An Overview of CalICo *

M. Delest, N. Rouillon, J.M. Fédou[†]
LaBRI, Université Bordeaux I,
Unité associée au C.N.R.S. n° 1304,
33405 Talence Cedex, France.

1 Introduction

Combinatorics has always been concerned by images and drawings because they give interpretations of enumeration formulae leading to simple proofs of these formulae, and sometimes they are themselves central to the problem. Even if some small example drawings do not contain all elements of the proof, they are often useful to guide the intuition.
On recent years, enumerative combinatorics has developed from this point of view, in interactions with theoretical computer science and graphics computer science. See for example [AEJV89]. In the field of bijective combinatorics linked with computer science, the DSV-methodology [Vie92], [Del95] has led to new formulae . The problem often consists in esablishing enumerating formulas by constructing bijections between sets of objects having the same cardinality. Computations are then suppressed and are replaced by effective constructions of bijective correspondences between sets. The formulae may then be viewed as reflecting combinatorial properties of the objects involved. The problem may also consist in finding geometrical transformations of objects reflecting the equations which relate them.
For several years already, enumerative combinatorics research uses symbolic computation. Most of the Computer Algebra Systems, including Macsyma [Sym84], Maple [CGG+92] and Mathematica [Wol88], allow easy algebraic and graphic manipulations but none of them gives library of functions working directly and interactively on the graphic representation of the object. Some projects have leaded to tools for performing such actions. For example, we can quote the Mathematica library done by Skiena [Ski90] which give tools for implementing combinatorial objects. This library do not allow to interact with the graphics itself. The aim of the CalICo project is the development of a software in which the manipulation of combinatorial objects can be made through all their representations: coding, formal power series and overall pictures.

We present here a short description of the whole software, see [Rou94] for complete information. In the next section, we show an overview of its structure. To guide the reader in this paper, we show simple examples. Then we describe the tutor (which concerns combinatorics knowledge) that is the heart of the CalICo environment. No formal description are given here. The reader can get more details from [Rou94] and [DFMR96]. In section 4, we describe the primary interface that allows the user to launch an application in the system and the model that we have designed to produce easily new interfaces. Some examples of this model are also given. We do not describe the communication software. This was done completely in [Rou94] and [GGR94].

2 First approach

In this section, we describe the general concept of CalICo. CalICo gathers in a single software environment a set of different applications, called *workshops*. These workshops can be run on remote computers linked by the Internet network. This software environment allows message exchange between each of those workshops. It offers a graphical and mathematical working environment.

We use the term *workshop* instead of the more usual terms *application*, *program* or *software* because it gives the intuitive notion of workshops working together in a factory. A workshop in CalICo is independent of the others and dedicated to a specific job. Independence means that all the workshops could be used without CalICo. Of course, in this case, the workshop loses some of its functionalities.

*Partially supported by EC grant CHRX-CT93-0400, PRC-Math-Info, GDR Programmation. This paper is an abstract of one to appear in Human Interaction for Symbolic Computation, *series:* Texts and Monographs in Symbolic Computation, éd. N. Kajler, Springer-Verlag. Ask for the software at calico@labri.u-bordeaux.fr.

[†]e-mail:calico@labri.u-bordeaux.fr

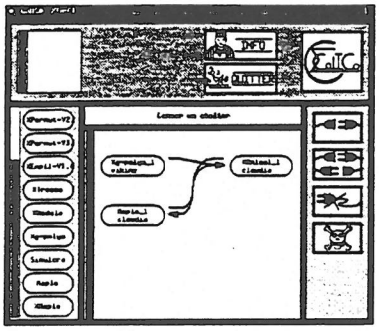

Figure 1. Piloting interface after connections.

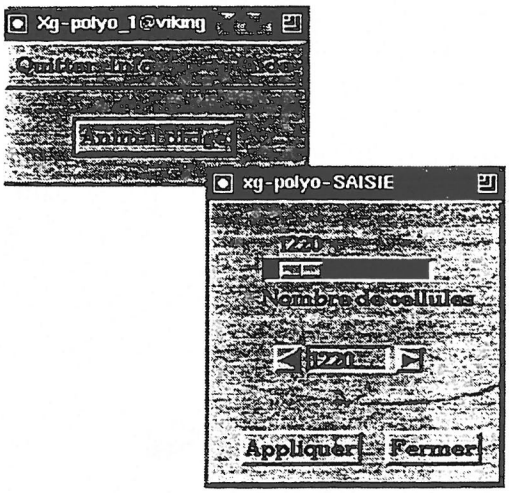

Figure 2. Random generation of animals.

Various workshops coexist within CalICo. The *tutor* is the main workshop. The others could be divided in three groups: visualization workshops, computation workshops, generating workshops. All of them work in a concurrent way. They are gathered together in a distributed architecture. The client/server model is used. The ultimate goal is to produce a system that can be operated even by a non computer scientist and to use methods leading to easy extensions and generalizations of already existing modules.
On the one hand, we group in a single software environment all the workshops we need. We propose a user-friendly graphical interface to allow communication between workshops. On the other hand, a *distributed architecture* has been chosen. Each workshop can be run on a remote computer. This distributed aspect allows parallel treatments of several processes. The interest is in running generating workshops on different computers to increase the speed of the system.
Workshops could also be dynamicaly visualized on a remote display. The interest is double. First, one can use several terminals or computers to display its work. Second, this opportunity allows us to visualize the result of a computation (by way of a drawing with colours) on the display of a person in another city or another country. This last notion is very close to the groupware field [BLK92]. In this aspect, we did not use X11 functions. Only the data giving the coding of the objects are transferred between two remote applications [GGR94].
From a technical point of view, the implementation of this work is based on the *client/server model.* GeCI (Gestionnaire de Communications Interactif [1]) is a unifying program. It manages dynamic connections between programs (even on remote computers) and the exchange of messages between them. We want to enhance a particular point: the data are transmitted if they are on the same display by a copy/paste operation even if the workshops are running on different computers.
CalICo can be configured at different levels. Some configurations can be done by a simple user, others need to know more about computer systems.

3 The use of CalICo as illustrated by examples

When one issues the `calico` command at the top level shell, its *configuration file*, which name is *.CalICo.config*, is read by a parser and a scanner [2]. This file has to be found in the user's home directory in the directory *.CalICo*. If it is not found there, a file by defect is copied from the system. This file contains the description of the available workshops and therefore describes exhaustively what we call piloting interface for CalICo (see figure 1).
On the left side of the window of the piloting interface, one finds all available workshops. On the right side, the buttons allow to connect the output of a workshop to the input of another and to destroy a launched workshop. As an example, launch successively the three workshops, *Xg-polyo, XAnimal et Maple* and connect *Xg-polyo* to *XAnimal* and *XAnimal* to *Maple* (and reciprocally). We get the figure 1.
When the user launches a workshop, a window gives him the possibility to change the name of this workshop occurrence. One can choose for the execution (and/or the visualization) machine other than the local one. If the user chooses a remote machine for the execution, it is necessary that CalICo daemon process called *calicod*. runs on this machine. Running this daemon process is achieved by issuing the `calicod` command at the top level of any user on the machine. On a workstation, user must remember to add this host to the list of machines that are allowed to connect to the local X server. Use server access control program for X and execute `xhost <hostname>`.

[1] *ie:* Interactive Comunications Manager
[2] GNU Products, Flex and Bison.

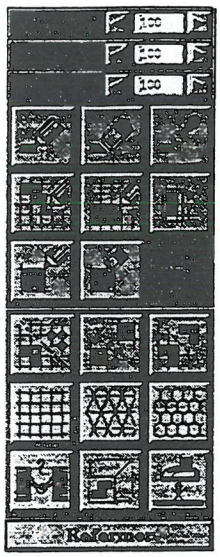

Figure 3. Toolbox of animals workshops.

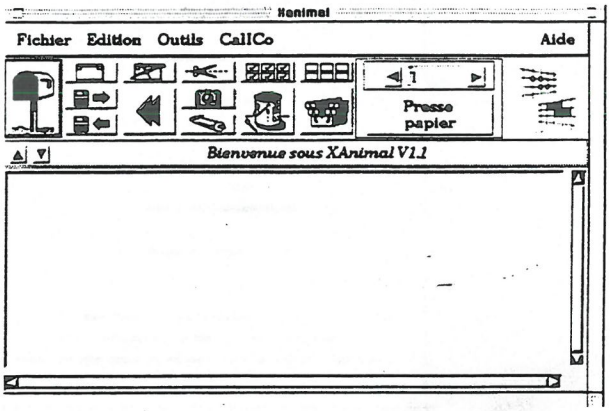

Figure 4. Animals and polyominoes visualization.

The launching is done by clicking the *OK* button. For example *Xg-polyo* (figure 2), generator for directed animals is launched on a remote machine (here viking, the local machine being claudia as illustrates on figure 1). *Xanimal* (figure 4), workshop for animals and polyominoes is launched on the local machine claudia.
The motivation of such configuration is a work presented in [DR92]. The work has been realized using the computer algebra package *Maple*. This is why a third workshop appears in the figure 1.
Let us consider another situation when one wants to test some Maple functions on a heap of pieces [Vie86]. In this case, we need a big number of input/output moves between the manipulation of heaps in the graphical interface and Maple. Here one desire rather to use *Xmaple* to have the ease of use of the IRIS interface. Exchanges can then be made by *copy/paste* using the mouse and the CONTROL button [3].
In the workshop *XEmpilement*, by a CONTROL-LeftButton action in a heap (figure 7) one can select a copy of the formal coding underlying the heap. In *XMaple*, a MiddleButton action pastes the coding as observed on the figure 6. Reciprocally, in *XMaple*, after application of the function f on the heap, one can copy using the mouse the consequent coding. If one pastes (clicking the mailbox icon with the MiddleButton) in the workshop, the formal coding is received and then translated. A new heap is pictured, figure 8.

4 The tutor

CalICo's tutor is intended for the manipulation of any combinatorial objects. Its two main purposes are to classify objects by an automatic process and to use colours on objects in order to obtain a symbolic representation of their combinatorial properties.
Manipulated combinatorial objects, called *elements*, are arranged in subsets, called *universe, worlds, family* and *sisterhood.*

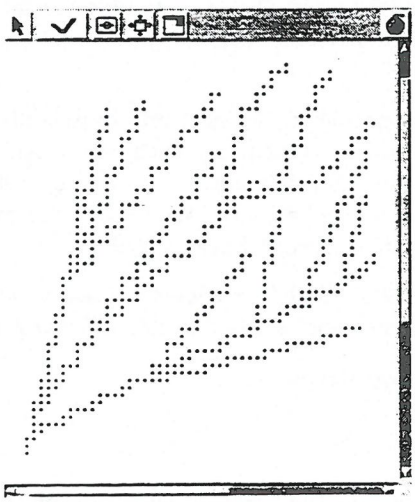

Figure 5. A random directed animal.

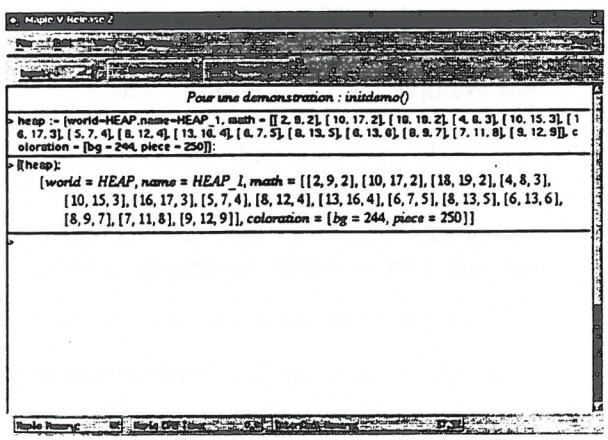

Figure 6. Xmaple session, with paste and copy.

[3] In the following, CONTROL-LeftButton means at the same time press CONTROL and Left Button, where Left Button (respectively Middle Button) means to press on left (respectively middle) button of the mouse.

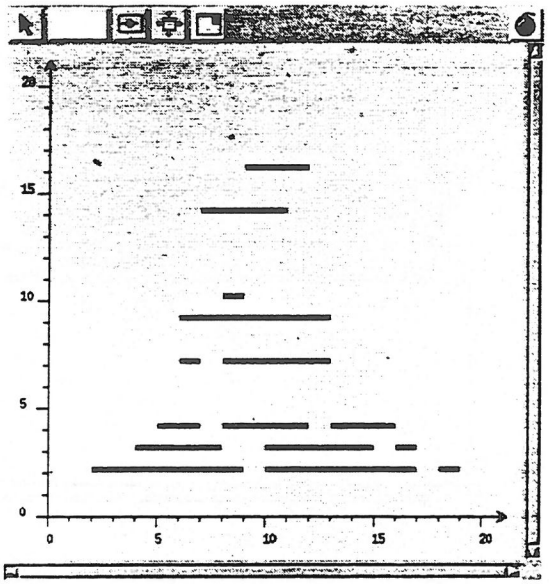

Figure 7. Initial heap, called h.

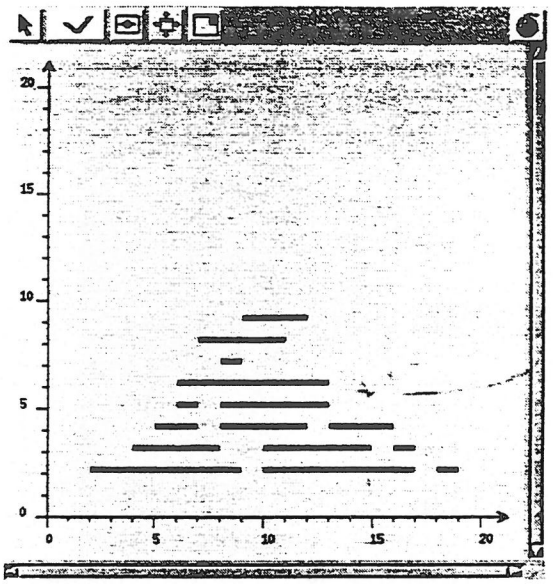

Figure 8. Pasted heap after the computation $f(h)$.

The CalICo *universe* represents the set of all existing objects and structures in CalICo. Other universes exist, with which CalICo may exchange informations: Maple [CGG+92], Mathematica [Wol88], $\Lambda\Upsilon\Omega$[FSZ88], Gaïa [Zim94, FZC94] and Darwin [BC88] for examples. A *world* is a sub-universe gathering together combinatorial objects that belong to the same class. CalICo's universe is composed of different worlds such as words world, polyominoes world or trees world. By analogy with statistics, we define quantitative and qualitative predicates. Let P be a predicate on an entity E. If there exists an application, denote by α from E to $\mathbb{Z}$ and an integer n such that:

$$\forall x \in E,\ P(x) \implies \alpha(x) \leq n,$$

we say that P is a *quantitative predicate*. If a predicate is not quantitative, we say that it is a *qualitative predicate*. Let E be a set of combinatorial objects of the same world and defined by the predicate P. If P is a quantitative predicate and if E is finite then E is a *sisterhood*, otherwise E is a *family*. For example, the set of trees verifying the qualitative predicate *to be a binary tree* is a family. The set of binary trees with size equal to 3 is a finite set. It is a sisterhood defined by the quantitative predicate *to have 3 vertices*. Any combinatorial object handled in CalICo is an *element*.

> *Remark.* In what follows, we will use the term *entity* when we want to refer either to a family or to a sisterhood.

We associate a *dependency graph* with each world in CalICo. Defining an entity implies new inheritance relationships in the dependency graph. The following information is required: the world where the entity will be classified, the entity's name, a parent entity, an additional predicate and a set of entities that become the parents of the created entity. Elements brought in CalICo's tutor have a visual representation which is an important feature in their study. The colour is used in order to visualize the combinatorial properties. Significance to the colouring can be understood by the notions of *attribute, colouring* and *interpretation* that we introduce below.

Attributes are linked to the drawing of an element. A drawing is geometrically split into indivisible structures. For example the tree drawing needs a circle for each vertex and a segment for each edge. An *attribute* is a type of indivisible structure. Occurrences of attribute are the element's parts that can be coloured. For each world, there is a finite number of attributes. For example, a tree has three attributes: vertex, edge and background. Note that for any object one can put a colour on the object's background.

Let us call weight the value of a statistic computed on an element. CalICo associates a colour with weights computed on elements. Intuitively, defining a colouring for an element consists in the following actions :

- consider the element on a geometric point of view selecting attributes
- colour the selected attributes according to the weight.

Colour an attribute of an element is not sufficient. Attributes are the direct illustration of the element's geometry. We need for the colouring to be close to the mathematical properties of the element.
For this purpose, we introduce *interpretations*, which are the links between the geometry and the combinatorics of objects.

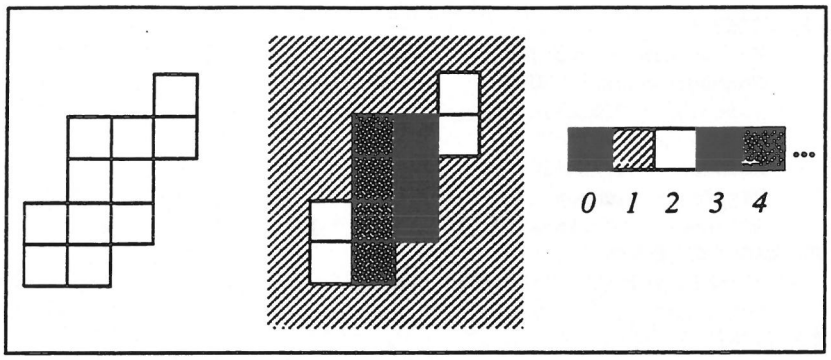

Figure 9: Example of colouring

Example. In figure 9, we see two occurrences of a polyominoe. The first one is without colouring. The second one has a colouring which uses two attributes: the cells and the background. The background is hachured if the polyominoe is a parallelogram polyominoe [4] and is black otherwise. The property "to be a parallelogram" for a polyominoe can be viewed as a statistic. The result, a boolean, is the weight for the polyominoe regarding this statistic. We associate two colours (hachured and black) with this weight. Until now, there is no interpretation. If we want to put the colour corresponding to the height of a column on the column, we have to use an interpretation. The statistic is the computation of the height of the column, the result of it is a weight. Setting this weight symbolically represented by a colour, on each cell of this column corresponds to the application of an interpretation on the polyominoe.
The implementation of definitions and properties presented in this section (worlds, families, brotherhoods, elements, interpretations, statistics) were done in Maple. We have defined an ***unique representation*** of these entities in order to handle them efficiently. This representation uses formal power series and deals with both mathematical and graphical components of the elements. Fundamental operations are defined on these entities: the falling operation which agree with the inheritance graph, recognition of home domains, colouring according to statistics and interpretations.
We have said that the colour in CalICo is used for suggesting the bijections between subsets of worlds. To discover a bijection from a set E_d to a set E_a corresponds to find a function of translation from the coding in the set E_d to the coding in the set E_a.
With a view of suggesting bijections automatically, three approaches have been proposed in [Rou94]:

- to compare sisterhood descriptions among themselves,
- to vizualise the modification after each modification of an element,

Recently, Dutour and Fédou [DF94] have proposed a new approach by the use of objects grammar .

5 The Piloting Interface

In order to conduct a working session using several workshops, CalICo uses a main graphical interface, called xgeci (see figures 1 page 2). It allows to launch different workshops and to set communications between them. Thus the user can construct a *virtual machine* [GBD+93]. Each application can be run on different machines and visualized on different displays.
The interface xgeci allows to launch visualization workshops, symbolic computation workshops or any other programs which have been declared in the configuration file ".CalICo/.CalICo.config". This file is analyzed using the GNU parsers Flex and Bison. The possible applications are shown on the left side of the window of xgeci. Clicking an application produces a new dialog box (see figure 11). You can specify there on the one hand the machine on which you want a run of the application and on the other hand the display where you want to visualize it.
The different machines and displays proposed by CalICo are those which have been declared in the configuration file. Notice that adding a few lines in this file allows any application to be activated by CalICo. A part of a CalICo config file is shown in figure 10.

[4] A polyominoe is a *parallelogram polyominoe* if it can be defined by two paths having elementary steps only north or east and having same first point and same last point.

```
APPLICATIONS {
        Nom Interne : "Xpermut"
        Communication : "OUI"
        Fonction : "IGLance_appli"
        Automatique : "NON"
        Machine : "claudia"
        Display : "audrey"
        Arguments : "-pipes","__ID_PIPES__" }
GROUPE MACHINES HOTES {
        "Combinatoire" : "claudia","nastasia"
        "Calculatoire" : "hector","labri"}
GROUPE MACHINES DISPLAY {
       "Visualisation" : "mylene","sharon","audrey"}
CHEMINS {
        Machines : "labri" {
            "Xpermut" : "/home/labri/calico/Ateliers/xpermut"}
        Machines : "claudia","nastasia","hector" {
            "Xpermut" : "XPermut" } }
```

Figure 10: Part of a CalICo config file

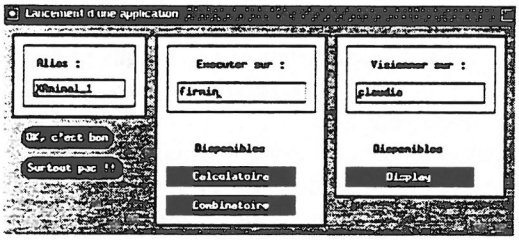

Figure 11. Launching applications from CalICo.

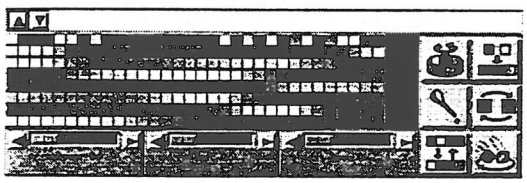

Figure 12. A colormap and its colormap tools.

The main graphical interface *xgeci* allows also one to set the communication channels between all these workshops. The communication tools are shown on the right side of *xgeci* window. Once you have ran applications, you can specify how these applications will communicate by clicking on the single/double/destroy icon and then clicking on the source/target application (see figure 1 page 2). Notice that you can add or remove applications or communication channels at every moment of your CalICo session.

6 Visualization and manipulation workshops.

The implementations of the workshops are unified using an oriented-object approach. We have developed a model for the visualization workshop, called *xmodel* [5]. By now, there are visualization workshops for polyominoes and animals (*xanimal*), permutations (*xpermut*), 2D heap of pieces (*xempil*), graphs (*xgraph*), paths (*xchemin*) and trees (*xarbre*). They all are xmodel-based interfaces.
Actions such as saving/recovering objects, copying/pasting/cutting an object, etc ... are independent from the nature of objects. These actions are accessible from the main window which is common to all visualization workshops (see figure 4 page 3). The common window also allows one to deal with the colourmaps of the visualization workshops. Moreover, the connection tools (send and receive) are also handled by xmodel. Concerning the menus, xmodel uses a language file which contains the name used in the different menus of the visualization workshops.

- **Main interface menus**

The different menus of the main window of the visualization interfaces (figure 4) allows basic operations on the combinatorial objects. In the menu *Fichier* one can found the functions *Nouveau* for creating a new drawing area, *Charger, Enregistrer* for loading/saving objects, *Imprimer* for printing. In the menu *Edition*, are the copy/past functions. The Painting Icon allows to open the colourmap and its tools.

- **Colormaps and colormap tools** (see figure 12). One important feature of CalICo is the use of colours to highlight properties of combinatorial objects. Every visualization workshop comes with a colourmap and colourmap tools. The colourmap is a set of 256 small coloured squares, each colour corresponds to an integer from 0 to 255. Using the colourmap tools allows one to transform the default colourmap, save or load a colourmap. The window of the colourmap tools is shown figure 12. For instance, one can swap two colours of the colourmap execute a circular permutation between several colours, generate a range of hues, create a brand new colour using RVB.

[5] xmodel has been developed using C++ and the Motif library.

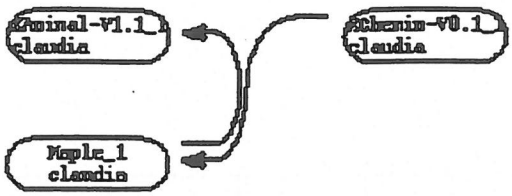

Figure 13: Configuration for the bijection Dyck to parallelogram

• **Toolbox.** Each specific visualization workshop has its own toolbox. It allows to draw objects with the "mouse" and to modify them (symmetries and their planar transformations for instance). As an example, the animal/polyominoes toolbox is shown on figure 3.
Geometric constructions or transformations and algebraic operations are performed on the object(s) using buttons of the toolbox. Each object lives in its own window; these are then itemized in the workshop's main window for special purposes (such as selection).
• **Mailbox** A mailbox is included in each visualization workshop. When invoking the mailbox, selected objects are sent through a channel that was previously established. More precisely, the workshop sends the formal coding of an object in the communication channel defined at *xgeci* level. When a visualization workshop receives a packet, a message is printed in the message window. Clicking the mail box makes the object appear as defined in the preference file (the new object can appear in a new set of objects or in the current set of objects).

7 Random generation workshops.

Random generation methods have several applications particularly in providing samples of data for evaluation and tests of software and algorithms. Coupling random generation workshops with visualization tools increases considerably the interest of such tools. At the moment, CalICo environment includes two random generation workshops. The first one concerns directed animals. Another one deals with several types of plane paths, each one based on a specific algorithm (see for instance [BPS92, Den93, Wil77]).

8 Animating bijection under CalICo

CalICo offers more than an environment for experimentation on combinatorial objects. Its conception was done in such a way that the user does not need to know the primitive drawing of an Xwindow. So Show-me function are available and takes as argument one of the usual mathematical coding of the object. For example, [1,4,5,7,3,2,6] for a permutation, adjency list for a graph, list of cells for the polyomino. Using maple, the user can easily implemante algorithm and animating them. For this, just he needs to anotate his Maple program with call to a Show-me function. If the user need first to receive an object similarly he can use a Give-me function which allow to handle an object (coming from a graphical interface) in a maple program. We describe briefly this functionality using one of the usual bijection between parallelogram polyominoes and Dyck paths [DV84]. First we need to write out a procedure dyck-to-parrallelogram which implemant the bijection between a Dyck path (list of letters in 0,1) in a set of cells (list of (x,y) with x and y integers). The squeletton of the procedure is

```
dyck-to-parrallelogram:=proc().....
.......read(the-path);
.....gluing process.....
end;
```

The modification on this program will be the following

```
dyck-to-parrallelogram:=proc().....
.......Give-me(the-path);
.....gluing process; Show-me(the-cells).....
end;
```

Then first we need to configure calICo, launching maple, Xanimal, Xchemin figure 13. We need to draw a path in Xchemin see figure 14 and send it using the mailbox of the Xchemin application. Reading and runing the procedure dyck-to-parrallelogram into the maple session will produce the sequence of the figure 15.

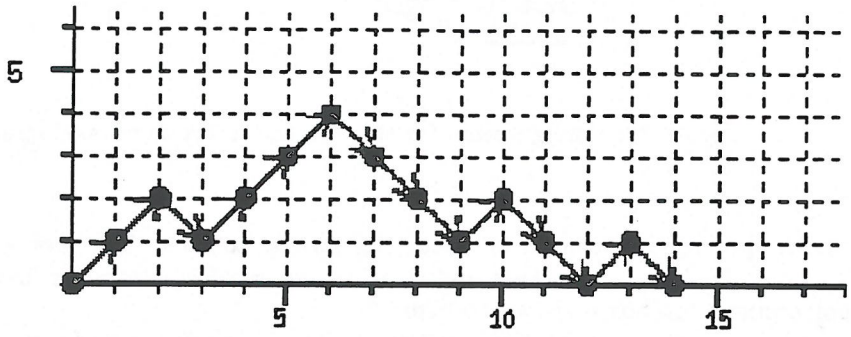

Figure 14: An example of Dyck path.

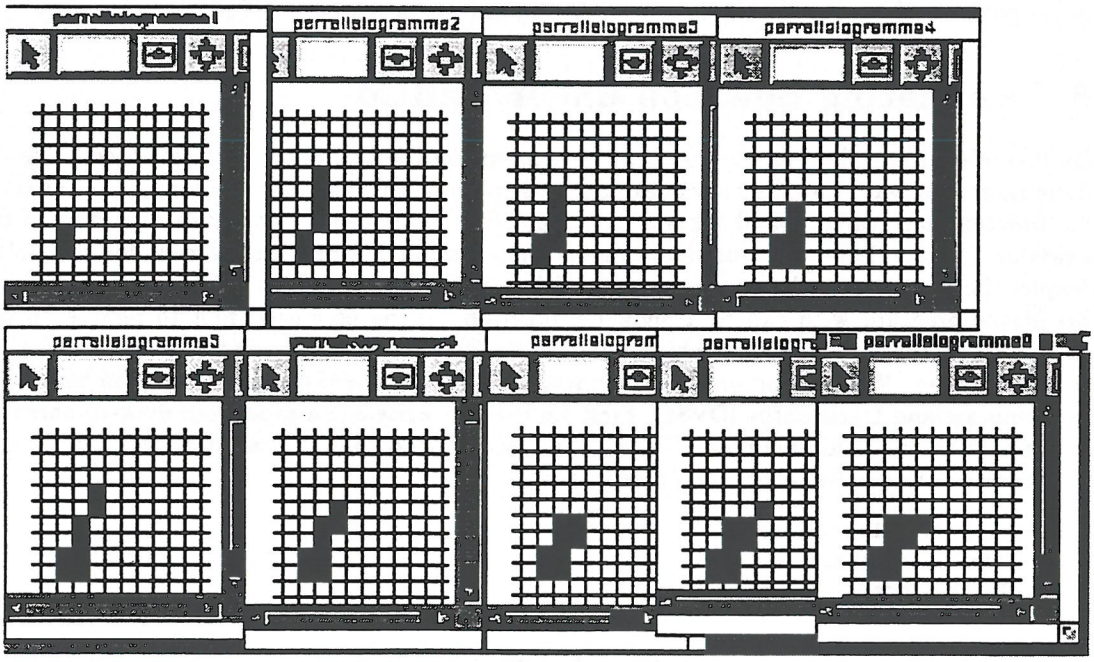

Figure 15: The animation for the previous Dyck path.

Memory in Ko	application
68	`geci`
576	`xgeci`
548	`xpolyo`
452	`xtresse`
616	`xpermut`
76	`geca`
344	`mapleV`
580	`xg-polyo`
3260	CalICo environment

Figure 16: Allocated memory for a CalICo session

9 Availability

An experimental version of CalICo can be obtained by mail at calico@labri.u-bordeaux.fr.
CalICo runs on workstation Sparc, under UNIX system, with an environment X-Window (X11 R4 or more).
GeCI, the interactive communication administrator GeCI in its version 1.1 has been carried in collaboration with the members of the project META2 of INRIA-Rocquencourt on the following equipments : Sparc SUN4, HP 9000, IBM RS 6000, DEC ALPHA, DEC MIPS, PC with LINUX.
The installation needs 8 Mo disk space.
The tutor needs the computer algebra software `Maple` (version ++V.2 or more).

It is not necessary to have an important central memory or a large zone of swap for an ordinary utilisation. We give, figure 16, the memory used for a session of five workshops,

10 Conclusion

As a conclusion let us do a brief abstract of the software we have presented in this paper: CalICo, which means "Calcul et Image en Combinatoire" (that is *Computation and Image in Combinatorics*). We have shown that CalICo is a system intended for combinatorial researchers. Thanks to GeCI, in a distributed environment the user handles combinatorial objects, both in a graphical and mathematical way. Having a consistent environment that integrates graphical programs, symbolic computation programs and a knowledge-based system, CalICo is a powerful tool which helps users to gain better insight into combinatorial problems. Graphical programs offer the possibility to draw objects and symbolic computation programs allow to work on their associated formal coding. A special tool, the tutor, written in Maple, classifies mathematical objects according to their combinatorial properties. In this approach, properties are automatically collected for each manipulated object. The colour is a main component in CalICo: it is the visual support that transcribes the computed statistics on objects. Of course the main remark is that the number of workshops is small and specially oriented on the objects that we are studying frequently in Bordeaux. Thus a future works would be mainly now to implement new workshops using the Xmodel generator. This can be made by the users themselves using the Xmodel tool. Recently experimentations using CalICo have begun and new asymptotic results has been predicted for polyominoes.

References

[AEJV89] D. Arques, G. Eyrolles, N. Janey, and X. Viennot. Combinatorial analysis of ramified patterns and computer imagery of trees. In *SIGGRAPH'89*, Boston, USA, August 1989. Computer Graphics.

[BC88] F. Bergeron and G. Cartier. Darwin: Computer algebra and enumerative combinatorics. *Springer lecture notes in Comp. Sc.*, 294:393–394, 1988. Proc. STACS-88.

[BLK92] M. Beaudoin-Lafon and A. Karsenty. Transparency and awareness in a real-time groupware system. In *ACM Symposium on User Interface Software and Technology UIST'92*, Monterey, CA, Novembre 1992.

[BPS92] E. Barcucci, R. Pinzani, and R. Sprugnoli. Génération aléatoire de chemins sous-diagonaux. In C. Reutenauer P. Leroux, editor, *Actes du $4^{ème}$ Colloque Séries Formelles et Combinatoire Algébrique, publi LaCIM 11*, Université du Québec à Montréal, 1992.

[CGG+92] B. Char, K. Geddes, G. Gonnet, B. Leong, M. Monogan, and S. Watt. *First Leaves: A Tutorial Introduction to Maple V.* Waterloo Maple Publishing, Spinger–Verlag, 1992.

[Del95] M. Delest. Algebraic languages: a bridge between combinatorics and computer science, 1995. To appear.

[Den93] A. Denise. Génération aléatoire et uniforme de mots. In A. Barlotti, M. Delest, and R. Pinzani, editors, *Actes du 5ème Colloque Séries Formelles et Combinatoire Algébrique,1993*, pages 153–164. Université de Florence, 1993. To appear in Discrete Mathematics.

[DF94] I. Dutour and J.M. Fédou. Grammaire d'objets. Technical Report 963-94, LaBRI, Université de Bordeaux I, 1994.

[DFMR96] M. Delest, J.M. FEDOU, G. MELANCON, and N. ROUILLON. *Human Interaction for Symbolic Computation*, chapter Computation and Images in Combinatorics, To appear. Texts and Monographs in Symbolic Computation. Springer-Verlag, 1996.

[DR92] A. Denise and N. Rouillon. Génération de structures arborescentes. In M Lucas, editor, *Actes des 5imejournées GROPLAN*, pages 83–90, Nantes, France, 1992.

[DV84] M. Delest and X. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 34:169–206, 1984.

[FSZ88] P. Flajolet, B. Salvy, and P. Zimmermann. $\Lambda_T\Omega$: an assistant algorithms analyser. In *AECC*, volume 6, Rome, Italy, Juillet 1988.

[FZC94] P. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132(1-2):1–35, 1994.

[GBD+93] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. PVM 3 users's guide and reference manual. Technical report, Oak Ridge National Maboratory, Tennessee, May 1993.

[GGR94] C. Gomez, M. Goursat, and N. Rouillon. GeCI + SCILab + CalICo + Metanet + ··· ou comment faire communiquer des outils existants. In *Journées du GDR Programmation*, Lille, Septembre 1994.

[Rou94] N. Rouillon. *Calcul et Image en Combinatoire.* PhD thesis, Université Bordeaux I, Décembre 1994.

[Ski90] S. Skiena. *Implementing discrete Mathematics : Combinatorics and graph theory with Mathematica.* Addison-Wesley, 1990.

[Sym84] Symbolics. *Macsyma reference manual.* Symbolics Inc., 3rd edition, 1984. version 10.

[Vie86] G. Viennot. Heaps of pieces i: Basic definitions and combinatorial lemmas. *Lectures Notes in Maths, Combinatoire Enumérative eds G. Labelle, P. Leroux*, 1234:210–245, 1986.

[Vie92] X. Viennot. A survey of polyominoes enumeration. In P. Leroux and C. Reutenauer, editors, *4ème Colloque Séries Formelles et Combinatoire Algébrique*, pages 399–420, Université du Québec à Montréal, 1992. Publications du LaCIM 11.

[Wil77] H.S. Wilf. A unified setting for sequencing, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, 24:281–291, 1977.

[Wol88] S. Wolfram. *Mathematica, A System for Doing Mathematics by Computer.* Addison-Wesley Publishing Company, 1988.

[Zim94] P. Zimmermann. Gaïa: a package for the random generation of combinatorial structures. *Maple Technical Newsletter*, 1(1):38–46, 1994.

11 ANNEX

The following pictures are some view of CalICo interfaces. All where developped using Xmodel by students under the direction of searchers. We thank them : I. Dutour for Xanimal (figure 17), N. Hanusse for Xarbre (figure 18), A. Denise for Xchemin (figure 19), Y. Chiricota and G. Melacon for Xempil (figure 20), O. Baudon for Xgraphe (figure 21), G. Melancon for Xpermut (figure 22).

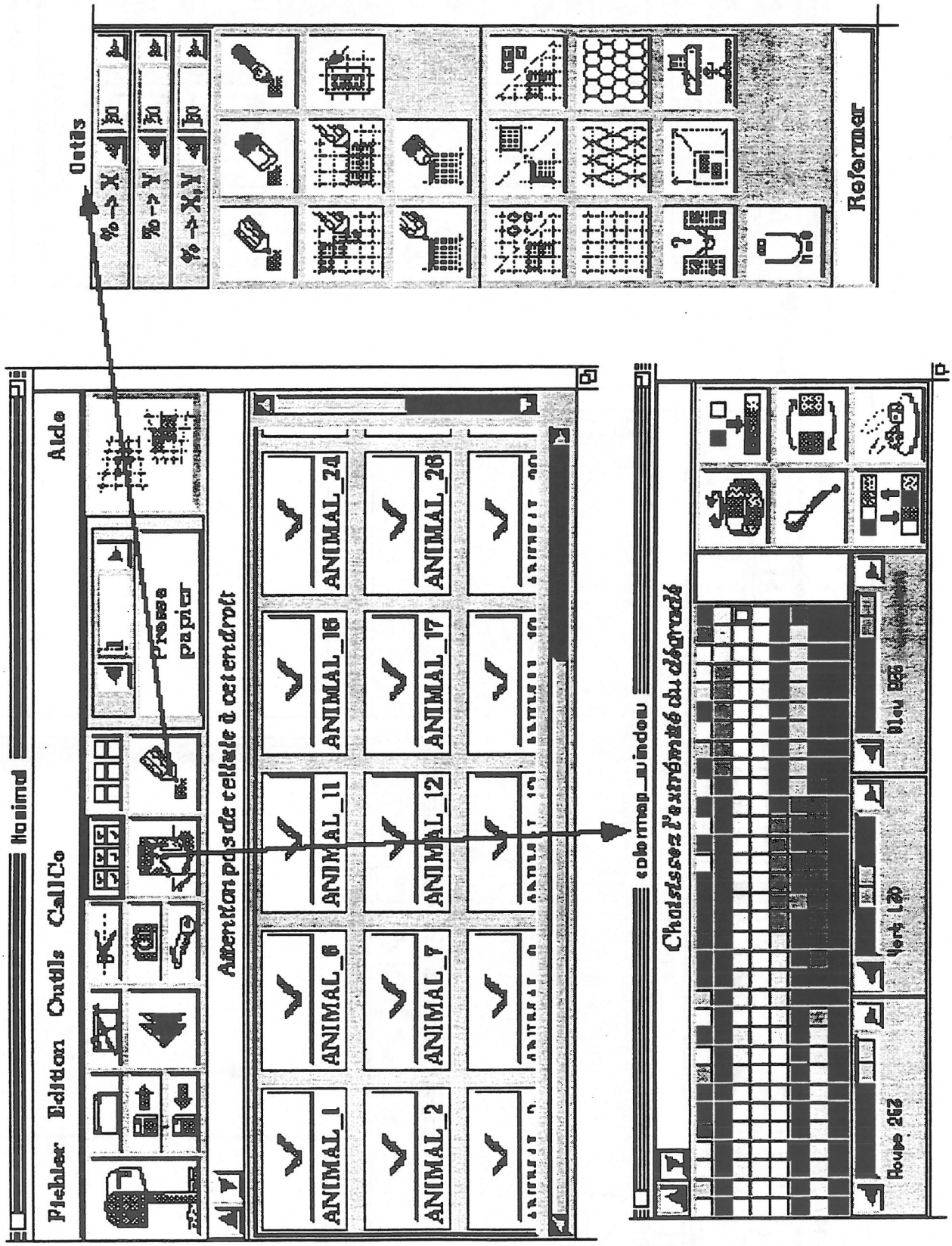

Figure 17: Graphical interface for animals

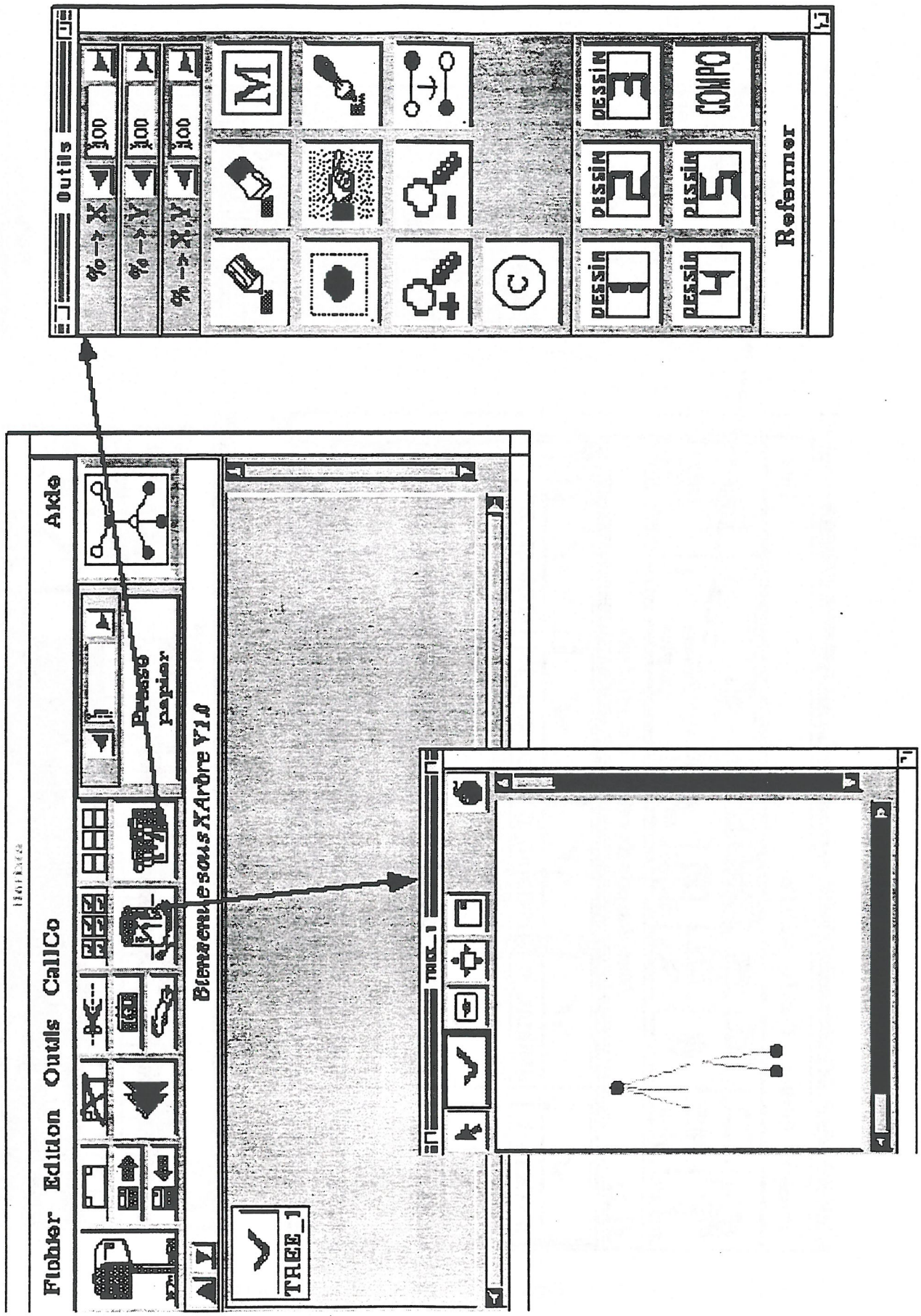

Figure 18: Graphical interface for trees

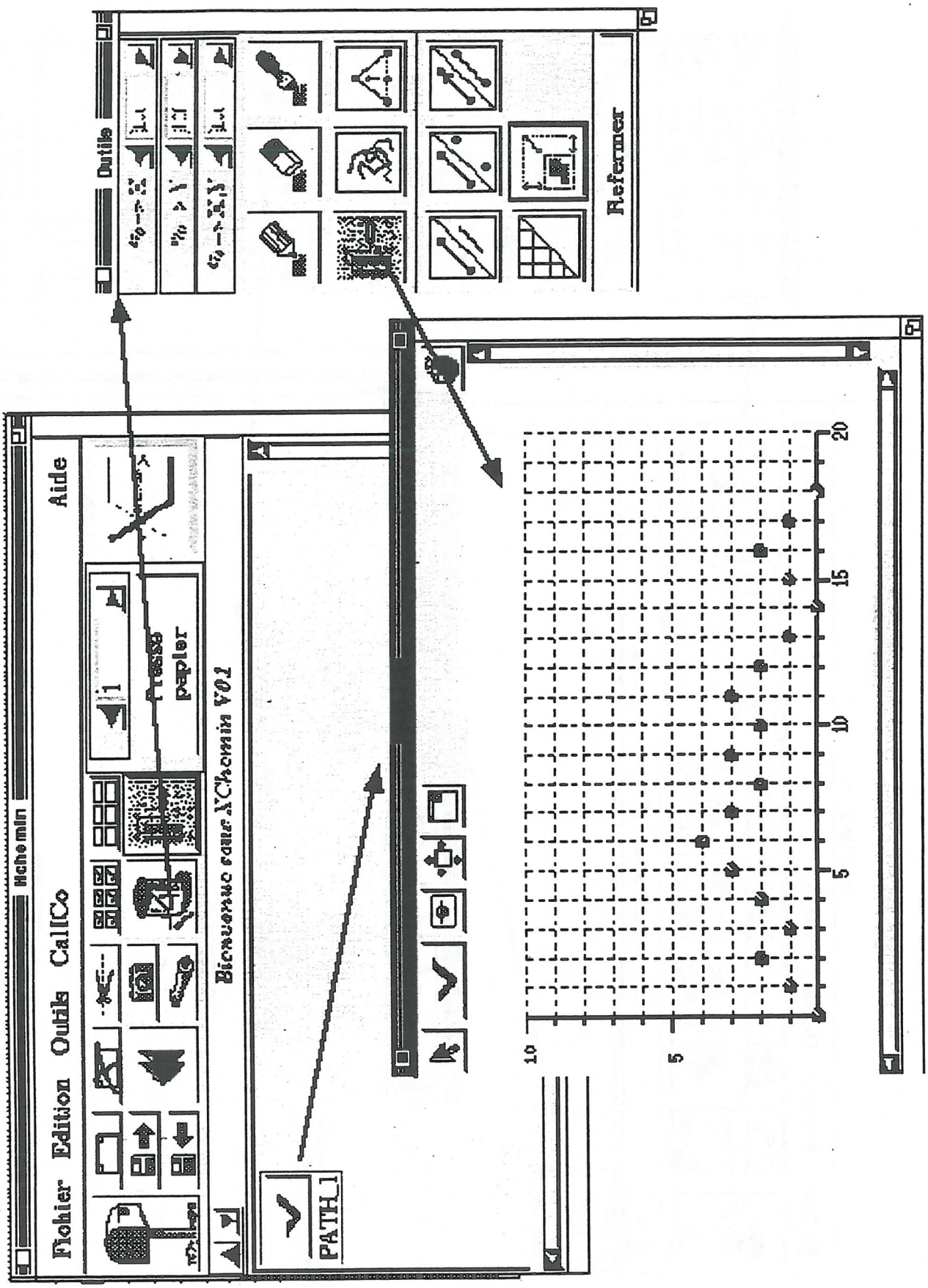

Figure 19: Graphical interface for paths

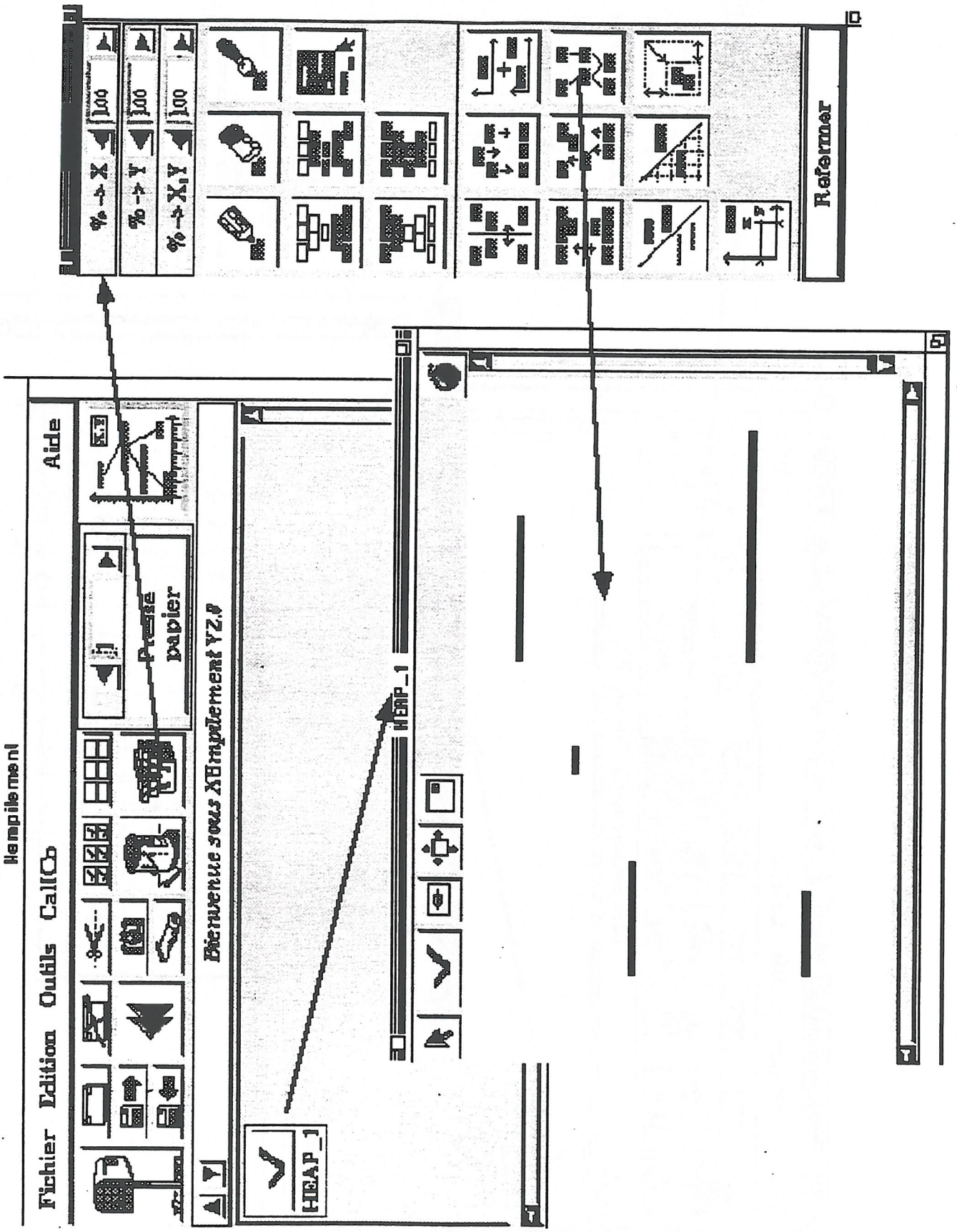

Figure 20: Graphical interface for 2D heaps

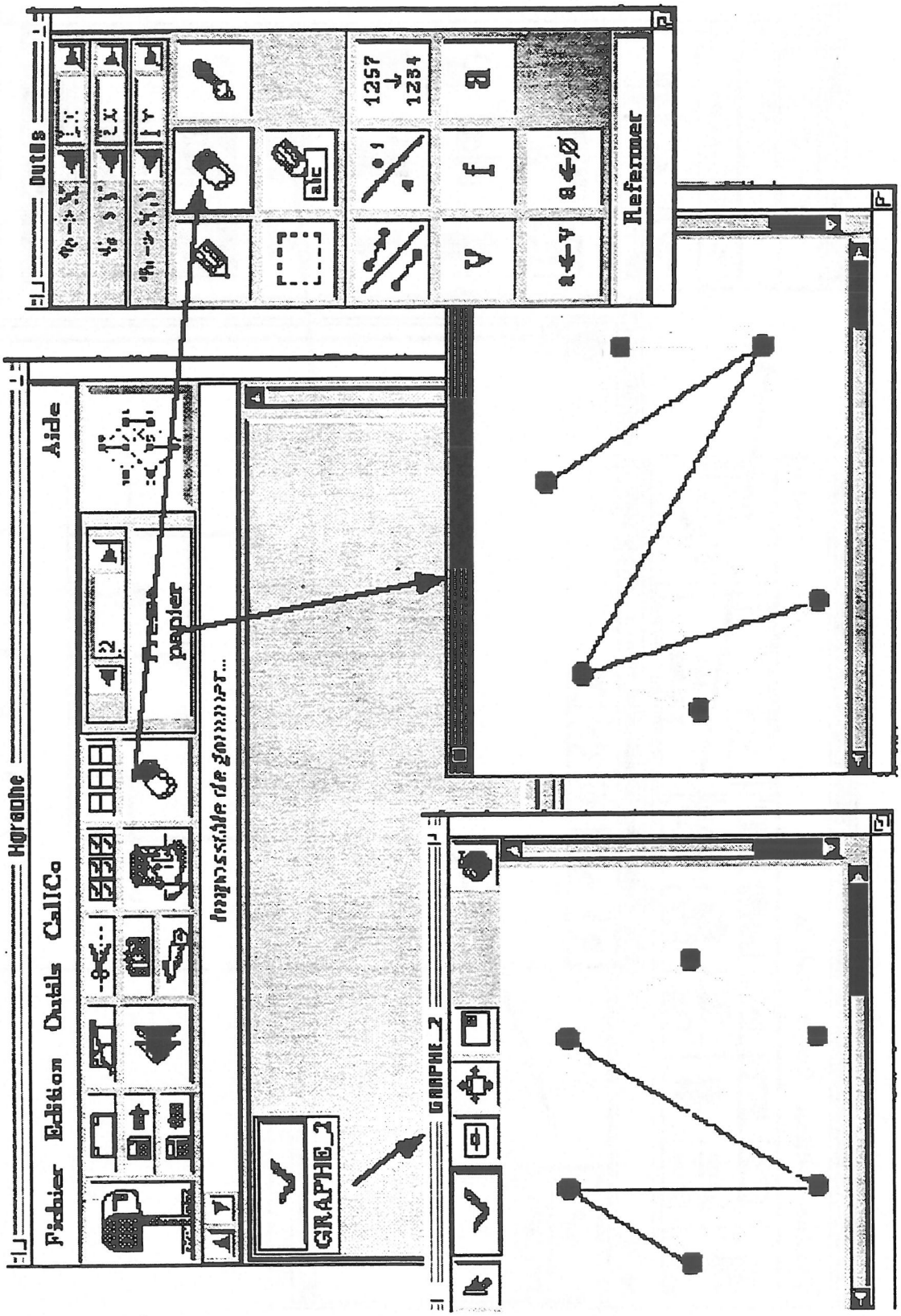

Figure 21: Graphical interface for graphs

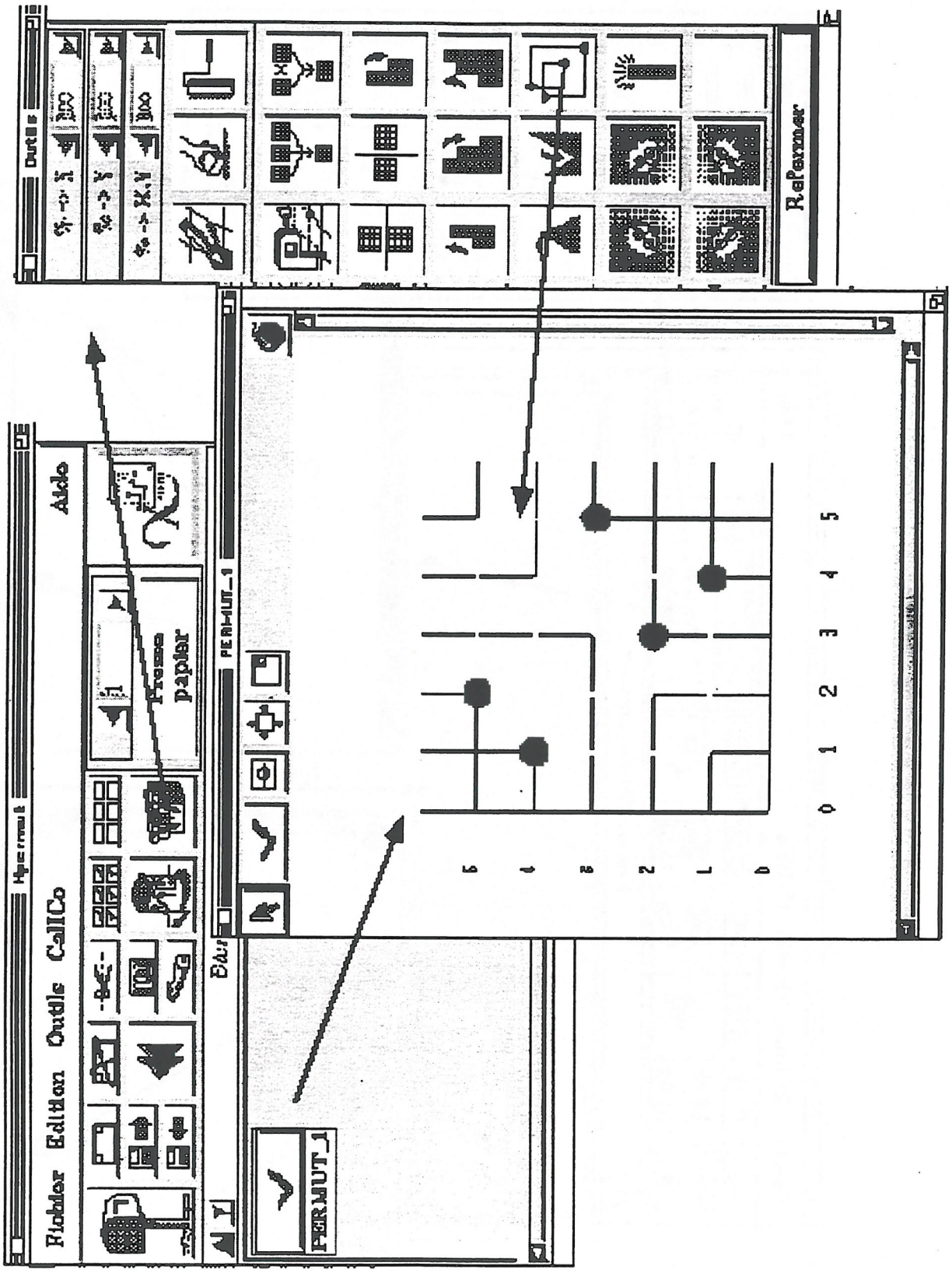

Figure 22: Graphical interface for permutations