Computer Technologies in Logic Education

Yefim Kats
Ellis College of New York Institute of Technology, USA

INTRODUCTION

The recent truly revolutionary changes in information technology triggered the rapid proliferation of educational software supporting introductory as well as advanced college-level logic courses. At the same time, many commercial software packages represent a more or less explicit implementation of logic-based programming paradigm. For example, sequential query language (SQL), designed for such popular database management products as Microsoft Access, Microsoft SQL Server, Oracle, and free- source MySQL, is based on logical query language called relational calculus. From this perspective, it seems not only desirable, but also imperative to introduce carefully selected industrial software packages into the standard Logic and Critical Thinking courses, thus, explicitly linking logical theory with existing as well as emerging applications in information technology. Some of such applications would include database systems, data mining, logic programming, and Web ontologies, among others. Artificial intelligence is still another multidisciplinary area where logic plays an especially prominent role. In this paper, we intend to show how logic-based industrial software can be used in conjunction with specialized as well as broad-based logic courses.

BACKGROUND: SOFTWARE IN LOGIC EDUCATION

Logic courses are currently an intrinsic part of practically any college curriculum. The departments of Philosophy as well as Humanities typically offer Introduction to Logic, Symbolic Logic, and Critical Thinking courses at different levels. Often a course in (mathematical) logic is offered for Mathematics majors, while those majoring in Computer Science have to study logic at least as a module in Discrete Structures course.

The use of computer software for logic courses can be traced to 1950, when "Patrick Suppes introduced his program *Valid* into…Stanford's elementary logic course" (Barwise & Etchemendy, 1996). Since then, a variety of software programs have been developed, assisting instructors and students in introductory and advanced logic courses. Modern educational logic software helps students in such areas as natural deduction in propositional and predicate calculus, syllogistic logic, visual argument representation, various techniques in modal logic, and so forth (Logic Software).

Among the most popular programs: *Tarski's World* for teaching first-order logic, natural deduction based *Fitch,* truth-table checking tool *Bool,* and computability theory software *Turing's World.* We should also mention *Gateway to Logic,* a European collection of elementary as well as advanced logic programs (European Software). In particular, *Tarski's World, Fitch*, and *Boole* are bundled with a popular logic textbook *Language, Truth, and Logic* (Barwise & Etchemendy, 2003); all aforementioned programs can be also successfully used with any standard introductory logic textbook.

While many logic software packages function as standalone tools, one can also find multiple Internet-based programs. Usually such programs are implemented in JAVA (for portability) and share user-friendly interface. We can mention *Causal and Statistical Reasoning System* by Carnegie Mellon University (CSRS), *Power of Logic* for Stephen Layman's textbook, natural deduction tool *Pier*, and *The Logics Workbench* (Layman, 2004; Logic Software). Some tools provide just a convenient self-test and grading facility, while others (like Pier) boast a full-fledged editor.

Logic software seems to be especially useful for those who are interested either in self-study or in pursuing the growingly popular online degree. The growing community of online learners can benefit not only from the automated grading facilities but also from a variety of visualization tools, making the study of logic ever more enjoyable and intuitive. In particular, the already mentioned *Tarski's World* program represents a good example of an intuitive approach to the study of logic, allowing testing first order calculus statements on "three-dimensional worlds inhabited by geometric blocks of various kinds and sizes." (Allwein, et al, 2003, p. 15).

LOGIC AND RELATIONAL DATABASES

Practically all currently available logic software can be used only for the purpose of teaching logic; the links to possible application areas are rather distant. In this paper we propose a completely novel computer-assisted approach to teaching Logic and Critical Thinking courses. This approach by no means is intended to substitute the already existing practice in this area. Our objective is to consider how logic-based industrial software can be explicitly introduced into standard logic courses taught at virtually every university around the world, thus, providing students with the practical skills they could use pursuing a career outside the academia. Our study is intended as a brief introduction to a new teaching methodology rather than a detailed treatment of the vast area of logic software in education. Among the relevant industrial information technology fields are database management, logic programming, and data mining.

As our first example, we consider quantification, the standard topic for any introductory course in formal logic (Copi & Cohen, 1998; Hurley, 2005). The discussion of quantification theory in logic courses can be integrated with, or even taught in, a framework of (tuple) relational calculus, proposed originally by Codd in 1972 as a logical query language for increasingly popular type of industrial software-relational database management system.

Tuple relational calculus is based on the standard first-order calculus, and intended to query industrial databases implementing the relational data model; among them are such widely used products as Oracle, Microsoft SQL Server, and the dominant microcomputer database management system Microsoft Access, a part of Microsoft Office software package installed on practically any PC.

Relational calculus could be integrated into the standard logic course in the following way. First, the students should be taught the standard predicate calculus and only then introduced to its applied modification, tuple relational calculus. Consequently, students may be asked to design (or to be provided with) a sample database, using one of the available commercial products. Once the database is in place, students can practice data manipulation queries using first the natural language queries, and then translating them into the relational calculus queries. At the next, more practical, phase, students may be required to express calculus queries in the standard industrial query language SQL. The order as well as specific details of this process can vary and are up to a logic instructor.

Let us consider a simple example. Assume the following *database schema*:

Bank (bankName, bankNo, location)
Account (accountNo, bankNo, accontType, yield)

Here "Bank" and "Account" are names of the tables created in any familiar-to-the-student relational database system such as Microsoft Access. It is instructive to understand that in the logical context of a relational data model, a table is actually a name of a *relation*, while a column represents an *attribute* of a relation. The structure of a relation is often called by the familiar logical term *intension*; a set of all rows or tuples of any table represents an *extension* of a relation.

These somewhat technical logical terms of intension and extension could be linked to the standard discussion of Socrates' contribution to philosophy. Socrates, as reported by his contemporaries, used to walk around Athens asking its citizens pointed questions like "What is beauty?" or "What is piety?" Usually all he could get in response was a definition by example: his opponents would simply give him a list of entities they would consider beautiful or a list of actions they would call pious. Now, such a list represents what we denote in logic by the term *extension*. However, it is not what Socrates wanted. He used his famous ironic criticism to show the listeners the inadequacy of the proposed definitions by extension; Socratic criticism was intended to demonstrate that such definitions, in modern logical terms, were either incomplete or inconsistent. What Socrates really insisted upon was an *essential definition* or a definition by *intension*, so that for each object presented, we could tell whether an object possesses a certain property such as beauty or piety. (Copi & Cohen, 1998, pp. 137-148)

The point we want to make here is a methodological one. In a standard Introduction to Philosophy course, especially one taught to technology or business majors, it may be instructive to show the connection between the moral inquiries of the old Athenian and the issues discussed in Logic or/and in the professionally oriented

Related Content