AI Education Matters: Biductive Computing with Prolog

Joshua Eckroth (Stetson University; jeckroth@stetson.edu)
DOI: 10.1145/3320254.3320261

Introduction

Prolog is a great language to include in an undergraduate AI course. Its logical programming paradigm challenges students to write code in a way that is usually very different from their other courses. Students struggle with its approach but, in my experience, come to appreciate its ability to handle a broad range of AI tasks elegantly.

In my recent Model AI Assignment,[1] I characterized one of Prolog's most distinctive and powerful features, which I've termed "biductive computing." This article describes biductive computing and discusses some projects that make use of it. We also cover student feedback and some resources for teachers who wish to incorporate Prolog into their courses.

Biductive Computing

Biductive computing refers to a combination of "deductive" and "abductive" inference modes. Prolog code is made of facts, rules, and queries. Prolog executes programs by performing inferences to determine if a query is true given the facts and rules. Below, we characterize the different kinds of inferences. The query can contain a variable, which causes Prolog to search for values that make the query true. Note, variables always start with a capital letter.

Consider a GPA calculator. Given a list of a student's courses and grades, Prolog can compute the student's GPA by following rules about credits and averaging. Suppose "grades" is shorthand for a list of courses and grades, and GPA is a variable that will be filled in by Prolog:

```
gpa(grades, GPA)
```

Suppose the resulting GPA is 3.5. We can call this deductive inference or deductive computing because the rules of calculation are followed in a deductive sense: the inputs are well-defined and the output is computed according to the rules.

In most situations, Prolog will also support reverse computations. By using the constraint logic programming library (Triska, 2012), Prolog will be able to determine the course grades in order to achieve a given GPA:

```
gpa(UnknownGrades, 3.5)
```

We can call this abductive computing because we are determining the inputs in order to get a desired output. This is also sometimes called "reverse deduction."

The key is that the same Prolog code supports both kinds of logical inference. One only needs to write code for the rules of the domain – Prolog takes care of the forward and backward inferences. Putting deductive and abductive modalities together, we have "biductive computing."

This means we can give partial information for either inputs or outputs. For example, we can know some of the student's grades but not their grades for the current semester, and we can know (or desire) that the GPA is within a certain range:

```
gpa(PartiallyKnownGrades, GPA),
  GPA >= 3.0
```

Prolog will fill in the unknown grades with real grades and give back the resulting GPA. If the student cannot achieve a 3.0 GPA, Prolog will tell us there is no solution.

Prolog as a State of Mind

My students and I have successfully used Prolog and biductive computing for a variety of projects. My Model AI Assignment shows some projects suitable for a class. These projects include database querying, planning, parsing, and even probabilistic reasoning. Modern Prolog systems can integrate

[1] http://modelai.gettysburg.edu/2018/biductive/

with Java and web environments, further enhancing the variety of use cases that Prolog can support. For example, my book *AI Blueprints* (Eckroth, 2018) includes a chapter in which Prolog is used with deep learning and NLP, using the Rasa library,[2] to develop an intelligent chatbot. The code for this example is freely available.[3]

The GPA example above is based on the Tarot course advising system that a student and I developed in Prolog using a biductive computing approach (Eckroth & Anderson, 2019). Tarot can find which classes a student needs to take to finish their degree and calculate their GPA. It even places these classes in a 4-year semester-by-semester plan. It's also capable of finding which major is easiest to switch to, finding whether a certain desired elective course can be taken at a certain time, and finding the optimal semester abroad. Without any changes to the code, it can also find when prerequisite courses should be offered in order for, say, a particular course to be taken by students in a certain semester. This is all accomplished in less than 150 lines of code.

One of my senior research students is developing an e-sports ranking system. Using biductive computing, he is able to code the logic of the rankings, and then ask if a certain team can achieve a certain rank, if a certain match is on a critical path to a championship, and so on.

This student's reflections on using Prolog for this project characterizes the common reaction I have seen from students who learn Prolog:

> "I've found that Prolog and more popular forms of AI, like machine learning, to be useful in almost opposite ways. Machine learning and neural nets are useful as a dynamic and unpredictable way of creating agents that can identify patterns in data and make decisions based on that with little human input, whereas Prolog is useful as a language for developing a machine with precise, encyclopedic knowledge of a complex system that I can define arbitrarily, with a capacity to handle emergent complexity from relatively simple rules and procedures. This makes it tremendously useful in the right circumstances." — Hayden Estey, Senior, Stetson University

Other students who worked with Prolog in my AI course had similar reactions. They said that Prolog was "fun but hard," probably due to the unfamiliar approach to problem-solving compared to Java, Python, etc. They also said that Prolog was more "principled" than popular libraries like Pandas and Scikit. They felt that Prolog established a set of rules and stuck to them, whereas other libraries sometimes include too many ways to accomplish simple tasks, making it difficult to learn the canonical way to write code. While Prolog is clearly not the best tool for all jobs – there's no reason to use it for building a neural network, for example – its precision at representing facts and rules and performing inferences in logical problem domains is unparalleled.

Pedagogical Resources

Have a look at the following resources for ideas about how to use Prolog in your course. From my Model AI Assignment, consider biductive computing projects in the areas of,

- Database querying to build a Pokédex (i.e., a Pokémon database)
- Course planning as a simplified version of the Tarot system
- Recursive parsing for knitting patterns
- Probabilistic reasoning for solving a crime

Additional resources include,

- Chapter 7 of *AI Blueprints* (Eckroth, 2018) for an example that integrates machine learning and Prolog to build a chatbot
- "Adventures with Prolog: Entering the Dungeon Lord's Lair", a Model AI Assignment from 2016 that helps students practice with the fundamentals of the language[4]
- "Learn Prolog Now!", a free online introductory book[5]

[2] https://github.com/RasaHQ/rasa_nlu

[3] https://github.com/PacktPublishing/AIBlueprints/tree/master/ch07-understanding-queries-and-generating-responses

[4] http://modelai.gettysburg.edu/2016/prolog/

[5] http://www.learnprolognow.org/

References

Eckroth, J. (2018). *AI Blueprints: How to build and deploy AI business projects*. Packt Publishing.

Eckroth, J., & Anderson, R. (2019). Tarot: A course advising system for the future. *The Journal of Computing Sciences in Colleges*, *34*(3), 108-116.

Triska, M. (2012). The finite domain constraint solver of SWI-Prolog. In *Flops* (Vol. 7294, p. 307-316).

Joshua Eckroth joined the Math and Computer Science Department at Stetson University in fall 2014 as an assistant professor. He earned his Ph.D. from The Ohio State University in the areas of artificial intelligence and cognitive science, focusing on abductive reasoning and metareasoning. He edits the website AITopics, one of the most important sources of information on Artificial Intelligence trends. He is also the chief architect at i2k Connect.