width 15.94
      height 27.299999999999997
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  node [
    id 5
    label "relations\ns10*d10 = 1,s10*d11 = 1,d11*d0 = d10*d0,d21*d10 = d20*d10,d22*d10 =
d20*d11,\nd22*d11 = d21*d11,s10*s21 = s10*s20,s21*d20 = d10*s10,s20*d20 = 1,s21*d21 =
1,\ns20*d21 = 1,s21*d22 = 1,s20*d22 = d11*s10"
    graphics [
```

```
      Image [
        Type ""
        Location ""
      ]
      center [
        x 0.0
        y -150.0
        z 0.0
      ]
      width 673.4
      height 91.19999999999999
      depth 20.0
    ]
    vgj [
      labelPosition "in"
      shape "Oval"
    ]
  ]
  edge [
    linestyle "solid"
    label "d22,d20,d21"
    source 2
    target 3
  ]
  edge [
    linestyle "solid"
    label "d11,d10"
    source 1
    target 2
  ]
  edge [
    linestyle "solid"
    label "s21,s20"
    source 3
    target 2
  ]
  edge [
    linestyle "solid"
    label "d0"
    source 0
    target 1
  ]
  edge [
    linestyle "solid"
    label "s10"
    source 2
    target 1
  ]
 ]
```

#This is the functor information

functor

del23

objects

0 |--> 0, 1 |--> 1, 2 |--> 2.

arrows
d0 |--> d0, d10 |--> d10, d11 |--> d11, s10 |--> s10.

APPENDIX B: MENU ITEM SHORTCUTS

The following short cuts to menu items are supported in Graphical Database for Category Theory:

CTRL+C	Create a Category
CTRL+H	Display Help File for GDCT
CTRL+F	Open a FGL Functor File
CTRL+G	Save a FGL Functor File
CTRL+D	Download a CGL Category File
CTRL+O	Open a CGL Category File
CTRL+S	Save a CGL Category File

APPENDIX C: MISC. INFORMATION

Version Information

The Graphical Database for Category Theory is a research project at Mount Allison University that now includes over 80 Java source files and over 30000 lines of code. Version 1.0 of the GDCT was released as a preliminary version which provides the basic functionality and demonstrates the abilities of the final version of the application. Currently, the below list of known problems is being dealt with and will be fixed by September 2000. It is the goal of the GDCT Development Team to have a stable version of the application available for download by that date.

Version 1.0: Released March 20, 2000

This is the preliminary version of Graphical Database for Category Theory The release information for this version is available at http://mathcs.mta.ca/research/rosebrugh/gdct/v1release.htm.

Version 1.1: Release Date Summer 2000

This version of GDCT will contain the following fixs and enhancements:

- The help file system has been rewrote and now includes a complete set of help files which will better enable the user to use the software
- Contact information in the bug report dialog has been updated
- Upon closing the application, a dialog window now prompts the user to see if they want to save changes in modified categories. This dialog window also tells the user what modifications were made so that they can more accuratly judge if they want to save these changes.
- The display of categories in the "Make Confluent Tool was changed so that a category isn't redrawn randomly. Instead, GDCT will only update the label of the node containing the list of relations.
- In "Add Data" for categories, the category is no longer redrawn randomly. New data is now added to the existing graphical representation as opposed to creating a new random representation.
- In "Remove Data" the problem where not all data is removed in some categories has been fixed Also, in "Remove Data" for categories, the text display is updated and the graphical display is updated as data is removed.
- In "View GML", the problem with the search feature not working right away has been corrected.
- The "Make Confluent" algorithm has been extensively error checked and all known bugs have been

fixed.

- Bugs in "Make Dual Category" have been fixed and the make display of a newly created dual category is no longer determined randomly. Instead, the display is based on the display of the category that was originally used to create the dual category.
- The"Equality of Composites" tool has been tested and all problems have been fixed.
- In the "Initial Object" and "Terminal Object" tools a warning is now displayed if the endomorphism limit has been reached.
- GDCT 1.0 was developed using Borland JBuilder 2.0 and used some borland libraries such as borland.jbcl.control and borland.jbcl.layout. This caused certain compiling problems in non-Borland environments. To correct this situation, all elements of borland libraries have been removed.

Known Bugs

Below is a list of known bugs in the current version of the GDCT application. If you have encountered additional bugs please fill out a comment form at http://mathcs.mta.ca/research/rosebrugh/gdct/contact.htm.

- The "Initial Object" algorithm needs to be rechecked when it is applied to categories with identity arrows in relations.
- The "Terminal Object" algorithm needs to be rechecked when it is applied to categories with identity arrows in relations.
- The "Sum" algorithm needs to be fixed.
- Using a modified "Sum" algorithm the "Product" algorithm needs to be completely developed
- In "Graphical Functor Settings" when the Category layout display is changed, the category graphical representations can no longer be scrolled and objects and arrows can no longer be selected. Animation of functor and other settings still work.

Proposed Enhancements

Below is a list of enhancements that will be made to the GDCT application. If you have ideas for additional enhancements please fill out a comment form at http://mathcs.mta.ca/research/rosebrugh/gdct/contact.htm.

.

- Open for *.FGL files
- Download for *.FGL files Open Recent for local *.FGL, *.FUN files
- Open Recent for server *.CAT, *.CGL, *.FUN, *.FGL files
- "Server Settings" should be saved in a file so that the user doesn’t have to change them every time the program is started. The current default settings should remain in a class file to be displayed in the event that the file containing the current settings is unavailable.
- "Endomorphism Limit" should be saved in a file as above for "Server Settings".