Simultaneously Teaching Mathematics and Prolog in School Curricula: A Mutual Benefit

Laurent Cervoni[1](✉), Julien Brasseur[1], and Jean Rohmer[2]

[1] Talan Research and Innovation Centre, Paris, France
{laurent.cervoni,julien.brasseur}@talan.com
[2] Institut Fredrik Bull, Paris, France

Abstract. Created in the 1970s, Prolog has its roots in mathematical logic. Its use to model logic problems is natural, but beyond logic, we suggest that using and learning Prolog for most of the topics in the high school math curriculum (probability, algebra, analysis or geometry) allows for a better assimilation of the course concepts. We argue that using Prolog is helpful in that it asks to properly model a problem, which is essential to develop problem-solving skills since it is often the key for finding a solution. At the same time, high school students discover a programming language that is easier to learn than imperative languages since the syntax is close to natural language and the language specification is more synthetic than traditional imperative languages.

Keywords: Prolog · Teaching · Mathematics · Education

1 Introduction

Based on predicate logic, Prolog is particularly well suited for expressing relationships between objects, checking properties about these relationships, validating the consistency of logical rules or for its intrinsic database capabilities.

However, the (re)introduction of Prolog into the educational system at the secondary school level, and its use as a complement to traditional school subjects, provides a new and enriching form of learning that may prove to facilitate the acquisition of certain knowledge.

There have been many examples in the past of the use of Prolog in mathematical analysis (derivative), geometry (Géometrix [6]) or chemistry [7,8]. Various papers [1–4], some going back to the 1980s, have illustrated, through a few examples, the pedagogical interest of Prolog in the teaching of elementary mathematics, or more advanced mathematics (such as the learning of recursion via fractals [5], or related to graphs [9,10]). Its educational potential in the history classroom [11] has also been considered.

Supported by Talan Research Center.

This collection of work shows the interest of exploring the usefulness of Prolog in a pedagogical context. The experiments we have conducted in 2022 with senior High School students seem to confirm its relevance. In this paper, we gather some of the case studies which were proposed to the students.

2 Prolog and Mathematics: A Natural Fit

An important step in solving mathematical problems is to correctly describe and express the problem. That is, being able to identify the givens of the problem and its assumptions. The intellectual process of breaking down the problem into simpler sub-problems is often an essential step towards the solution. Then, the expression of the different components of the problem and its links with the knowledge acquired in class make it possible for the student to devise a process to solve the problem.

Many mathematics teachers (especially, but not only) seem to encourage students to proceed to this "decomposition" of a problem which allows them to better understand how to solve it.

Transcribing a problem into Prolog is a relevant and efficient way to perform the analysis and the elementary decomposition of a problem. Indeed, it involves identifying all the elements of a problem and putting them into factual form. Then, the student must identify the givens of the problem and write them down as rules and facts. (Re)introducing Prolog into mathematics education thus has the double advantage of giving students the keys to a rigorous method for approaching problem solving with an initiation to declarative programming.

The experiments we conducted in a French high school, during the year 2022, with Senior Year students show that it is not necessary to use complex terms (unification, recursion, resolution, for example) to make students understand the concepts. In three sessions of 3 h, they were able to understand the basic principles of writing Prolog programs and to prepare a presentation for middle school students by themselves.

On the other hand, the declarative approach will facilitate the expression of the problem and the solution in "natural language", will invite to make clear sentences, which is an important skill to train. To describe certain problems, they will have to write or express sentences such as:

1. An empty list has a length of zero
2. The last element of a list containing only one element is the element itself
3. The parents of my ancestors are my ancestors.

More generally, writing in Prolog allow to articulate both natural language, drawing (of genealogical relationships for example) and writing an imperative program. Prolog thus allows to "navigate" between abstract and concrete, or between modelling and experimental verification. In order to do this, it is essential to take time with the students, and, above all, not to rush into absorbing the Prolog reference manual.

3 Some Examples

The intrinsic qualities of Prolog justify its implementation in high schools (not an intention to apply it artificially out of context). As we will illustrate with a few examples, its use has the potential to allow students to assimilate the elements of the course and to manipulate them and exploit them easily during exercises.

As we shall illustrate, its use has the potential to allow students to assimilate the elements of the course and to manipulate them easily during the exercises. When solving problems or doing exercises, Prolog may give students a way to understand the flow and application of certain theorems. Let us now consider a few examples.

3.1 Counting Triangles

A classic mathematical puzzle is to count how many triangles there are in a geometric figure made of line segments.[1]

First of all, one has to find a suitable formalism to describe the figure, and another to express the definition of a triangle. This is very easy in Prolog.

It is only needed to give a name to each line, for example by choosing two distinct points through which it passes: AB, AC, DE, DF, ... (in general it will pass through more points than the two chosen to name it). Then, we describe the membership of points to lines:

```
line_point(ab,a).
line_point(ab,b).
line_point(ef,a).
```

and so on. Finally, we express the knowledge of what a triangle is:

```
triangle([A,B,C]):-
      line_point(AB,A),
      line_point(AB,B),
      line_point(BC,B),
      line_point(BC,C),
      line_point(CA,C),
      line_point(CA,A).
```

One can now experiment with this first try in Prolog on a case study, and see that it is not totally satisfactory: Prolog will tell us that a point is a triangle (the three vertices being in coincidence), that a segment line is a triangle (the three vertices being aligned), and, finally, it will find that with three vertices, we can name six triangles $[A, B, C]$, $[A, C, B]$, ..., which is not of much relevance, as these "six" triangles are nothing but one.

[1] See, for example: https://www.rd.com/article/triangle-puzzle/.

These experiments, and the modifications to our Prolog program that they may require (specifying that the vertices and segments must be distinct, e.g. by defining a new *ad hoc* predicate), are an excellent ground to encourage careful, precise exploration of the relationships between concepts, models, experiences, real world, and human perception.[2]

3.2 Polynomials

In early mathematics course, second-degree polynomials are defined as follows:

> *a second-degree polynomial function P is a function defined on $\mathbb{R}$ by: $P(x) = ax^2 + bx + c$, where $a, b, c \in \mathbb{R}$ are real numbers with $a \neq 0$.*

In Prolog, this polynomial in the x variable can be written as:

```
quadPolynomial(A*x^2+B*x+C) :- A=\=0, number(A),
                               number(B), number(C).
quadPolynomial(A*x^2+B*x) :- A=\=0, number(A), number(B).
quadPolynomial(A*x^2+ C) :- A=\=0, number(A), number(C).
```

By writing these three clauses, the pupil becomes aware of different, more or less complete forms of polynomials (the teacher then guides him/her in this). They may start with only the first rule and find that it is not very precise. They can then move on to the standard application exercises of the course, where the aim is to check whether a function is a second degree trinomial or not, for example:

```
quadPolynomial(7). FALSE
quadPolynomial(12*x^2+1*x+0). TRUE
```

The advantage of using Prolog is that the student can transcribe the course definition exactly with a minimum of learning. In the same way, if he has to represent the solution of a second degree polynomial, the course is also written directly in Prolog. Introducing the concept of the undefined variable "_", this can be written as follows:

```
solvePoly(A*x^2+B*x+C, Discriminant, _, _) :-
    Discriminant is B*B - 4*A*C,
    Discriminant $<$ 0.
    /* The discriminant is negative, no solutions to display */
solvePoly(A*x^2+B*x+C, Discriminant, X1, X1) :-
    Discriminant is B*B - 4*A*C,
    Discriminant = 0,
    A =\= 0,
```

[2] More details can be found at: https://fr.slideshare.net/Jean_Rohmer/compter-les-triangles-en-prolog.

```
    X1 is -(B/(2*A)).
    /* The discriminant is zero, X1 is the only solution */
solvePoly(A*x^2+B*x+C, Discriminant, X1, X2) :-
    Discriminant is B*B - 4*A*C,
    Discriminant > 0,
    A =\= 0,
    X1 is (-B-sqrt(Discriminant))/(2*A),
    X2 is (-B+sqrt(Discriminant))/(2*A).
```

Then, "`solvePoly(1*x^2+2*x+1,D,A,B).`", will give `A=B=-1` and `D=0`. This ability to directly represent the course concepts in the form of Prolog facts and clauses contributes to a reinforcement of the learning of the course basics (the student appropriates the concepts and retranscribes them after assimilating a few writing rules, to begin with). Progressively, he also acquires the fundamentals of logic programming (unification, resolution, etc.).

3.3 Euclidean Geometry

These same principles can be applied to various areas of mathematics. Thus, for example, a simple geometry exercise allows the students to express the different stages of the problem but also to describe the elements at their disposal.

We give below an illustration where we "translate" two elementary geometry theorems into Prolog.

```
/* a triangle ABC is right-angled at B,
   if it is inscribed in a circle of which AC is a diameter */
rightTriangle(B, [A,B,C], Circle) :-
    diameter([A,C], Circle),
    inscribed([A,B,C], Circle).
/* AB is perpendicular to EF if there are 2 triangles
   ABE and ABF, both right-angled at B */
perpendicular([A,B],[E,F]) :-
    rightTriangle(B,[A,B,E], Circle1),
    rightTriangle(B,[A,B,F], Circle2).
```

If, now, a student is asked to verify the following property:

> *Let A, B, E and F be four points in the plane. Suppose that the segment $[B,E]$ is a diameter of a circle C_1, that $[B,F]$ is the diameter of another circle C_2 and, finally, that the triangles BAE and BAF are inscribed in the circles C_1 and C_2, respectively. Show that the segments [B,A] and [E,F] are perpendicular.*

It will suffice, then, to describe the problem to Prolog, by writing that:

```
diameter([a,e],c1).
diameter([a,f],c2).
```

```
inscribed([a,b,e],c1).
inscribed([a,b,f],c2).
```

For example, the student can easily check that $[B, A]$ and $[E, F]$ are perpendicular by asking whether "`perpendicular([a,b],[e,f]).`" is true, and Prolog will return "`TRUE`".

Although Prolog is not a substitute for learning how to solve a problem, it is a valuable tool for learning how to understand and structure a problem, which is the first step towards solving it. Learning to pose a problem correctly is sometimes enough to trivialise its solution.

Pythagoras, Thales or midpoint theorems can be expressed just as simply in the form of Prolog clauses and, with the use of the traditional trace when executing a Prolog program, the student can see the solution of an exercise step by step.

In a first step, the teacher can show the students how to express theorems in Prolog, with the learners having to describe the exercises as facts (as in the example above). Then, in a second step, the students write all the case studies (theorems or propositions and descriptions of the exercises) themselves.

4 Conclusion

Solving a problem in Prolog means describing it. As we have seen from a few examples, no matter how you describe the problem, no matter how diverse the description, Prolog will always return an answer. The main interest of Prolog for mathematics education is that it learns to understand, express and structure a problem. This step alone often makes it possible, if not to solve the problem, at least to identify possible lines of attack opening the way to its resolution. It seems to us that this step is fundamental in the problem solving process and that Prolog, by its nature, is adapted to its learning.

The interrelationship between the modelling of theorems or course principles in Prolog and the progressive learning of the language seems to us to be more relevant than in traditional imperative languages where an algorithm must first be imagined. With Prolog's native solving principle, the student discovers the mechanisms that allow him to solve an exercise. The trace helps him to better understand which theorems apply or why some descriptions he may have made are incorrect or incomplete.

However, by calling the Prolog interpreter, the student will get the information that the property can be derived, but not how. The sequence of predicates is essential, and can be obtained by using the trace which, beyond debugging, also allows to understand the "reasoning" implemented and its relevance. Moreover, the solution obtained is relative to the known and described knowledge. A Prolog answer confirms that the knowledge domain is sufficient to reach a satisfactory conclusion, whereas a failure does not show that the requested objective cannot be reached. It does not show that the objective cannot be reached, but rather that the knowledge expressed (in the form of rules and facts) is sufficient.

Finally, it is important to note that languages evolve rapidly and are likely to move from widespread use to more moderate use; therefore, while it is good to introduce students to a computer language, it should not necessarily be chosen on the basis of its popularity, but rather its educational potential.

References

1. Ball, D.: PROLOG and mathematics teaching. Educ. Rev. **39**(2), 155–161 (1987)
2. Bensky, T.: Teaching and learning mathematics with Prolog. arXiv preprint arXiv:2108.09893 (2021)
3. Buscaroli, R., Chesani, F., Giuliani, G., Loreti, D., Mello, P.: A Prolog application for reasoning on maths puzzles with diagrams. J. Exp. Theor. Artif. Intel. 1–21 (2022)
4. Connes, A.: Micro-Prolog et géométrie élémentaire. Bulletin de l'EPI (Enseignement Public et Informatique) **44**, 125–137 (1986)
5. Elenbogen, B. S., O'Kennon, M. R.: Teaching recursion using fractals in Prolog. In Proceedings of the nineteenth SIGCSE technical symposium on Computer science education, 263–266 (1988)
6. Géométrix website, http://geometrix.free.fr/site. Accessed 9 Feb 2023
7. Kleywegt, G. J., Luinge, H. J., Schuman, B. J. P.: PROLOG for chemists. Part 1. Chemom. Intel. Lab. Syst. **4**(4), 273–297 (1988)
8. Kleywegt, G. J., Luinge, H. J., Schuman, B. J. P.: PROLOG for chemists. Part 2. Chemom. Intel. Lab. Syst. **5**(2), 117–128 (1989)
9. McGrail, R. W., Nguyen, T. T., Granda, M. S.: Knot Coloring as Verification. In: 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). pp. 24–31. IEEE (2020)
10. Volk, A. C.: Graph Algorithms in PROLOG, CPS 499/592 Emerging Languages, University of Dayton, Spring (2016)
11. Weissberg, D.: Micro-prolog en classe d'histoire: Montségur au risque de l'informatique. Bulletin de l'EPI (Enseignement Public et Informatique) **39**, 115–120 (1985)