Combien? a Software to Teach Students How to Solve Combinatorics Exercises

Françoise Le Calvez, Hélène Giroire, Jacques Duma, Gérard Tisseau, Marie Urtasun

Combien? a Software to Teach Students How to Solve Combinatorics Exercises

Françoise LE CALVEZ*, Hélène GIROIRE**, Jacques DUMA***,
Gérard TISSEAU**, Marie URTASUN*
*CRIP5, Université René DESCARTES, UFR de Mathématique et Informatique,
45 rue des Saints Pères, 75270 Paris Cedex 06,
** Equipe SysDeF - LIP6,Université Paris6, Boîte 169, Tour 46-0 2° étage
4 Place Jussieu, 75252 Paris Cedex 05, France
*** Lycée technique Jacquard,2 rue Bouret, 75019 Paris, France

Abstract : In the Combien? (How Many?) project, we built pedagogical interfaces to help students to learn combinatorics. In this paper, we situate combinatorics teaching in the French curriculum and define a pedagogical objective. Then, we present the solving method on which the interfaces are based. Combinatorics problems are classified according to their solving schemata. Each interface corresponds to a class of problems. It allows the student to build a solution and detects the errors incrementally. Then, we show the pedagogical progression inherent to the use of these interfaces. Finally we describe the experiments in different contexts (learners, teachers).

Keywords : Learning environment, pedagogical interface, modelling, problem solving, error detection, experiment, combinatorics.

Introduction

The objective of the "Combien?" group is to define methodology for use in designing the various components of an ITS. This project involves building a pedagogical system to help students to learn combinatorics using mathematical language. The aim is not so much to turn the students into counting experts, able to determine the number of elements of a set, than to train them for a modelling task and to make them able to represent a situation by a complex structure. We think that counting problems are a good starting point for this objective and that similar processes can be found in other domains like probabilities and algorithmics.

From our experience of teaching combinatorics in the classroom, we have defined the mathematical bases of a solving method corresponding to the typical answer given by students, which we call "the constructive method" in the rest of this paper. It is adapted to the usual student's conceptions and gives access to the mathematical theory of the domain [1]. We have defined a classification of the domain problems and solution schemata associated with the different classes. We have introduced, for each class a "solution-building machine". For the student, each machine is a pedagogical interface which leads him to build a solution to an exercise of the given class. We present then various experiments realised with different types of students (from secondary schools, universities, etc.).

1.   The Combien? Project

1.1.   Pedagogical Objectives

Defining a pedagogical objective is not an easy task. This can be seen in the particular evolution of the mathematical programme as it has been formulated over the years for French secondary schools. Thirty years ago, only contents were described. Then, little by little comments appeared on what the student is expected to know at the end of the year (or semester) as well as instructions for practical sessions. This slow evolution covers three aspects: main objectives (motivations, problematics), contents, and how to do it (examples, links to be illustrated, etc.)

Now official documents are published [2] explaining how to develop the know-how that the student should acquire (see to it that the learning be progressive and active, questions and examples should be systematically used, attend to the quality of the reasoning of the student etc.).

In the Combien? system, the objective perceptible by the student is to acquire a satisfying competence for solving combinatorics exercises of a certain type. However, this is not the main objective: in fact the point is to succeed in

-    mastering the associated concepts,
-    building efficient mental representations,
-    formulating reasonings using basic notions of logic and set theory,
-    getting familiar with conceptual modelling and programming.

It is in fact a large and long-term objective which is much more difficult to evaluate than a well targeted short-term one

1.2.   Special Features of Combinatorics Problems

The domain of discrete mathematics is specific as far as the procedures of proof and modelling is concerned [3]. In many other domains of mathematics, solving problems consists in using inference rules to get new facts or in rewriting rules to transform an expression. In combinatorics we start with a constrained system and we work with a set of objects (called *configurations*) verifying those constraints in order to calculate their numbers. The main difficulty is to find a suitable representation of the problem and an appropriate modelling of the solution.

The exercises that Combien? proposes to the student, are taken from the first course on combinatorics, as it is taught in the last year of secondary school in France. Here is an example of a combinatorics problem: with a pack of 32 playing cards, how many five-card hands with exactly two spades and two hearts is it possible to form? The combinatorics problems that we consider all have an analogous form: given a set or sets (here, the pack of cards), count within some "universe" of configurations (here, the set of all possible five-card hands) the elements satisfying some constraints (here, including two spades and two hearts). Rigorously solving this kind of problem requires an appropriate, often abstract and complex representation using the concepts of set theory (e.g. sets, mappings, sets of sets). For a beginner, this approach is inaccessible because the representation is very different from the usual mental image.

However, some students are able to solve these problems (see [4] for a study of the combinatorial capacity in children and adolescents). For example, if they were asked: "how many five-letter words are there with exactly two occurrences of the letter A?", their answer would be something like: "I first select two positions for the A, which gives me 10 possibilities, then I complete the remaining letters, which gives me $25 \times 25 \times 25$ possibilities (15625). So there are 156250 words satisfying the requirements". This is correct, but such an answer is not a real mathematical proof, and students are generally unable to justify every aspect of their answer. Moreover, the apparent simplicity of this answer hides the fact that it is difficult to find: many students do not manage to solve the problem.

2.   Our Procedure

First, we defined the mathematical bases of a solving method "the constructive method"; then we worked out a classification of combinatorics problems according to their solution; finally we built some interfaces, the "machines", to allow the student to build his solution.

2.1.   The Constructive Method

To calculate the cardinal of the set of the configurations which satisfy the constraints given in the wording, it is possible to do so without enumerating the configurations but simply by reasoning on a description of the set of the configurations. Actually, the list of the elements can be given as the result of an enumeration algorithm. So, an efficient method for solving combinatorics problems consists in explicating this algorithm and then analysing it in order to predict how many elements it will generate, without having to execute it. It is this type of reasoning that we would like the student to follow.

It is often used by students and in textbooks, but in an informal and implicit way that may produce some difficulties. Thus, we defined the mathematical bases of such a method which we called the «constructive method». This method has the advantage that it allows the formulation of a rigorous proof of the solutions, but it demands a modelling of the problem and uses mathematical concepts which are not familiar to the students involved. It is adapted to the usual student's conceptions and gives access to the mathematical theory of the domain [1].

2.2.   Problem and Interface Classification

Observation of experts of the domain at work reveals that they know classes of classical problems and that they know how to link to them valid schemes of constructive one-to-one definitions. They do this by determining the problem class through the analysis of its wording, and then instantiating the associated scheme in order to generate the equivalent constructive definition. The advantage of this method is that, when applicable, it guarantees the validity of the solutions thus obtained.

We worked out a classification of combinatorics problems bearing in mind their solution [1]. To each class we associated a machine to build a solution. The reasoning on the various steps of the construction makes it possible to calculate the number of elements of the set to enumerate. All the machines have the same general aspect: the important idea is that the class and the method are embedded in the look and the inner workings of the machine. We assumed that it would help the student to fully integrate the method into his field of competence, and the experiments have indeed proved that this is the case.

2.3.   Pedagogical Progression

The machines are intended to be used in quasi self training. A user handbook is associated with each machine. The corresponding contextual help is proposed for each step of the construction of the solution.

The student has at his disposal the machines corresponding to each class. Each machine contains appropriate exercises. Thus, the student can learn to recognize the class of each exercise. This recognition is not so easy. The exercise class is not immediately recognisable from the wording of the exercise. For example, the two exercises: "From a group of 10 persons, how many ways are there to choose a president, a treasurer and two secretaries?" and "How many ten-letters words can be built from the set {p, t, s, r} with 1 p, 1 t, and 2 s?" belong to the same class "ListBuilding". When the student learns to recognize the class of the problem, he uses a general machine in which he will specify what particular machine he wants to use to solve the problem.

3. The Work Done

3.1. Tools Defined and Machines Built

We have defined and constructed an interface editor, called EDIREC, to build machines which are specified by using interactors. By means of EDIREC, we have built the machines associated with the four problem classes which we named "SetBuilding", "ListBuilding", "CaseSetBuilding" and "CaseListBuilding". In section 1.2, the exercise about card hands is an example of the SetBuilding class, and second one is an exercise of the ListBuilding class.

As said above, all the machines were built according to the same pattern and have the same appearance. Figure 1 represents a snapshot of the SetBuildingMachine during the elaboration by a student of the solution of an exercise of this class.
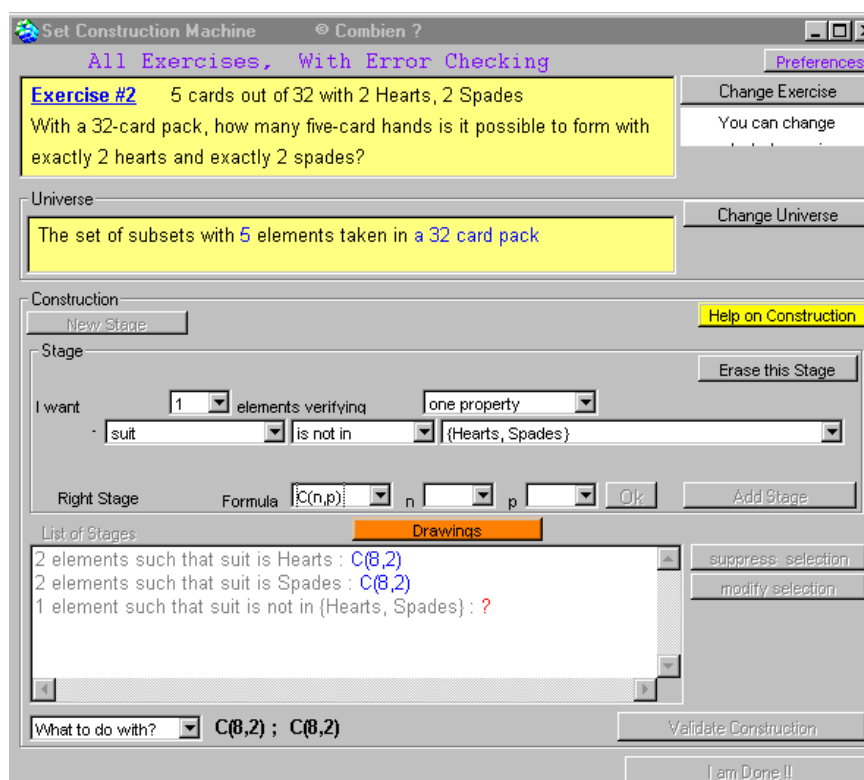

Figure 1

Let us give an example of an exercise of the CaseSetBuilding class, a class whose exercises divide into several exercises of other classes: "With a 32-card pack, how many five-card hands is it possible to form with exactly 3 queens and exactly 2 hearts?". This exercise divides into exercise E1 for which each solution-configuration does not contain the queen of hearts, and exercise E2 for which each solution-configuration contains the queen of hearts. The set of the solution-configurations SC (five-card hands satisfying the constraints) is the union of the set of the solution-configurations of E1 (SC1) and the set of those of E2 (SC2). As those two sets are disjoint, the cardinal of SC is equal to the sum of the cardinal of SC1 and the cardinal of SC2. The principle of addition, used in the teaching of combinatorics, has been applied. The two cases need to be solved separately as they both belong to the SetBuilding class. The CaseListBuilding class is similar to the CaseSetBuilding class but its exercises divide in two ListBuilding class problems. Here is an example: "how many five-letter words are there with exactly three vowels and at least one occurrence of the letter A"?

Construction of the Solution

We have been particularly careful to define our building machines as pedagogical interfaces which guide the student in the construction of his solution. He defines the universe and then, step by step,

the constraints. Through the interface, the student edits his solution in the form of a tree-like structure. In fact, the presentation hides the details of the formalism. Each node of the tree is valued by an object of the conceptual model of the domain that we have defined [5]. At each input the machine builds a tree-like sub-structure and adds it to the structure under construction. At this very moment the machine searches for the possible student errors by using an incremental error detection model that is described in [6].

Error detection

In a pedagogical interface, it is better to let the student give his solution and make errors, but only the errors that are interesting from a pedagogical point of view. In the Combien? project we restricted the domain by requiring the student to use a specific solving method which we call the "constructive method". We eliminated a list of uninteresting errors by controlling the interface (dynamically built pulldown menus, enabled or disabled buttons, sequential control…). So, for each machine, we were able to establish a list of possible errors (some errors are common to several machines). We associated an error schema with each type of error. Each schema is defined by a tree-like pattern in which nodes are valued by variables and by a condition binding some of these variables. At each addition of a tree-like sub-structure (a part of the solution built by the student), the machine searches in its database of error schemata, for those that are appropriate in the corresponding context. For each of these error schemata, the machine compares the pattern with the tree-like sub-structure. If it matches, the error condition is tested with the instantiations of the variables bound by the condition. If the test is true, there is an error.

Each machine is associated with a database of error schemata. All the machines are built according to the same model, and have their own error database.

3.2. Experimentations

First, in 2002, we conducted three kinds of experimentations. The target public was different for each of them: students, mathematics teachers and one ergonomist. These experimentations were carried out principally with the SetBuilding machine which contained about ten exercises. The exercise list is in increasing difficulty order.

In 2003, we started a new series of experimentations with the four machines: "SetBuildingMachine", "ListBuildingMachine", "CaseSetBuilding-Machine" and "CaseListBuildingMachine".

For each experiment, the user handbook is available online, but also in paper form. Online, the learner has access to contextual pages using the command "?". In each experimentation, the machines are presented as problem solving tools. The experimentation protocols differed somewhat according to the users. Generally, the machines were used freely (the order and the number of the exercises to be solved was not dictated). We asked several students to express their reasoning aloud and we recorded them while they were using the machines. At the end of each session, each student completed a questionnaire to give his evaluation of the software.

Target public: students from last year of secondary school

In 2002, we carried out two different experimentations with voluntary students. The first 3-student group had no knowledge whatsoever of combinatorics. They rushed to use the interface. They considered it as a discovery game and had many exchanges among themselves. They nearly never used the help tools. They were keen to play, and solved all the exercises although we thought they would solve only a few of them. Then, they asked for other exercises belonging to other classes. So, we shown them the other machines, and when they attended the first combinatorics class, they knew how to solve the SetBuilding exercises and had an idea of how to solve the exercises from the other classes. The second 3-student group worked with the machine after they had five hours of paper solving combinatorics exercises, without having a proper combinatorics lessons. Although, they were less motivated by mathematics than the others, they were faster and enjoyed solving the exercises.

In the two groups, the students begun to focus on the numerical value to be obtained, then they attended to the demonstration that the machine forced them to do. After a time, they showed no

more interest in the numerical value, but focused on the definition of the different steps of the demonstration.

In April 2003, an experimentation the entire final year class will be conducted.

Target public : 2nd year university students of discrete mathematics

Here again, the students showed great interest in this experimentation. They were motivated and solved the exercises rather easily. They appreciated the possibility offered to them to systematize the solution. More than that, some of them have "discovered" that they could use a "method": "Now, I know that there is a systematic way of solving this type of problem" (quotation of a student).

Target public : students in their teacher training year (in mathematics)

Here again, the software was appreciated. Some students found it difficult to follow the method and they were bothered by the constraints of the software; they would have liked to be allowed to solve the exercises differently. However, the students have expressed the desire to use the software by themselves and indicated that they would like be allowed to use it in the classroom.

Target public : mathematics teachers

They experienced more difficulty following the method induced by the machine. Normally, when they teach combinatorics, they do not encourage pupils to systematically use the method. Thus, they have had to change their way of reasoning. For the first problems proposed, the easiest ones, they would have preferred to have given the solution without any justification. However, for the more difficult problems of the same class, they appreciated having to justify each step.

Results

All these users appreciated the look of the interface of the "SetBuilding" machine. They also liked the fact that the other machines were of the same general type. This helps the learner to concentrate on the underlying concepts. They said that this method was interesting because it enabled them to solve problems whose solutions were far from obvious. They solved all the exercises, following the order in which they were given, which was not requested of them. They appreciated the pedagogical progression which enabled them first to become familiar with the machine, and then to concentrate on the difficulties of the problems.

One of the objectives of our approach is to convince the students that they have to "prove" a result, and that using a systematic approach makes it possible to solve more and more difficult problems. Our software has thus proved that it is an efficient tool to make the student justify his numerical results in combinatorics problems.

3.3.   The software

Combien? is developed on the two platforms: MacOS and Windows. The programming language used is Smalltak, in the Visualworks environment.

To describe the various resources, we have defined a specific language which we call DESCRIPT. This makes the modification of resources easier. The resources are the conceptual model, exercises, writing formats, multilanguage texts, configurations, error schemata. Thus, with Descript, we can add some new exercises, and we can easily change the texts used in the "machines". At the moment, the texts are available in French, English and Spanish, and any other language can be added if the translation is provided. With Descript, the modifications or additions to the system are made in a declarative form which is easier to use. The machines can be configured for various users. This means that the user (teacher or indeed the learner) can choose to activate the error detection or not and select what exercises are present in the machine.

As already said, in §3.1, the system uses an environment of construction of interfaces named EDIREC [7]. This programme allows the designer to describe the behaviour of the interfaces in terms of what we call interactors, and then to create the corresponding code.

For the error treatment (error detection and response to the errors - action and messages -), we have defined data bases of error schemata.

At the moment, the system is composed of four kinds of activities which correspond to four classes of problems: "SetBuilding", "ListBuilding", "CaseSetBuilding" and "CaseListBuilding".

Conclusion

The COMBIEN? software sets up a problem situation: it gives the student an immediately understandable aim (answer the question: How Many?). This aim is not directly attainable as it requires several steps of action, modelling and reasoning. There is no systematic algorithm to realise it. Moreover the activity can produce some cognitive conflicts. For example, if in the wording of the proposed problem we find the constraint "a card hand with 3 hearts" and when the student says: "I choose 3 hearts", the construction is false. The process and the solution are not easy, but students feel that they can manage them. And this is indeed true. The nine exercises of the first machine were solved by beginner combinatorics pupils. So the learners are really motivated and actively search for a solution. There are several paths to the solution (a student says "the various valid reasonings to find a solution can be expressed: there is not just one possible way"). To attain the solution, the user must implement and sometimes discover particular processes and knowledge, which is the objective of the learning. This is largely due to the dictated constraints.

Using an informatics environment to realize this situation has some advantages: the environment systematically demands that constraints be respected, which ensure that the user actually learns (the user must use the constructive method and explicitly determine the different steps). The environment provides explicit resources (help, error messages) as well as implicit ones (language used, organisation and verification of the interface). It offers visualisation means (dynamically- and contextually-created examples and counter-examples). It frees the student from several supplementary tasks which are not relevant to the targeted objective (numerical calculations, pretty formats). It prevents non-instructive errors from being made (automatic inhibition of several possibilities, controlled lists of acceptable values).

Each problem situation requires further reinforcement training to be sure that the acquired knowledge is explicit and reused. For the moment, this training is not provided by the system. We plan to ensure this training (for example, asking the user to give an explicit proof of the equivalence between wording and construction).

Evaluation of the system efficiency in terms of learning is very difficult because of the ambitious and long term nature of its objective. Evaluation will have to be done over a long period and should involve the various interfaces, corresponding to the different types of problems, which will be used according to the expected progression

References:

[1] Le Calvez F., Urtasun M. , Tisseau G., Giroire H., Duma J. *Les machines à construire : des interfaces pour apprendre une méthode constructive de dénombrement*. Actes des 5èmes Journées francophones EIAO, pp 49-60, M. Baron, P. Mendelsohn, J.F. Nicaud eds, Hermès, 1997.

[2] Ministère de l'Éducation nationale, Direction de l'Enseignement scolaire, *Mathématiques classe de première des séries générales*, série Accompagnement des programmes, CNDP, 2001.

[3] GRENIER D., PAYAN Ch*., Spécifités de la preuve et de la modélisation en mathématiques discrètes*. Recherches en Didactique des Mathématiques, Vol. 18, n°1, 1998, p. 59-100.

[4] Fischbein E., Gazit A., *The combinatorial solving capacity in children and adolescents*. Zentralblatt für Didaktik der Mathematik, 5, 1988, 193-198.

[5] Tisseau G., Giroire H., Le Calvez F., Urtasun M., and Duma J*., Design principles for a system to teach problem solving by modelling*. Lecture Notes in Computer Science N° 1839, ITS'2000, pp. 393-402, Springer-Verlag, Montréal, 2000.

[6] Giroire H., Le Calvez F., Duma J., Tisseau G., Urtasun M.,*Un mécanisme de détection incrémentale d'erreurs et son application à un logiciel pédagogique*. RFIA 2002, pp1063-1072, Angers.

[7] Tisseau G., Duma J., Giroire H., Le Calvez F., Urtasun M., Spécification du dialogue et génération d'interfaces à l'aide d'interacteurs à réseau de contrôle. Actes de la 11ème Conférence francophone IHM'99 p. 94-101.

Web site: http://www.math-info.univ-paris5.fr/combien