Роль логического программирования в изучении информатики

Николай ПЕЛИН,
доктор информатики, доцент,
Тираспольский государственный университет (Кишинэу)

SUMMARY

The paper contains the analysis of the opinions of a number of scholars and specialists on the importance and the role in logic programming methodology of studying computer science, philosophy about the logic programs and interpreter, concerning the burden of which is opposite to the programmer if there is logic interpreter. The presented material is meant, according to the author, to help the reader to understand more easily the analyzed multilateral problem.

Keywords: *logic programming, training, interpreter, indeterminate character of the programs.*

REZUMAT

Lucrarea conține analiza opiniilor unui șir de savanți și specialiști privind importanța și rolul programării logice în metodologia studierii informaticii, filozofiei reprezentării despe programele logice și interpretator, privind sarcina care este opusă în fața programatorului în cazul existenței interpretatorului logic. Materialul prezentat este menit, în opinia autorului, să-l ajute pe cititor să înțeleagă mai ușor caracterul multilateral al problemei analizate.

Cuvinte-cheie: *programare logică, instruire, interpretator, caracterul indeterminat al programelor.*

Состояние проблемы. К. Хоггер пишет в своей книге [1, с.14]: «Логическому программированию с успехом обучали детей младшего школьного возраста, используя при этом содержательные понятия логического следствия и логического вывода. Такой неформальный подход оказался в высшей степени полезным, поскольку он позволяет неискушенному пользователю сравнительно безболезненно усвоить основные принципы».

Разработчик языка логического программирования А. Колмероо и др. писали в их работе [2, с. 30]: «При правильном преподавании информатики необходимо использовать такой язык, который помогал бы структурировать мышление. Сначала людей взращивали на Алголе-60, затем на Паскале. Лучше их воспитывать на Лиспе, но еще лучше - на Прологе». Это был 1988 год.

Что касается обучения тех, кто уже программирует на других языках программирования, то в соответствии с [3] «читать программы очень просто, так как в языке очень мало специальных символов и ключевых слов и они легко переводятся на естественный язык. Главная ошибка программиста состоит в том, что он хочет сразу представить, как программа работает, а не прочитать, что программа описывает, поэтому мне кажется обучить незатуманенный мозг обычного человека го-

раздо проще, чем программиста».

А нужно ли его учить сейчас? Близкий друг другу ответ на этот вопрос можно найти среди многих источников. В [4] он гласит следующим образом: «Думается, что и в настоящее время Пролог остается наиболее популярным языком искусственного интеллекта в Японии и Европе (в США, традиционно, более распространен другой язык искусственного интеллекта - язык функционального программирования Лисп)».

В ответ на аналогичный вопрос, заданный одним из студентов на http://www.cyberforum.ru/prolog/thread223363.html, некто написал: «Пролог - необычайно красивый язык. И выучить его полезно, хотя бы в силу этого обстоятельства. Точно так же, как знание иностранного языка делает человека культурнее, так знание дополнительного языка программирования обогащает». И с этим трудно не согласиться.

О попытках внедрения в Республике Молдова. Возвращаясь к вышеизложенному, взятого из источника [2, с. 30], к концу 80-х годов прошлого века в мире уже был опыт обучения в школах Паскалю, уже были получены некоторые результаты и можно было делать определенные выводы на будущее. И даже если некоторые соседние страны и не вняли этому совету в начале 90-х годов XX-го столетия, у нас была возможность сделать свой выбор.

На заре становления государственности Республики Молдова (1990 г.), когда уже предчувствовалась необходимость в подготовке концепции обучения информатики в школах, автору настоящей статьи удалось объяснить и убедить руководство Государственного педагогического университета им. И. Крянгэ разработать и совместно внести на рассмотрение Министерства образования концепцию программы курса информатики в школах Республики Молдова. Заседание вел заместитель министра, докладчиком был автор настоящей статьи. Присутствовали многие из заведующих профильными кафедрами и подразделениями вузов, техникумов, школ, преподаватели и специалисты в области информатики. Предложенная концепция курса информатики в школах предполагала внесение в программу обучения изучение языка Пролог, как языка способствующего развитию логики обучаемого, языка, и который лучше других языков помогал бы структурировать его мышление. Предложение обосновывалось необходимостью втягивания обучаемого в процесс программирования простотой языка Пролог, его логичностью, близостью к естественному языку. Программа на Прологе не является программой в привычном большинству программистов классическом понимании, поскольку не содержит явных управляющих конструкций типа условных операторов, операторов цикла и т. д. Она представляет собой модель фрагмента предметной области, о котором идет речь в задаче. Решение задачи также формулируется не в терминах компьютера, а в терминах предметной области решаемой задачи [4].

Язык Пролог явно более приемлем для учащихся гуманитарного профиля, но также необходим и для реального профиля. С методологической точки зрения легче учить вначале логический язык, каким является Пролог, а затем алгоритмический Бэйсик, Фортран, Паскаль, С и др. Изучая и программируя в Прологе, учащийся одновременно учит и элементы логики, развивает свое логическое мышление, быстрее освобождается от психологического барьера при программировании компьютера, получив с большей вероятностью продукт, а не только некий результат в виде написанной контрольной работы или в иной форме контроля знаний.

Однако, концепция была отклонена.

Впоследствии в основе базового языка для изучения курса информатики в Республике Молдова лег язык Паскаль. Такой выбор в период, когда уже явно просматривались иные горизонты, автор считал и продолжает считать ошибочным. С первых же шагов по изучению информатики многие из учащихся не могут преодолеть трудности в освоении теории и практики алгоритмического программирования и, собственно, языка Паскаль. Те же, кто преодолел этот искусственно созданный барьер из-за ошибок в стратегии и методологии обучения, тоже в убытке. У многих недостаточно высокий уровень логического мышления, трудности в моделировании реальных ситуаций, сложности в изучении логического программирования после приобретения навыков императивного (алгоритмического) программирования. Несомненно, системе обучения Республики Молдова нанесен реальный урон.

Насколько правильный был выбор языка Паскаль, а не Пролог (или хотя бы ЛИСП, или одновременное обучение языков декларативного и императивного программирования) для изучения в гимназиях и лицеях Республики Молдова в рамках дисциплины «Информатика» отчасти можно судить, в том числе, после ознакомления и с содержанием настоящей статьи.

Во всяком случае, проблема получения логических знаний, развития логического мышления у гимназистов, лицеистов, студентов актуальна. Логическое программирование – одна из возможностей для этого вне зависимости от профиля или специализации. Для тех же, кто специализируется в области информатики, необходим полный и обязательный пакет дисциплин, от структуризации знаний и теории логического программирования, до техники программирования на языке Пролог и его применения для проектирования систем искусственного интеллекта.

Некоторые представления о логических программах и интерпретаторе Пролога. Логическое программирование базируется на представлении, что не человека следует обучать мышлению в терминах операций компьютера, а компьютер должен выполнять инструкции, свойственные человеку [5, с. 10]. В своем предельном и чистом виде логическое программирование предполагает, что сами инструкции даже не задаются, а вместо этого явно, в виде логических аксиом формулируются сведения о задаче и предположения, достаточные для ее решения. Такое множество аксиом является альтернативой обычной программе и будет выполнено в том случае, если будет задано целевое утверждение, которое должно быть формализовано в виде логического утверждения.

Пролог принадлежит к группе *декларативных языков программирования*. В его основе лежит формализованная человеческая логика. Задача программиста заключается в том, чтобы описать решаемую задачу в виде прологовских утверждений, а поиском решения занимается императивная система программирования. Как следствие можно рассчитывать на большую скорость разработки приложений, значительно меньший размер исходного кода, легкость записи знаний на декларативных языках, а также более понятные, по сравнению с императивными языками программы [4].

Реальный мир в логической программе, реализуемой на компьютере. В логическом представлении (в логике), реальная ситуация может быть описана фразами (дизъюнктами) Хорна, т. е. множеством фактов, правил и вопросов (целевых утверждений). При компьютерной обработке эти фразы (дизъюнкты), по заранее установленной стратегии управления (к примеру, поиск в глубину), подвергаются управляющему воздействию,

используя такие понятия, как унификация и метод резолюций.

Упорядоченное множество фактов и правил, как и установленная последовательность целевых утверждений (логические предложения, фразы), формирует то, что мы обычно называем прикладной компьютерной программой. Логическим интерпретатором называют программу, знающую, как выводить следствия из этого множества логических предложений (фраз).

В соответствии с вышеизложенным, возникает вопрос об адекватности интерпретации понятия *алгоритм*, понятия *программа* по отношению к традиционному представлению.

Понятие об алгоритме в философии логического программирования. По определению [6, с. 55], алгоритм (в традиционном представлении) – совокупность правил, определяющих эффективную процедуру решения любой задачи из некоторого заданного класса задач. Алгоритмические языки можно рассматривать как уточнение понятия алгоритм. В этом случае алгоритм трактуется как текст, записанный в алгоритмическом языке. Семантика такого языка определяет для каждого алгоритма (программы), записанного в языке, некоторую совокупность процессов вычислений, которые реализуются в зависимости от состояния информации, перерабатываемой алгоритмом. Исходя из такого определения алгоритма, задача и процесс управления ее решением – нечто единое целое.

В философии логического программирования идея представления алгоритмов с помощью раздельного определения их логических и управляющих компонент играет центральную роль [1, с. 127-128]. Р. Ковальский представил эту идею в виде следующей схемы:

алгоритм = логика + управление.

Отделение логики от управления (логического вывода как вычислительного механизма) упрощает задачу анализа логических возможностей алгоритма и его эффективности в период исполнения.

Программа в алгоритмических (традиционных) языках программирования. Цель каждого вычисления состоит в том, чтобы преобразовать начальное состояние в некоторое конечное состояние, заставляя машину следовать заданной последовательности переходов [1, с. 299]. Программа нужна для того, чтобы точно определить эту последовательность, и она должна сама храниться в памяти машины. Следовательно, программа должна состоять из дискретных инструкций, подробно описывающих как переходы состояний, так и их последовательность. Так мы приходим к понятию фон-неймановского языка программирования, как языка инструкций для машины. Все алгоритмические языки являются языками фон-неймановского типа. Исходя из этого, дальнейшие рассуждения, приведенные в [1, с. 127-128], также очевидны. Если программа является описанием алгоритма, то схематично т. е.

программа = описание алгоритма.

С учетом схемы Ковальского, последнюю схему можно представить как:

программа = описание (логики + управления) .

Разложить, таким образом, составление программ в алгоритмических языках в большинстве случаев невозможно. В последних ход исполнения регулируется состояниями, в которых находятся переменные, однако этим, в свою очередь, определяется, какие присваивания будут иметь место и в каком порядке, т. е. описать (логику + управление) можно только как единый составной объект. В этом случае логические связи между переменными нельзя анализировать и объяснять, не ссылаясь на исполнение программы. А это обстоятельство пагубно для практики программирования.

Задача, стоящая перед програм-

мистом при наличии логического интерпретатора. Программируя в логике, программист уже имеет заранее выполненное описание управления, заложенное в самом логическом интерпретаторе (некий механизм вывода). Оно (описание управления) не зависит от решаемой задачи. Задача программиста, в основном, касается составления описания логики решения задачи.

Программист должен лишь правильно понять поведение имеющегося у него интерпретатора и, выбирая соответствующую логику входной программы, он получает большие возможности для управления поведением своей программы.

Итак, в логических программах практически не делается никакого описания управления, поскольку оно находится в компетенции интерпретатора. Из вышеизложенного следует, что формула сводится к выражению:

программа = описание логики,

т. е. для обозначения множества логических утверждений можно использовать термин «программа».

С точки зрения понимания разницы между логическим и алгоритмическим программированием, вышеприведенные схемы имеют большое значение.

Недетерминированность логических программ. Логические программы могут давать целое множество вычислений. Если конкретная программа допускает более одного вычисления, то она является недетерминированной. Недетерминированность означает отсутствие факторов, точно определяющих ход исполнения программы. Пролог – недетерминированный язык программирования, т. к. в утверждениях логических программ отсутствуют факторы, определяющие ход ее исполнения.

Интерпретатор языка Пролог. Термин *«интерпретация»*, в переводе с латинского, означает истолкование, перевод. Интерпретатор – это языковый процессор, который построчно анализирует исходную программу и одновременно выполняет предписанные действия, а не формирует на машинном языке скомпилированную программу, которая выполняется впоследствии [7, с. 245].

Интерпретатор языка Пролог – это механизм логического вывода, та «активная сила», которая фактически выполняет программы, написанные на Прологе.

Пролог - реляционный язык. Пролог очень хорошо подходит для описания взаимоотношений между объектами. Поэтому Пролог называют реляционным языком. Причем ,,реляционность" Пролога значительно более мощная и развитая, чем ,,реляционность" языков, используемых для обработки баз данных. Часто Пролог используется для создания систем управления базами данных, где применяются очень сложные запросы, которые довольно легко записать на Прологе [4].

Заключение. Работая над освоением, применением и развитием методологии изучения логического программирования нельзя не принять мысль, что:

1. Логическое программирование является мощным средством для программирования компьютера при решении любой задачи, будь то традиционные приложения или базы данных. Что касается проектирования систем искусственного интеллекта, то здесь даже допущения мысли о предоставлении логическому программированию вторичных ролей недопустимо, т. к. до настоящего времени это лучшее, что у нас есть.

2. Логическое программирование это хорошее средство для моделирования систем. Теория систем, системный подход, как и логическое программирование, имеют общие корни, так как базируются на математической логике.

3. Логическое программирование

это среда, в которой работая, то ли моделируя реальный мир, то ли разрабатывая конкретную компьютерную программу на основе полученной модели, человек в то же время развивает свое логическое мышление. Тем, кому представляется, что при программировании на императивных языках программирования также эффективно развивается их логическое мышление, как при программировании на декларативных языках программирования – заблуждаются.

БИБЛИОГРАФИЯ

1. Хоггер К. Введение в логическое программирование: Пер. с англ. М:-Мир,1998, 348 с.

2. А. Колмероо и др. Пролог - теоретические основы и современное развитие. В сборнике «Логическое программирование». М.:- «Мир», 1988.

3. Victor Shcherb. Prolog — грамматический разбор и языковые проблемы. http://habrahabr.ru/post/124636/

4. Григорьева И. В. Рекурсивно-логическое программирование. ftp://ftp.kemsu.ru/incoming/M-116/sao/RLP/lect1.htm

5. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог: М., Мир, 1987.

6. Словарь по кибернетике. Под ред. В. С. Михалевича. 2-е изд. -Киев: Гл. ред. УСЭ, 1989, 751 с.

7. Толковый словарь по вычислительным системам. Под ред. В. Иллингуорта и др.: -М.: Машиностроение, 1989, 568 с.

Prezentat: 22 iunie 2016.

E-mail: nicolae_pelin@yahoo.com