PROLOG, A TECHNOLOGICAL APPROACH TO TEACHING: THE CASE OF MATHEMATICS AND PROBLEM SOLVING

Laurent Cervoni[1] and Julien Brasseur[1]
[1]*Dr*
Talan – Center for Research and Innovation, Rue Dumont d'Urville, 21 – Paris, France

ABSTRACT

A Prolog program consists of a set of facts and rules rather than imperative statements, commonly used in most other programming languages. Therefore, the Prolog language is used to encode logic, from which the inference engine deduces logical conclusions. In this article, we argue that the use of the Prolog language can be useful to help students become familiar with mathematics or more broadly, any problem-solving based discipline.

KEYWORDS

Prolog, Declarative Language, Teaching, Logic

1. INTRODUCTION: THE PROLOG LANGUAGE

Created in the 1970s by Alain Colmerauer and his team (Colmerauer et al., 1972), the Prolog language, whose 50th anniversary we are celebrating in 2022, has its roots in first-order mathematical logic. At first glance, its operation is remarkably simple compared to imperative computer languages: a Prolog program consists of a set of facts and rules - describing a "universe", i.e., a given problem -. The solution principles and the inference engine then allow to "interrogate" this universe to deduce logical conclusions. The philosophy of this unique language can thus be summarized by the motto: describe rather than code.

By its very nature, Prolog is therefore suitable for problems structured in rules and facts. Of course, this covers only a fraction of the types of problems one might be confronted with; but it is far from negligible. Indeed, modern applications of Prolog or Prolog-based languages, despite being few, span over a wide variety of projects such as programming research tools (Garcia-Contreras, Morales, Hermenegildo, 2016; Shumi and Sun, 2022), legal and accounting tools (Satoh et al., 2011) and industrial applications (Wong Chong and Zhang, 2021).

In this article, we argue that Prolog can be useful to help students become familiar with problem-solving methods in scientific disciplines. We will consider the case of mathematics - in particular at secondary school level - as well as the virtues of Prolog in the context of more advanced problem solving. This topic has already been of interest to researchers and teachers. Indeed, several authors have considered the use of Prolog in the teaching of elementary mathematics (Ball, 1987; Bensky, 2021; Buscaroli, et al., 2022; Connes, 1986) as well as more advanced mathematics (Elenbogen & O'Kennon, 1988; Gressier, 2015), and even in domains that are a priori remote, such as history (Weissberg, 1985). These small-scale studies show that despite its shortcomings (sparse user base, lack of interactivity and error reporting, weak optimization), Prolog appears to be a useful pedagogical tool in the teaching of young students.

The 50th anniversary of Prolog offers an opportunity to insist more on its pedagogical virtues, which appear to be underestimated or sometimes poorly known by teachers, and we wish, to place them in a much wider context than just mathematics (this approach is, indeed, applicable to physics, chemistry, for example).

2. CHANGES IN COMPUTER USAGE

In a competitive education environment, the teaching of older programming languages such as Prolog may appear to compete with the use of more widely applied languages such as Python or C++. However, we believe these languages occupy a different space in the education of students.

Indeed, like many computer languages, Prolog has followed an oscillating popularity trajectory. If we refer to the Tiobe Index, which measures the use of computer languages, Prolog went from being the 3rd most used language in 1987, to the 33rd in 2012 before rising slightly to 24th in 2022. Nevertheless, as the case of Pascal attests, (which has declined from the 3rd place in 1992 to the 270th) computer languages are sometimes subject to obsolescence. This is why we feel it is appropriate to specify from the outset that, in an educational context, the pedagogical relevance of a language must take precedence over its present popularity.

Moreover, imperative programming languages such as Python are generally taught to give students a head-start in their career. In this context these languages are then generally viewed as an algorithmic tool to encode the solution of a given problem. Prolog however can be used to teach logic and deductive reasoning, in addition to formulating problems, all of which are a prerequisite to algorithmic reasoning. For that reason, we believe Prolog should be more specifically employed in the context of teaching reasoning to younger students rather than in more applied courses. Moreover, Prolog is a representative of a very particular type of language: to deprive oneself of it would be to renounce a form of reasoning precisely adapted to teaching.

3. PROLOG AND MATHEMATICS EDUCATION

Although the introduction of programming languages in primary and secondary education is now quite old, the trend seems to favor imperative languages to the detriment of declarative languages (such as Prolog). Moreover, this is tantamount to favoring algorithmic reasoning which, while undoubtedly necessary and relevant for many problems, is far from being natural - let alone learnable - or suitable for many of them.

Euclidean geometry, for example, is difficult and unintuitive to approach from an algorithmic point of view. The natural way of reasoning about such problems is to make a series of logical deductions from remarkable propositions, facts or theorems - of which there are relatively few in junior high school or even high school. Here, Prolog is more suitable than languages like Python. Below we give an illustration in which 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 */
rectangle(B, [A,B,C], Circle) :- diameter([A,C], Circle), inscribed([A,B,C], Circle).
/* AB is perpendicular to EF if there are 2 triangles ABC and ABF rectangles at B */
perpendicular([A,B],[E,F]) :- rectangle(B,[A,B,E], Circle1), rectangle(B,[A,B,F], Circle2).
```

Note that one of the remarkable features of Prolog is its concise syntax, which, incidentally, is close to "natural language". If, now, a student is tasked to solve the following problem:

Let A, B, E and F be four points in the plane. Suppose that the segment [B,E] is a diameter of a circle C1, that [B,F] is the diameter of another circle C2 and, finally, that the triangles BAE and BAF are inscribed in the circles C1 and C2 , respectively. Show that the segments [B,A] and [E,F] are perpendicular.

In addition to the rules (predicates) defined above, it will suffice to declare to Prolog the data of the problem (facts), by writing that: diameter([b,e],c1). diameter([b,f],c2). inscribed([b,a,e],c1). inscribed([b,a,f],c2).

In this way, the student can easily check that "perpendicular([b,a],[e,f])" is true, i.e., Prolog will return the value "TRUE".

Although Prolog is not a substitute for classical problem solving, it appears as a valuable aid in learning to understand and structure problems, which is the first step towards solving them. Learning how to pose a problem correctly is sometimes enough to trivialize its solution; and it is precisely for this reason that Prolog is useful in teaching.

But Prolog's interest is not limited to elementary geometry. Let us consider, for example, the learning of the rudiments of calculus: the computation of derivatives, typically taught in the first year of high school. Although there are libraries for certain imperative languages that can perform symbolic operations (e.g. Sympy, in Python), they function as "black boxes". In Prolog, the computation of a derivative is done by declaring the

rules for its computation (see below), together with some facts such as the derivatives of the usual functions. In doing so, it is possible for the student, by deleting or adding a rule, to see what properties are necessary for the calculation of a given derivative and, thus, to better discern the mechanism that governs them.

```
d(U+V,X,DU+DV) :- d(U,X,DU), d(V,X,DV).
d(U*V,X,DU*V+U*DV) :- d(U,X,DU), d(V,X,DV).
d(U-V,X,DU-DV):- d(U,X,DU), d(V,X,DV).
d(U/V,X,(DU*V-U*DV)/V^2) :- d(U,X,DU),d(V,X,DV).
```

4. PROLOG AND PROBLEM SOLVING: LEARNING TO REASON

Let us add that Prolog is intrinsically (by the principle of SL-resolution) a problem solver. Thus, it differs from traditionally taught languages (Python, C++) in that finding an algorithm is not a necessary goal. A Prolog developer is expected to describe the problem to be solved correctly.

To illustrate this, consider the following problem:
Baptiste has a cat. Julien does not live in a villa. Laurent lives in a flat, but the horse is not there. Everyone lives in a different house and has a different animal. Who lives in the castle and who has the fish?
Solving this problem in Prolog is just a matter of describing it:

```
    /* the 3 dwellings */                    /* the 3 animals */
    housing (castle).                        animal(cat).
    housing (flat).                          animal (fish).
    housing(villa).                          animal (horse).
/* the predicate relation constitutes the relation between a person, his animal and his home */
    relationship(baptiste, L, cat):- accommodation(L).
    relation(julien, L, A):- housing(L), L==villa, animal(A).
    relation(laurent, flat,A):- animal(A), A==horse.
/* the predicate different is true only if its 3 parameters are different */
    different(X,X,_):- !, fail.              different(_,X,X):- !, fail.
    different(X,_,X):- !, fail.              different(_,_,_).
/* the predicate "solve" indicates the 4 unknowns to find */
    solve(LogBaptiste, LogJulien, AnimalJulien, AnimalLaurent) :-
    relation(baptist, LogBaptist, cat).
    relation(julien, LogJulien, AnimalJulien).
    different(LogBaptiste, LogJulien, flat).
    relationship(laurent, flat, AnimalLaurent).
    different(AnimalJulien, AnimalLaurent, cat).
```

Then the user can simply ask Prolog:

"Resolve(LogBaptiste, LogJulien, AnimalJulien, AnimalLaurent)",

which will then return:

```
"AnimalJulien = horse,
AnimalLaurent = fish,
LogBaptiste = villa,
LogJulien = chateau".
```

This example is purposefully simple, as it illustrates the potential of Prolog's built-in solving engine. As we have shown in the case of mathematics, Prolog has the advantage of teaching students to reason and to describe a problem by breaking it down into "unitary" elements. The student does not look for which algorithm is needed but tries to analyze the problem posed and break it down into known information (theorems, axioms or facts) before querying the engine.

A second advantage of the language is its syntactic simplicity, which, following first-order mathematical logic, is written in much the same way as a mathematical assertion: a predicate is "true" if a certain conjunctive or disjunctive normal form is true.

5. CONCLUSION

After a minimal effort to familiarize oneself with its syntax, which, as we have seen, is particularly intuitive, Prolog can be used to solve many problems, simply by describing them. This specificity avoids the need to resort to algorithmic reasoning, which is not always intuitive and requires a certain amount of experience to master.

Describing rather than coding emphasizes, in a pedagogical context, the need to take the time to properly pose a problem and its hypotheses before solving it. Incidentally, it allows a pupil to check for him/herself what rules, facts, propositions and other assertions are necessary to deduce the result; this seems to us to be a playful approach to apprehending the logical-deductive mechanisms in problem solving and to develop the pupil's intuition.

REFERENCES

Ball, D. (1987). *PROLOG and Mathematics Teaching*. Educational Review, 39(2), 155-161.

Bensky, T. (2021). *Teaching and learning mathematics with Prolog*. arXiv preprint arXiv:2108.09893.

Buscaroli, R., *et al*. (2022). *A Prolog application for reasoning on maths puzzles with diagrams*. Journal of Experimental & Theoretical Artificial Intelligence, 1-21.

Colmerauer, A., *et al*. (1972). *Un système de communication en français*, preliminary report at the end of the IRIA contract, artificial intelligence group. Technical report, Technical Report, Faculté des Sciences de Luminy, Université Aix-Marseille II.

Connes, A. (1986). *Micro-Prolog and elementary geometry*. Bulletin de l'EPI (Enseignement Public et Informatique), (44), 125-137.

Elenbogen, B. S., & O'Kennon, M. R. (1988, February). *Teaching recursion using fractals in Prolog*. In Proceedings of the nineteenth SIGCSE technical symposium on Computer science education (pp. 263-266).

Gressier, J. *Geometrix website*, 2015. Online: http://geometrix.free.fr/site.

I. Garcia-Contreras, J. F. Morales, & M. V. Hermenegildo. (2016). *Semantic Code Browsing*. Arxiv:1608.02565

Satoh, K. et al. (2011). PROLEG: An Implementation of the Presupposed Ultimate Fact Theory of Japanese Civil Code by PROLOG Technology. In: Onada, T., Bekki, D., McCready, E. (eds) *New Frontiers in Artificial Intelligence*. JSAI-isAI 2010. Lecture Notes in Computer Science, vol 6797. Springer, Berlin, Heidelberg.

Shumi, R. and Sun, J. (2022). ExAIS: Executable AI Semantics. Arxiv:2202.09868

Weissberg, D. (1985). *Micro-prolog in the history classroom: Montsegur at the risk of computers*. Bulletin de l'EPI (Enseignement Public et Informatique), (39), 115-120.

Wong Chong, O. and Zhang, J. (2021). Logic representation and reasoning for automated BIM analysis to support automation in offsite construction, Automation in Construction. Automation in construction, vol 129 (103756)