УДК 004.8

И.А. Андрианов
Вологодский государственный университет

АВТОМАТИЗАЦИЯ ПРОВЕРКИ УЧЕБНЫХ ЗАДАНИЙ ПО ЯЗЫКУ PROLOG

В статье рассмотрены особенности реализации автоматической проверки решений заданий студентов по языку логического программирования Prolog. Приведен пример задания, для которого реализована автоматическая проверка. Полученный программный продукт может использоваться в таких учебных курсах, как «Логика и теория алгоритмов», «Функциональное и логическое программирование».

Язык Prolog, логическое программирование, обучающие системы, электронное обучение.

Логическое программирование – это способ программирования, в котором программы задаются в форме логических утверждений и правил вывода. Можно сказать, что данный подход реализует парадигму декларативного программирования. В декларативном программировании (в отличие от традиционного императивного) разработчик не составляет детальный пошаговый алгоритм решения задачи, а вместо этого определенным образом описывает объекты предметной области, их свойства, связи, а также описывает, что нужно получить в качестве ответа.

История логического программирования насчитывает уже более 50 лет. В разные годы с развитием новых языков и концепций интерес к данной области то увеличивался, то затихал. Наиболее сильно интерес к логическому программированию поднимался в 80-х годах при разработке японской национальной программы компьютеров пятого поколения. Разработчики надеялись, что с помощью данной технологии им удастся добиться значительных успехов в развитии искусственного интеллекта, но фактические успехи были довольно скромные. В настоящее время можно снова заметить рост интереса к логическому программированию в связи с развитием искусственного интеллекта, дедуктивных баз данных, возможности использования его для описания бизнес-логики предприятий и др. Много интересных примеров можно найти в недавней книге [1].

Одним из наиболее известных языков логического программирования является Пролог (Prolog). Он был разработан еще в 70-е годы, но продолжает использоваться и в настоящее время. Математическую основу языка составляет исчисление предикатов первого порядка. Входные данные и связи между ними представляются в виде дизъюнктов Хорна. В качестве механизма логического вывода используются метод резолюций, также для поиска решений применяется перебор с возвратом и отсечениями [2].

На кафедре автоматики и вычислительной техники Вологодского государственного университета язык Пролог используется в основном в двух учебных курсах. Первоначальное знакомство с языком происходит в курсе «Логика и теория алгоритмов», а более подробное изучение – в курсе «Функциональное и логическое программирование». На лекциях студенты изучают особенности данного языка, а затем закрепляют полученные знания во время лабораторных и практических работ.

При этом перед преподавателями возникает следующая проблема. Ручная проверка заданий отнимает от 5 до 15 минут на одно решение (в зависимости от сложности задания). Возникает потребность автоматизировать проверку решений. Решение этой задачи даст целый ряд плюсов: позволит освободить преподавателей от части рутинной работы, повысит качество проверки, позволит сдавать решения дистанционно, в том числе создавать онлайн-курсы с неограниченным количеством обучающихся.

Основная сложность поставленной задачи состоит в следующем. В традиционных электронных обучающих системах (таких как Moodle) можно автоматизировать лишь простые проверки: тесты с выбором вариантов ответов, вводом конкретных чисел и слов. В специализированных же системах, поддерживающих проверку программного кода (Яndex.Contest, Codeforces) язык Пролог не поддерживается. При этом возможность добавления новых языков программирования в них отсутствует.

К счастью, на кафедре автоматики и вычислительной техники ВоГУ разработана и уже несколько лет используется собственная автоматическая проверяющая система – дистанционный практикум по программированию и базам данных, доступный по адресу http://avt.vogu35.ru/acm. Практикум содержит более двух тысяч заданий разной сложности с автоматической проверкой решений [3, 4]. Для нас важно то, что данная система поддерживает добавление новых языков программирования.

На следующем этапе необходимо было решить, какой именно транслятор Пролога добавлять в систему. Существуют несколько трансляторов данного языка. К наиболее известным можно отнести SWI Prolog, GNU Prolog и Visual Prolog. Все перечислен-

ные проекты поддерживаются и развиваются, периодически выходят новые версии.

Нами был выбран SWI Prolog по следующим причинам. В данной версии, кроме консольного транслятора, имеется также онлайн-среда разработки. Это удобно для лабораторных работ, поскольку не требует установки каких-либо программ на компьютере в классе. Кроме того, онлайн-среда поддерживает создание не только традиционных программ, но и так называемых ноутбуков (по аналогии с Jupyter-ноутбуками языка Python). Ноутбук состоит из отдельных ячеек, в каждой из которой написан либо программный код, либо отформатированный текст. Ячейки с кодом можно изменять и запускать в произвольном порядке, что удобно при проведении экспериментов с данными. Кроме того, SWI Prolog содержит целый ряд дополнительных библиотек, например для графики [5].

При добавлении SWI Prolog в систему мы столкнулись с некоторыми особенностями и тонкостями. На первом шаге был установлен транслятор на сервер – это делается обычным образом. Далее для проверки работы была написана простейшая программа для вычисления суммы двух целых чисел:

```
main :- read(A),
  read(B),
  S is A + B,
  writeln(S).
```

Чтобы запустить данную программу из командной строки, использовалась следующая команда:

```
swipl.exe --quiet=true --stack_limit=64M -O -t main aplusb.pl <input.txt
```

Однако при попытке записать два числа в файл input.txt и запустить программу было получено довольно странное сообщение об ошибке:

```
ERROR: main/0: Arithmetic: `end_of_file/0' is not a function
```

Поиск причины данной ошибки показал следующее. Программа начинает работать верно, если числа во входном файле будут завершаться точкой, например «5. 7.». Однако так работать неудобно по двум причинам. Во-первых, в онлайн-среде SWI Prolog при вводе чисел точек добавлять к ним не нужно. Если это будет требоваться на сервере, то вызовет лишнюю путаницу у обучающихся. Во-вторых, хотелось бы, чтобы на Прологе можно было решать не только новые, но и уже имеющиеся в системе задачи. А в их входных данных лишних точек, разумеется, не содержится.

Для решения возникшей проблемы была изучена документация к SWI Prolog, в которой был найден способ чтения чисел без дополнительных символов. Для этого можно создать следующий предикат и использовать его вместо предиката read:

```
readTermWithoutDot(X) :- readln(X1, _, [13, 10, 32], _, _),
    atomics_to_string(X1, X2),
    term_string(X, X2).
```

Но такой вариант, во-первых, громоздок. Во-вторых, он работает только в консоли, но не работает в онлайн-среде. Поэтому окончательно был принят следующий подход. При написании программ в онлайн-среде обучающиеся используют предикат read. При отправке решения на проверяющий сервер в начало программы автоматически добавляется вышеприведенный код, а все вызовы read заменяются на вызовы readTermWithoutDot. Для этого был написан специальный скрипт на языке Python, который выполняет, по сути, препроцессинг исходного кода перед вызовом транслятора swipl.

В заключение приведем пример задания по языку Пролог с автоматической проверкой решений. Имеется база данных на языке Prolog, содержащая описание некоторой семьи. Фрагмент базы выглядит так:

```
parent(bob, liz).
man(bob).
woman(liz).
```

Для подключения базы данных к своей программе нужно написать в начале программы строчку:

```
:- use_module(library(family)).
```

Требуется для двух заданных имен A и B определить, является ли A матерью B. Пример возможного решения:

```
:- use_module(library(family)).
brother_or_sister(X, Y) :- parent(A, X), parent(A, Y).
main :- read(X),  read(Y),
 (parent(X, Y), woman(X) -> write("Yes"); write("No")), nl.
```

Опыт использования реализованной автоматической проверки заданий в учебном процессе подтвердил, что поставленные цели успешно достигнуты. У преподавателя значительно сокращаются временные затраты на проверку решений, а обучающиеся получают уверенность, что их решения будут проверены быстро, качественно и в любое время.

Литература

1. Дженесерет, М. Введение в логическое программирование / Майкл Дженесерет, Винай К. Чаудри ; перевод с английского С. В. Минц. – Москва : ДМК Пресс, 2002. – 192 с.

2. Андрианов, И. А. Математическая логика и теория алгоритмов : методические указания к практическим занятиям и курсовой работе / И. А. Андрианов, А. Н. Сорокин. – Вологда : Вологодский государственный университет, 2013. – 42 с.

3. Web Resource for Teaching Programming in the Form of Tournaments / I. A. Andrianov, S. U. Rzheut-skaya, A. V. Rzheutskiy [et al.] – DOI 10.1109 / Inforino53888.2022.9782974 // 2022 6th International Conference on Information Technologies in Engineering Education, Inforino 2022 – Proceedings : 6, Moscow, 12–15 апреля 2022 года. – Moscow, 2022.

4. Андрианов, И. А. Применение дистанционного практикума по программированию для проверки решений задач математической экономики / И. А. Андрианов // Реформирование экономики: проблемы, успехи, перспективы : Материалы Международной научно-практической конференции (Вологда, 1–2 октября 2020 г.) / главный редактор А. В. Маклахов. – Вологда : ВоГУ, 2021. – С. 29–32.

5. SWI Prolog : официальный сайт. – URL: www.swi-prolog.org (дата обращения: 01.11.2022). – Текст : электронный.

I.A. Andrianov
Vologda State University

AUTOMATION OF CHECKING CLASS ASSIGNMENTS IN THE PROLOG LANGUAGE

The article discusses the features of the implementation of automatic checking of students' tasks in the logic programming language Prolog. An example of a task with automatic check is given. The resulting software product can be used in such training courses as «Logic and Theory of Algorithms», «Functional and Logic Programming».

Prolog language, logic programming, learning systems, e-learning.