Using APL to build science tutors for the high school level

Manuel Alfonseca

Universidad Autónoma de Madrid

(Manuel.Alfonseca@ii.uam.es)

Abstract

This paper describes the procedure used to build several courses on the sciences for the high school level. An APL2 program has been written that accepts problem models, including explanation models, and uses them to generate many different problems. Each course is provided with about one hundred problem models, from which the student is invited to solve many thousands of different actual problems. The unique features of APL2 have made it very simple to develop the program that supports the courses, which exists in both DOS and Windows versions.

Introduction

Computer-aided education is a flourishing area. Educational multimedia products are announced every day on all conceivable subjects: Mathematics, Physics, Chemistry, Astronomy, Geology, Biology, Medicine, Languages, Grammar, Reading, Writing, Spelling, Drawing, Art, History, Geography, Dictionaries, Encyclopedias, and general information [1].

We have developed a procedure to generate a family of educational multimedia applications, currently applied successfully to Mathematics and Physics at the high school level. These courses are different from the run of the mill, for they do not impart theoretical information, but guide the student to solve a potentially unlimited number of applied problems.

The courses

Each course consists of a number of lessons, each containing five problem models. Every time the program proposes a given lesson, the models are used to generate problems. The definition of each problem model contains random variables, which receive a different value whenever the problem model is used. In this way, a single problem model may give rise to several different actual problems (up to several thousands, in some cases), so that the probability of the same student being invited to solve the same problem again is small.

The following is an example of a problem proposed by the lowest level course on Mathematics:

```
Which is the
5 th term of the arithmetic progression
whose first term is 3
and whose difference is 2?
```

The student is supposed to solve the problem on a piece of paper, using a calculator or any other means, and to type the solution at the keyboard. The program solves the same problem, compares the student solution with its own (11 in the example) and provides feedback. If the given result is incorrect, the student is offered another try. If the new solution is also wrong, the correct way to solve the problem is explained, and a new problem of the

same model is proposed to find out if the pupil has followed the explanations.

In the example, the explanation would be:

```
The Nth term of an arithmetic progression
      is
An = A + D.(N-1)
where A is the first term and D is the
      progression's difference.
Replacing in this formula the values we
      have been given, we get:
 An = 3 + 2 .( 5 - 1 ) = 11
```

Each lesson is a text file containing all the program models, separated by a special line. A program model contains a number of lines with ASCII text, executable instructions, or result definitions. The definition of the program model for the previous example is as follows:

```
XEC A←?10
XEC D←3+?5
XEC N←7+?5
 Which is the
XEC N 'th term of the arithmetic
      progression'
XEC 'whose first term is' A
XEC 'and whose difference is' D '?'
RES A+D×N-1
```

where the lines beginning by XEC contain APL2 instructions that will be executed by the program, the line beginning by RES gives an APL2 expression calculating the result of the problem, and the remaining lines are assumed to be text that will be output as it is. The results of the executed instructions are shown at the screen only if the instruction does not contain any assignment. From this model, the program may generate 250 different actual problems, one of which has been given above.

The explanation model is located after the program model, also separated by a special line. In our example, the explanation model is the following:

```
The Nth term of an arithmetic progression
      is
 An = A + D.(N-1)
where A is the first term and D is
      the progression's difference.
Replacing in this formula the values we
      have been given, we get:
XEC 'An =' A '+' D '.(' N '- 1 ) ='
      (A+D×N-1)
```

The rules to interpret the explanation are the same as indicated for the problem enunciation. In this case, we only have executable lines and simple text lines.

Let us look at a more complicated model, this time chosen from the highest course on Physics:

```
XEC H←?20
XEC (A1 W)←1+?2⍴9
XEC T←.01×?10
XEC R1←W×H
XEC E←A1×R1
XEC R←(×E1)×.1×⌊10×|E1←-E×2○W×T
 A circuit contains a coil with a
      self-induction coefficient
XEC 'of' H 'henrys. The instantaneous
      intensity is'
XEC 'I =' A1 'sin' W 't. Find the value
      of the potential difference'
XEC 'at t =' T 'seconds.'
Use a calculator to solve the
      trigonometric functions.
If you have to give a value to pi, use
      3.14159265
Write the result in volts, without
      specifying the units,
with a single decimal figure, without
      rounding.
RES R
```

The number of different problems that may be generated from this model is 16200. The model provides two different explanations:

```
The reactance (in ohms) of a coil, due to its
      self-induction, is
  R = w.L
where L is the self-induction coefficient of
      the coil, in henrys,
and w is the angular velocity of the
      alternating current through the coil.
In our example:
XEC ' w =' W 'rad/sec'
XEC ' L =' H 'henrys'
Replacing and operating, we get:
XEC ' R =' W '.' H '=' R1 'ohms'
The impedance due to the inductive reactance is
      a positive imaginary number equal to
XEC ' ' R1 'i ohms.'
whose module is the inductive reactance and its
      argument 90 degrees
(pi/2 rad). We are told that the intensity is:
XEC ' I =' A1 'sen' W 't'
The potential difference, then, will be:
  E = Em . sen (wt + fi)
where
XEC ' Em = maximum potential difference =
      Im . |Z| =' A1 '.' R1 '=' E 'volts'
and the phase fi is the argument of the complex
      impedance, i.e. pi/2.
Therefore, the instantaneous potential
      difference is:
XEC ' E =' E 'sin (' W 't + pi/2 ) = -' E 'cos
      (' W 't )'
XEC 'We are requested its value at t =' T
      'seconds.'
Replacing values:
XEC ' E = -' E '. cos (' W '.' T ') =' E1
      'volts.'
```

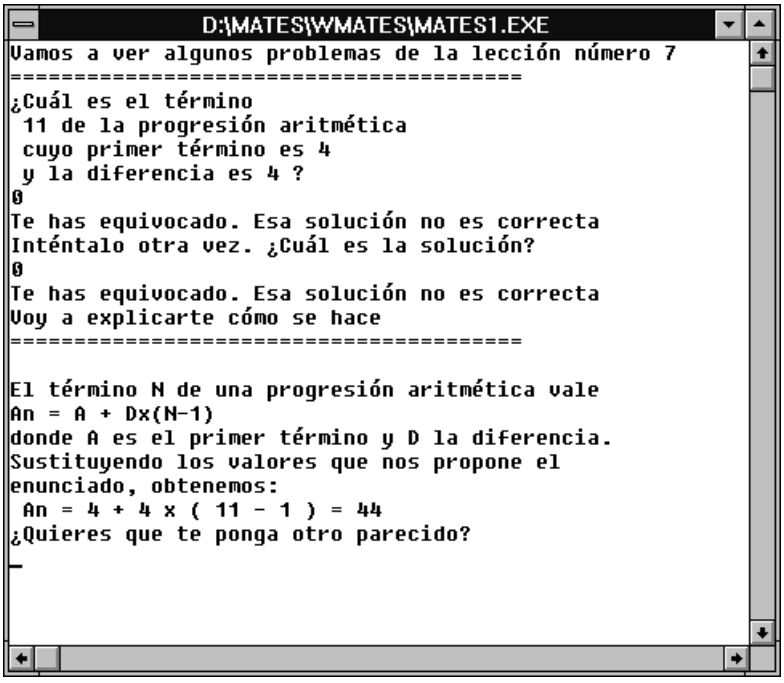

Figure 1: The Windows version of the APL2 educational program

```
. . . . . . . . . . . . . . . . . . . . . . . .
Another way to do it: we know that the
      potential difference across
     a coil through which traverses an
      intensity i is:
  E = - L di/dt
where L is the self-induction coefficient
      of the coil, in henrys,
and di/dt is the derivative of the
      intensity with respect to time.
We are told that the intensity is:
XEC ' I =' A1 'sin' W 't'
Its time derivative is:
XEC ' I =' A1 '.' W '. cos' W 't ='
      (A1×W) 'cos' W 't'
Therefore
XEC ' E = -' H '.' (A1×W) '. cos' W 't =
      -' E 'cos' W 't'
XEC 'We are requested its value at t =' T
      'seconds.'
XEC ' E = -' E '. cos (' W '.' T ') =' E1
      'volts.'
The same we got with the other method.
We must give the result with a single
      decimal figure, without rounding.
Therefore, we shall write:
XEC R
```

The educational APL2 program

A simple APL2 program (385 instructions in all), packaged with the interpreter as an executable PC program, executes five different application courses:

- Three courses on Mathematics
- Two courses on Physics

Several unique features of APL2 have been used to develop these courses:

- The packaged program includes the interpreter, which makes it possible to use the execute primitive to instanciate the problem models.
- Many problem models use the function fixing system function to generate new functions, thus expanding their possibilities enormously. In fact, one of the models denerates three APL2 functions and uses them both during the enunciation of the problem and in the explanation.
- The two APL random generators (monadic and dyadic ?) are used to obtain many different specific problems from a single problem model.

All the five courses exist in DOS and Windows 3.1 versions. The Windows versions (see figure 1) are partially written in C++ [2] and partially in APL2. A procedure has been developed to make it possible to call APL2 programs from C++ programs. The whole interpreter is included in a library compatible with the Borland C++ compilers.

The library interface is defined by the following apl2.h file:

```
   typedef struct _APL2Header {
       unsigned int ptr;
       unsigned int dsize;
       unsigned int nelm;
       unsigned char type;
       unsigned char rank;
       unsigned int dim1;
       } APL2Header;

 short APL2 (char *input, APL2Header
**output);
 short APL2F (unsigned int Ptr, int code,
                    APL2Header **output);
 void APL2gets (unsigned char *buf);
 void APL2print (APL2Header *salida, char
                    separator);
 void APL2more (void);
```

The main function, APL2, sends to the interpreter a character string containing the APL2 instruction to be executed and is returned (in variable **output**) the result of the computation, with the structure indicated by the APL2Header type (the standard APL2/PC structure of APL2 variables). The explicit result of the function is a return code, where eventual errors in the APL2 instruction to be executed are passed back.

The APL2F function receives an APL2 reference and returns the value of the APL2 object represented by the reference. With its use, the C++ program may access recursively the different levels and components of a general array.

The APL2gets subroutine performs an APL2 quote-quad read on a string buffer. The APL2print subroutine performs an APL2 quote-quad write on the screen of the APL2 object passed as its input. The APL2more subroutine manages the orderly apparition on the screen of results that would have too many lines.

The APL2 interpreter in the library contains a small workspace (about 40 kilobytes), sufficient for the execution of this application. The line sent to the interpreter may also be a system command such as)IN or)OUT, which makes it possible to partitionate larger workspaces to fit in the available space.

The program keeps information about the performance of each student on the different lessons. This information is used by the program the next time the student decides to tackle a lesson, and may also be obtained by the teacher. The student record on a lesson may be improved by repeating it.

Each of the courses contains a printed book [3-7] describing the essential theoretical questions needed to solve all the problem models. In future versions of the courses, these books will be replaced by an on-line help facility written in html.

A graphical approach to educational courses in APL2

A different procedure, which makes extensive use of the graphics auxiliary processor (AP207) provided with APL2, has been used to build a course on Chemistry (inorganic formulation) for DOS [8].

In this case, the workspace contains only the functions embodying the graphical user interface. The 18 lessons making up the course are stored in independent files, each of which contains a single APL2 function programmed to automatically generate all the possible formulae of a given family of inorganic compounds. Whenever the student changes the lesson being studied, the function corresponding to the preceding lesson is erased, and a new function is read from the appropriate disk file.

The user interface may work in three different states:

- Lesson state: the student is tested on inorganic formulation in two different modes (see figure 2):
  - The program proposes a formula and requests the name of the compound.
  - The program proposes a name and requests its formula.

  Three different systems of nomenclature may be tested: the traditional system, the stechiometric system, and the Stokes system. The selected system may be changed at any point.
- Text help state: The text of the book accompanying the program and corresponding to the lesson being studied may be seen at the screen. A special APL2 function is used to show the texts using AP206, in such a way that subindices and superindices in formulae appear correctly (see figure 3).

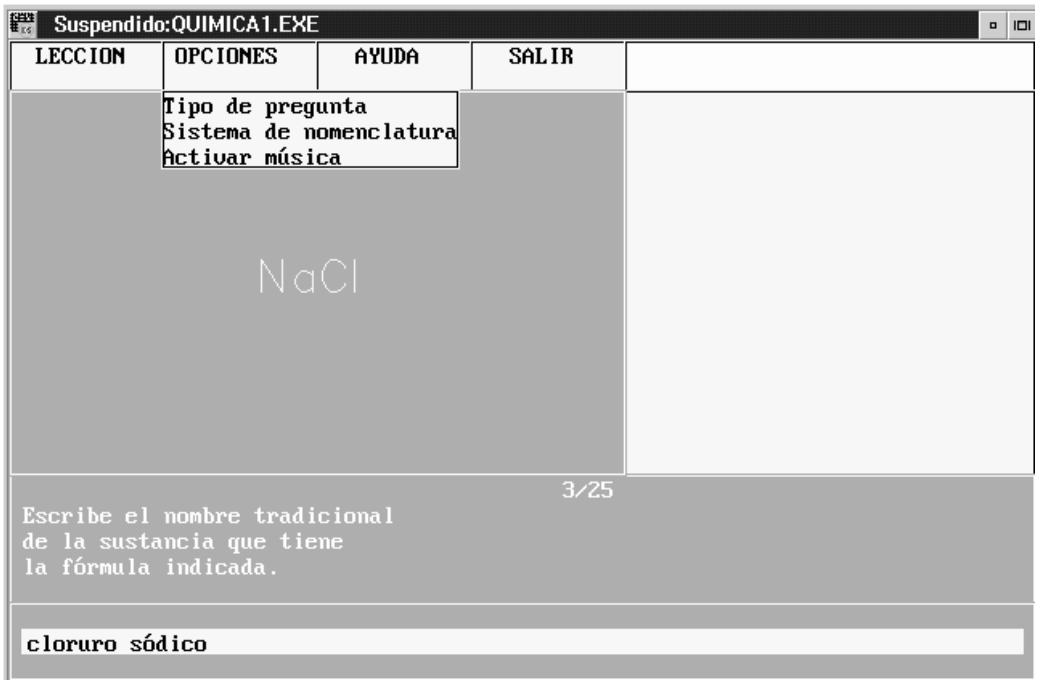

Figure 2: The main screen of the APL2 tutor on Chemistry

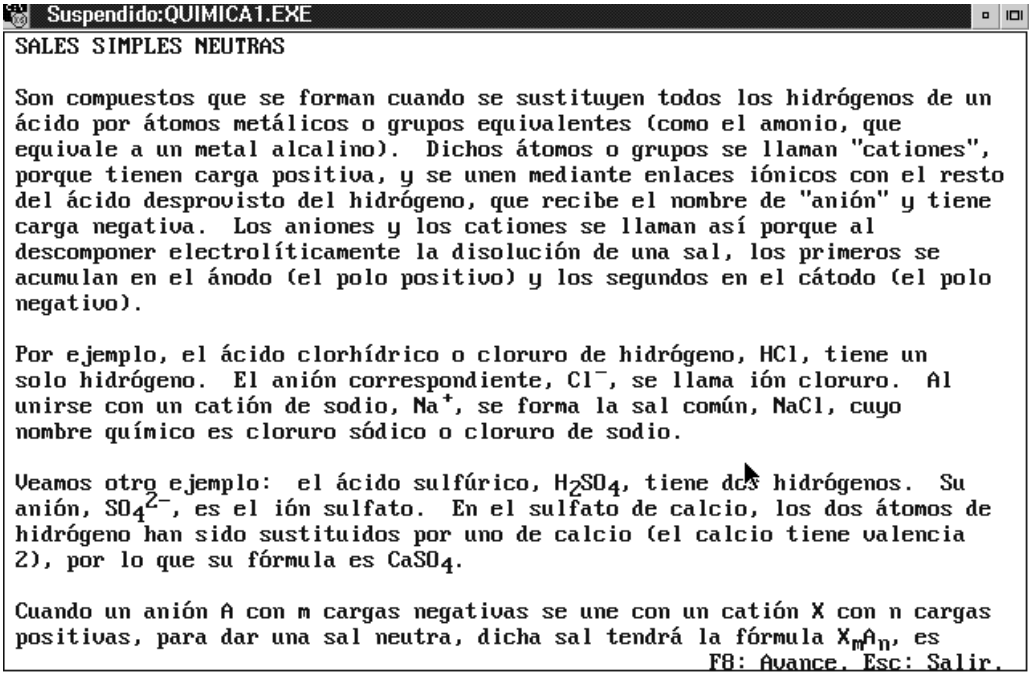

Figure 3: The help screen for the APL2 tutor on Chemistry

- Periodic table help state: Another APL2 function shows the periodic table of elements and provides additional information on each element, such as their atomic weight, density, valences, electronic shell composition, etc, as well as a graphic representation of these properties along the rows and columns of the periodic table (see figure 4).

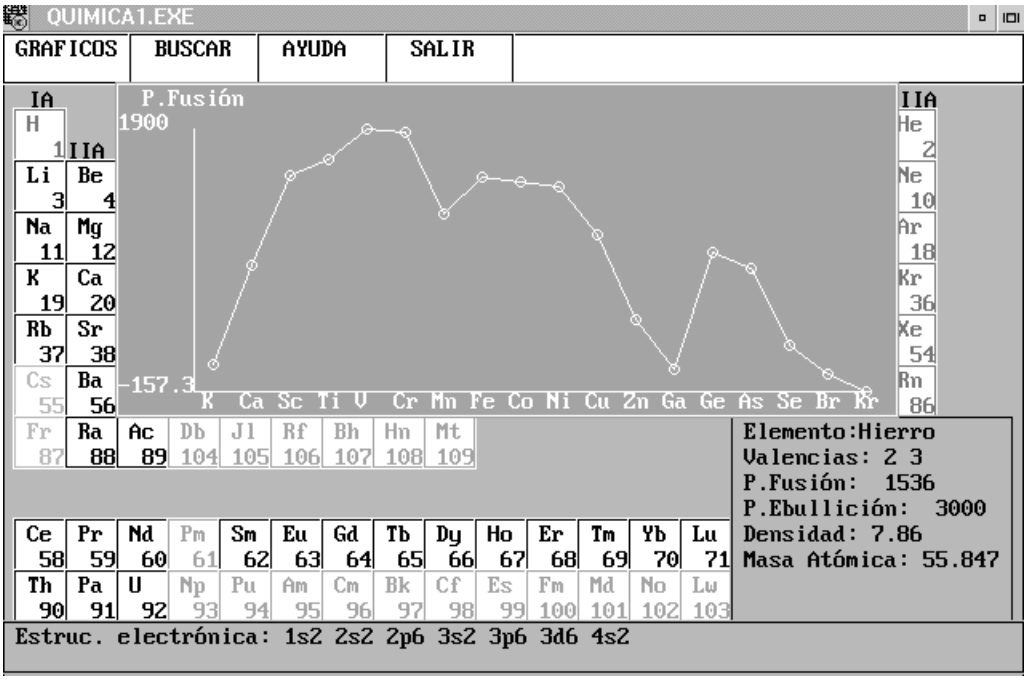

Figure 4: The periodic table in the APL2 tutor in Chemistry

Conclusion

The experience of writing educational science courses in APL2 was successful for the following reasons:

A single problem model may generate thousands of different problems.

Models and explanations may contain APL2 statements and expressions.

Writing both the models and the support program in APL2 made their construction much faster and flexible.

The models must be interpreted, and this means that the APL2 interpreter has to packaged with the executable application program. The standard packaging possibilities of APL2/DOS have been used to generate the DOS version of the courses. We have developed our own packaging procedure to generate the Windows 3.1 versions, where APL2 functions are called from C++ programs.

Writing this application in any other language than APL2 would have been either practically impossible, or much more complicated. This applies also to other interpreted languages, such as Lisp or Smalltalk [9].

The six courses described in the paper have become commercial educational products and have been used successfully in several hundreds of high schools in Spain.

References

[1] *Programas educativos recomendados*, Promo-Soft, Madrid, 1997.

[2] B.STROUSTRUP, *The C++ Programming Language*, Addison-Wesley, Reading (Mass.), 1991-1997.

[3] CUBERO, M.A.; ALFONSECA, M., *Matemáticas (MATES-1), Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1990.

[4] CUBERO, M.A.; ALFONSECA, M., *Matemáticas (MATES-2), Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1991.

[5] CUBERO, M.A.; ALFONSECA, M., *Matemáticas (MATES-3), Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1992.

[6] CUBERO, M.A.; ALFONSECA, M., *Física-1, Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1994.

[7] CUBERO, M.A.; ALFONSECA, M., *Física-2, Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1995.

[8] CUBERO, M.A.; ALFONSECA, M., *Química-1, Manual del tutor, generador y corrector de problemas*, Promo-Soft, Madrid, 1996.

[9] DIGITALK INC., *Smalltalk/V PM*, Digitalk Inc., Los Angeles, 1990.