Tutoring Prolog Novices Based on Programming Techniques

Jun Hong

School of Information and Software Engineering
University of Ulster at Jordanstown
Newtownabbey, Co. Antrim BT37 0QB, United Kingdom
j.hong@ulst.ac.uk

Abstract. We present a techniques-based approach to the tutoring of Prolog programming. The concept of a programming technique is used to characterise and classify programs. We define a set of technique grammar rules for each class of programs, which can be used for program classification, technique and program recognition, and program construction. We use both technique and program frames to represent technique-related and program-related knowledge that provides the basis of error diagnosis and explanation generation for tutoring. Our approach to error diagnosis and explanation generation, however, does not rely on the representation of buggy versions of the program.

1 Introduction

Because Prolog has a simple syntax and few program constructs, it is fairly easy to start Prolog programming. Even though the language knowledge itself does not give the novice any serious problems, studies show that they often have many difficulties with Prolog programming due to the lack of Prolog programming knowledge [9], [10]. This type of metaknowledge in contrast to the language knowledge is therefore essential in Prolog programming.

Many efforts have been made to identify this type of knowledge, and help the Prolog programmer to learn and use it. Brna et al [2] described the concept of a Prolog programming technique. They showed that a knowledge of programming techniques can be useful in both program construction and program debugging. Gegg-Harrison [3] described the notion of basic Prolog construct schemata and showed how they could be used to represent the basic constructs of a structured Prolog for recursive list processing. The basic constructs of Prolog can be viewed as Prolog programming techniques.

All these efforts [2], [3] have been mainly focused on facilitating program construction rather than program debugging and tutoring. Looi [7] described an algorithms-based approach to the debugging of the student Prolog program, that detects errors in the program and proposes the corrections. Looi’s work however has some limitations: The algorithms are randomly picked up without characterisation; All the possible implementations of an algorithm have to be represented; It relies on the representation of buggy versions of programs for
error explanation; It can only give code-related tutoring commentaries to the
student.

In this paper, we present a techniques-based approach to the tutoring of
Prolog programming, that can overcome the limitations in Looi’s algorithms-

based approach. The concept of a programming technique is used to charac-
terise and classify programs. We define a set of technique grammar rules for
each class of programs. These grammar rules can be used for program clas-
sification, technique and program recognition, and program construction. The
technique-related knowledge is represented for each class of programs in a tech-
nique frame. The coding-related knowledge is represented for each program in
a program frame. These two types of knowledge provide the basis of error diag-

nosis and explanation generation for tutoring. Our approach to error diagnosis
and explanation generation however does not rely on the representation of buggy
versions of the program.

We present a system for tutoring Prolog programming to the student. It poses
programming exercises to the student and asks him to do the exercises in the
form of Prolog programs. It analyses the student’s programs and the techniques
used in the programs, and gives comments, suggestions or corrections when ap-
propriate. If the student is not able to program on his own and has failed to
use a programming technique in his program, he is given directive guidance and
provided with program templates in which the relevant programming techniques
have been embedded.

2 Representation of Prolog Programming Techniques

A Prolog programming technique is a common pattern of code used by the
Prolog programmer in a fairly systematic way [2]. Prolog programs can, therefore,
be characterised and classified into different categories, with each category of
programs corresponding to a particular programming technique.

2.1 Schema Representation of Prolog Programming Techniques

Gegg-Harrson [3] provided a Prolog schema language for defining the basic Pro-
log construct schemata that represent the common syntactic features shared by
the corresponding sets of programs. This schema language is an extension of
Prolog supporting first-order schema variables, second-order predicate variables,
optional, arbitrary, and permutable arguments and goals. For example, in the
following schema, any number of arguments can replace schema variables &1,
..., &7 while the terms pre\_pred<<&3>>,H,<<&4>>> and post\_pred<<&6>>,H,
<<&7>>> may be instantiated to either the null string or a single invocation of
the goal.

```
schema\_A([],<<&1>>>).

schema\_A([H|T],<<&2>>>) :-
  pre\_pred<<&3>>,H,<<&4>>>),
  schema\_A(T,<<&5>>>),
  post\_pred<<&6>>,H,<<&7>>>).
```