A Web-Based Tutoring Tool for Calculating Default Logic Extensions

Uwe Egly, Michael Fink, Axel Polleres, and Hans Tompits
Abt. Wissensbasierte Systeme
Technische Universität Wien, Austria
e-mail: [uwe,michael,axel,tompits]@kr.tuwien.ac.at

Introduction and Background

In this paper, we report on the use of a tutoring program supporting an advanced university course on knowledge-based systems for students of computer science. The course is suggested to be attended at the sixth semester and deals with both practical and theoretical issues regarding knowledge-representation techniques. In previous installments of the lecture, we noticed that certain topics caused problems in understanding among students. A particular hurdle represented the calculation of extensions in Reiter's default logic [Reiter 1980]. Reasons for this difficulties can be found in the fact that (i) the notion of an extension is defined only in a *non-constructive* fashion, and (ii) in order to check whether an object represents an extension, certain skills of formal logic are required, which most students ostensibly lacked, although an undergraduate course on mathematical logic is mandatory in their curriculum. In order to tackle this situation, we had to find a way to make the complex issues easier to comprehend, and, at the same time, provide a better motivation for the students than pure class-room teaching.

Since computer-based education is growing in popularity, and students in general enjoy applying new technologies, we decided to implement a computer program which explains the problematic topic as detailed as possible. The following items where chosen as the main specification of the program:

- it must not only visualize the computation of extensions, but also explain the required steps in detail;
- it should contain examples where the characteristics of default logic are expressed; and finally
- it should require no special software and should run on any computer.

To fulfill the third requirement, we opted to use a JAVA-applet to guarantee highest possible accessibility because all students have access to the Web and any machine which has a JAVA-capable browser can be used to execute the program. In fact, all standard browsers claim to be JAVA-compatible and are free for non-commercial use.

The Applet

Let us first briefly sketch what the tool is actually supposed to visualize.

Default logic belongs to that class of logical formalisms devoted to the study of human common-sense reasoning, i.e., the process of inferring "plausible" conclusions given less than conclusive evidence. A characteristic instance of this sort of reasoning is the frequent approach to assert a particular statement as long as there is no evidence to the contrary. In default logic, such assertions are facilitated by special kinds of inference rules, the so-called *default rules*, stating what is expected to hold under *normal* circumstances. A *default theory*, then, is a collection of default rules, together with a set of definite facts (called the *premises* of the theory).

Since the application of a default depends on both *the presence and the absence of certain knowledge*, different defaults can be mutually blocking and hence the total knowledge induced by a given default theory can give rise to several (if any!) possible "states of affairs". These sets of total beliefs are called *extensions* of the given default theory and play a vital role in default logic.

Unfortunately, the formal definition of an extension is rather tricky and involves some intricate fixed-point construction. (N.B. A *fixed-point* of an operator f is a value x such that $f(x) = x$ holds.) However, for a wide class of default theories, a concrete generate-and-test algorithm can be given which outputs all extensions of a given default theory. Our applet now has the task to visualize this algorithm. The algorithm is as follows: in the first stage, possible candidates for being an extension are generated; and in the second stage, the candidates are checked whether they represent an extension or not.

The program itself consists of several examples to choose from; *in toto* representing characteristic properties of default logic. Each example is provided with a detailed step-by-step solution, running either automatically or manually, in which case the student clicks on a button in order to proceed to the next step. In automatic mode, the speed of the simulation can be adjusted; it can be stopped at any point and also restarted if desired.

The generation of the candidates, and the checking of the candidates are presented in an own window, respectively, and the corresponding steps are given in a structured diagram at the right-hand-side of these windows. Explanations for each step can be requested by simply clicking on the respective text, which are then displayed in a small pop-up window. Help and a general description of each example can also be requested, which will appear in a new browser window.

Responses

Generally, student response was predominately favourable. The few negative reactions all centered around the inability of the respective students to execute the program. However, such situations occurred only if the student disregarded the information we provided specifying the particular versions of the browser which guarantee a trouble-free execution of the applet. For instance, some older (intermediate) versions of Netscape for Linux exhibited certain unpleasant font-related problems, which, however, have been resolved by the current release.

The most interesting question of course was whether the program fulfills our expectation of improving the students skills and examination results. Analyzing the examinations which have been carried out since the availability of the tool, there is an affirmative answer to this question. Let us discuss this in more detail.

Before we supported the students with our tool, their knowledge and ability to generate extensions of given (simple) default theories were rather disappointing. Since we knew these deficiencies, we recapitulated the topic several times during the lectures and discussed many examples. Although we stressed the importance of these examples with respect to the examination at the end of the course, less than 50% of the students got more than 50% of the possible points in those examples concerning default logic.

Since the availability of our tool, however, there is a clear improvement over the previous situation. First of all, the number of students using the tool increased quite significantly over time (from 32% to 58% of all students performing the examination). The reason is that the tool becomes more and more tested and some inevitable software glitches have been removed. Moreover, students profiting from the tool recommend its use to other students. From a performance point of view, there is a marked difference between those students which claim usage of the tool and those either claiming non-usage or giving no answer at all. Tool users got between 8% and 16% more points on the default-logic example than non-users, and 21% more points than students not answering the question about the tool usage.

Future Issues

In the current version of the applet, the examples are hard-coded. However, the program is written in such a way that additional examples can be added without much ado, and that it allows the straightforward inclusion of a simple theorem prover, enabling users to create their own examples. In fact, currently we are working on this extension of the program, and we hope that the new version will be of even greater benefit than the current one. Also, since the concepts underlying the construction of extensions in default logic are closely related to certain semantics for logic programming with negation as failure—which represent similar problems among students—we plan to develop a related visualization tool for that area as well.

References

Reiter, R. (1980). A Logic for Default Reasoning. *Artificial Intelligence Journal*, 13, 81–132.