Курс лекций «Введение в ИИ.». Часть I. От психофизиологической проблемы до экспертных систем. Лекция 5. ЯЛП ПРОЛОГ. Часть I

О.Г. Чанышев

Содержание

В средние века знание латинского и греческого языков являлось существенной частью образования любого ученого. Ученый, владеющий только одним языком, неизбежно чувствовал себя неполноценным, поскольку он был лишен той полноты восприятия, которая возникает благодаря возможности посмотреть на мир сразу с двух точек зрения. Таким же неполноценным ощущает себя сегодняшний исследователь в области искусственного интеллекта, если он не обладает основательным знакомством как с Лиспом, так и с Прологом - этими двумя основополагающими языками искусственного интеллекта, без знания которых невозможен более широкий взгляд на предмет исследования.

Иван Братко.
Из предисловия к книге «Программирование на языке Пролог для искусственного интеллекта» [2]

Логическое программирование базируется на убеждении, что не человека надо обучать мышлению в терминах операций компьютера. . . ,а компьютер должен выполнять инструкции, свойственные человеку.

Леон Стерлинг, Эхуд Шапиро.
Из предисловия к книге “Искусство программирования на языке Пролог” [7]

Введение

Эволюция языков программирования – это движение от языков императивных (как делать) к языкам декларативным (что делать). Или иначе – от программирования в инструкциях компьютера к программированию в логике. Понимание того, что алгоритм — это аксиоматическое

задание функции, а его выполнение — частный случай логического вывода, и привело к возникновению логического программирования и языка Пролог. Почти 30 лет прошло с момента его появления. На мой взгляд, сегодня, с появлением Visual Prolog'а пятой версии (компания PDC), Пролог – это самый технологичный язык программирования. Интерес к Прологу вообще на Западе несколько спал после неудачи проекта 5-го поколения ЭВМ, однако кризис доверия быстро миновал и сегодня он более популярен, чем прежде. В России же, если судить по программистской литературе последних лет, Пролог никому не нужен. И этому есть свое объяснение.

Во-первых, в Советском Союзе существовало множество высокопрофессиональных коллективов, занимающихся разработкой программного обеспечения для различных отраслей народного хозяйства. Технология предполагала четкое разделение труда, при котором собственно программирование (кодирование) отдельных небольших модулей производилось кодировщиками на основании подробной "гостированной" технической документации, разработанной постановщиками задач под руководством главного конструктора проекта. В таком случае рабочим языком мог быть ассемблер. Итоговый программный продукт, пройдя комплексную отладку, получался и надежным и быстрым. Пролог же был еще во многом языком экзотическим и не позволял создавать программы, столь же оптимальные по затратам памяти и быстродействию. К тому же, автоматизировались задачи хорошо формализованные, при решении которых главное достоинство Пролога – логический вывод из системы аксиом – как бы затушевывалось вычислительной мощностью вечно молодого дедушки - Фортрана.

Подавляющее большинство кодировщиков не имело никакого опыта в проектировании программных систем и, фигурально выражаясь, их любимым занятием было изобретение наибыстрейших алгоритмов извлечения квадратного корня, которыми, как писал Дж. Мартин, "у меня заполнена вся корзина для ненужных бумаг" [9].

После распада Союза большинство организаций типа ПКБ АСУ прекратило свое существование; сильные в единстве коллективы конструкторов, постановщиков и кодировщиков распались. Наука потеряла в общественном сознании статус интеллектуального эталона и в программистском сообществе стали задавать тон кодировщики, вооруженные философией наибыстрейшего извлечения квадратного корня.

Пролог, как язык концептульного программирования - незаменимое средство индивидуальной разработки программного продукта, но для этого нужно концептуально мыслить.

Какой смысл я вкладываю в термин язык концептуального программирования?

Прежде всего, это средство решения при помощи компьютера широкого класса задач таким образом, чтобы максимально большая часть нейронов мозга была занята самой задачей и следствиями, которые могут быть получены в результате ее правильного решения, а не способом описания решения для компьютера. В конце концов, это одна из причин возникновения научного направления "искусственнный интеллект". Нужно вырыть яму, а не исследовать свойства лопаты. Пролог прост, поскольку использует небольшой набор базовых механизмов, включающих сопоставление образцов, древовидное представление структур и "бэктрекинг" (автоматический возврат при поиске решения). Пролог заставляет Вас сконцентрироваться на осознании задачи и описании предметной области, а не на деталях программной реализации. Языки же типа Си представляют все возможности, чтобы забыть о цели и заставить думать о себе.

На мой взляд две причины мешают овладению Прологом. О первой мы говорили. Если Вы не ученый, не эксперт в определенной предметной области, если Вы пишите программы для того, чтобы их писать и сам процесс писания (программистская графомания) доставляет Вам удовольствие, не пишите Прологе. Он примитивен.

Вторая причина – привычка все объяснять тупому компьютеру. (Позволю себе вспомнить

старый анекдот, не слишком удаляясь от темы: страна Советов – это страна, в которой все советуют. Особенно те, кто меньше знает, тем, кто знает больше (ограниченность кругозора создает иллюзию единственнности личной модели мира)). Например, пусть имеется массив с именем cifra, хранящий арабские цифры (0..9). Конструкция типа:

```
cicle()
{
integer i;
for (i=0 to 9 by 1) write(cifra[i]);
}
```

понятна и привычна в отличие от функционально аналогичной записи на Прологе:

```
cicle:- cifra(C),write(C),fail.
cicle.
```

Пролог значительно меньше нуждается в “советах”, чем другие языки. Он будет искать истинное решение до тех пор, пока это возможно (в нашем примере он находит его во втором правиле для предиката cicle, перебрав все факты терма cifra(integer)).

Правда, Вы можете написать на Прологе и в императивном стиле:

```
...
recurs_cicle(0,9),
...
recurs_cicle(I,N):-
    I<=N,
    cifra(I),!,write(I),
    Inew=I+1,
recurs_cicle(INew,N);
!.
```

Но зачем?!

Признаюсь, когда в 92-ом я начал программировать на Turbo Prolog’e v.2.0., то писал во вполне “императивном” стиле. Но простота конструкций экономила массу времени. И только после полутора лет работы с Прологом я как-то вдруг осознал возможности по решению плохо формализованных задач, задач распознавания образов, кластеризации и т.п., скрывающиеся за термином “бэктрекинг”. По критерию – количество деталей реализации, которые нужно держать в голове при программировании “с листа” - Пролог не знает себе равных.

Система управления внешними («бинарными») БД Пролога логически идеально подходит для организации сложных сетевых структур – неотемлемой части систем искусственного интеллекта - выгодно отличаясь от разнообразных реляционных СУБД, удобных только для формализации коммерческих задач.

Пролог – язык быстрого прототипирования. Это означает, в частности, что Вы можете в течении какого-нибудь часа написать и отладить программу для эксперимeтальной проверки положений вашей концепции. Это означает, что Вы можете подойти к решению задачи, как физик-экспериментатор: быстро разработать программный продукт для исследовательских целей, проанализировать экспериментально исследуемое явление, внести в модель поправки и перейти к следующей серии опытов. В теории экспертных систем это называется инкрементальным методом: “Поскольку для достижения высокого качества работы необходимо экусперементирование, то экспертная система развивается постепенно.Такой эволюционный или инкрементальный метод создания стал доминирующим в области экспертных систем” [10].

1 История Пролога

[1] В 1973 г. группа исследователей из Марсельского университета под руководством Алана Колмероэ, опираясь на работы Джона Робинсона, посвященные методу резолюций, создала программу для доказательства теорем, которая была реализована на языке Фортран. Впоследствии этот продукт получил название Пролог (Prolog — от Programmation en Logique).

В течение ряда лет совершенствовалась теория логического программирования. Существенный вклад в ее развитие внесла работа Р. Ковальского «Логика предикатов как язык программирования». В 1976 г. Ковальский и М. ван Эмден предложили два подхода к прочтению текстов логических программ — процедурный и декларативный.

В 1977 г. Д. Уоррен и Ф. Перейра создают в университете Эдинбурга интерпретатор/компилятор языка Пролог для ЭВМ DEC-10, тем самым переведя методы логического программирования в практичекую плоскость. В 1980 г. К. Кларк и Ф. Маккейб в Великобритании разработали версию Пролога для персональных ЭВМ.

В октябре 1981 г. была широко распространена информация о японском проекте создания ЭВМ пятого поколения. В основу методологии разработки программных средств было положено логическое программирование. Целью проекта декларировалось создание систем обработки информации, базирующихся на знаниях. В это же время появляется множество коммерческих реализаций Пролога практически для всех типов компьютеров. К наиболее известным можно отнести CProlog, Quintus Prolog, Silogic Knowledge Workbench, Prolog-2, Arity Prolog, Prolog-86, Turbo Prolog и др.

Наиболее известна в России система программирования Turbo Prolog — коммерческая реализация языка для IBM-совместимых ПК. Ее первая версия разработана датской компанией Prolog Development Center (PDC) в содружестве с фирмой Borland International в 1986 г. Самым существенным отступлением от неофициального стандарта [5] было введение строгой типизации данных, что это позволило значительно ускорить трансляцию и выполнение программ.

В 1988 г. вышла значительно более мощная версия Turbo Prolog 2.0, включающая усовершенствованную интегрированную среду разработки программ, быстрый компилятор и средства низкоуровневого программирования. Фирма Borland распространяла эту версию до 1990 г., а затем компания PDC приобрела монопольное право на использование исходных текстов компилятора и дальнейшее продвижение системы программирования на рынок под названием PDC Prolog. В июне 1992 г. появилась версия 3.31 — эффективный универсальный инструмент профессиональных программистов, который вскоре стал одним из наиболее широко используемых. PDC Prolog 3.31 работал в среде MS DOS, OS/2, UNIX, XENIX, PharLap DOS Extender, MS Windows. Эта версия была хорошо совместима с традиционными языками программирования, в первую очередь с Си. В ней были расширены возможности создания приложений с интерфейсом GUI (Graphical User Interface), принятым в MS Windows и OS/2. В 1996 г. Prolog Development Center выпускает на рынок систему Visual Prolog 4.0. В этой работе участвовали российские программисты под руководством Виктора Юхтенко, который позже стал техническим директором компании «Пролог-Софт», представляющей интересы PDC в России.

В Visual Prolog входят: интерактивная среда визуальной разработки (VDE — Visual Develop Environment), которая включает текстовый и различные графические редакторы, инструментальные средства генерации кода, конструирующие управляющую логику (Experts), а также интерфейс визуального программирования VPI (Visual Programming Interface), Пролог- компилятор, набор различных подключаемых файлов и библиотек, редактор связей, файлы, содержа-

[1] Излагается по [1]

щие примеры и помощь.

Visual Prolog – многоплатформенная среда программирования, позволяющая разрабатывать приложения, работающие под управлением различных операционных систем - MS-DOS, PharLap-Extended DOS, всеми версиями Windows, 16- и 32-битовыми целевыми платформами OS/2, UNIX. Ресурсы и инструментальные средства (окна, меню, диалоги, органы управления, перья, кисти, курсоры мыши, графические курсоры, рисунки и т. п.), представляются в виде несложных Пролог-структур.

В декабре 1997 г. фирма PDC выпустила Visual Prolog 5.0, с января 1999 г. приступила к распространению версии 5.1. В настоящее время все желающие могут бесплатно скопировать через Internet последнюю версию системы Visual Prolog 5.2 Personal Edition, функционирующую в средах Windows 3.1/95/98, NT, OS/2, SCO UNIX и Linux. Ее загрузочный файл объемом 20 Мбайт можно найти по адресам:

`http://www.visual-prolog.com/vip/vipinfo/freeware_version.htm,`
`http://www.pdc.dk/vip/vipinfo/freeware_version.htm.`

Вариант Personal Edition предназначен для некоммерческого использования, и сообщения об этом постоянно имеются во всех приложениях, созданных с его помощью.

Все продукты PDC, включая Visual Prolog, — это системы, порождающие исполняемый код (EXE или DLL), что еще раз подтверждает высокую эффективность Пролога.

2 Синтаксис и семантика Пролог-программ

2.1 Объекты данных

[2] Объекты данных в Прологе могут быть простыми данными и стуктурами. Простые данные могут быть константами и переменными. Константы могут быть атомами, числами и строками.

Пролог- система распознает тип объекта по его синтаксической форме в тексте программы.

Атомы. Атом – комбинация букв, цифр и знака подчеркивания, начинающаяся со строчной буквы. Примеры: a, `это_атом`, `this_is_atom`.

Переменные. Переменная – комбинация букв, цифр и знака подчеркивания, начинающаяся с прописной буквы. Примеры: V, `это_переменная25`.

Структуры. Структурные объекты (или просто структуры) – это объеты, состоящие из нескольких компонент. Компоненты, в свою очередь, могут быть структурами. Для объединения компонент в структуру используется функтор:

date(17,июнь,1999), date – функтор.

date(Day,июнь,1999), здесь – Day – переменная, которая может получить значение (стать связанной переменной) на ком-то этапе вычислений. Синтаксически все объекты данных в Прологе есть термы.

2.2 Сопоставление.

Наиболее важная операция над термами – сопоставление. Два терма сопоставимы, если:

- они идентичны,

- переменным в обоих термах можно присвоить в качстве значений объекты таким образом, что после подстновки они станут идентичными. Например, date(Day,июнь,1999) и date(Day1,июнь,1999)

[2] Излагается по [2, 7]

сопоставимы, поскольку переменным Day и Day1 можно присвоить одинаковые значения от 1 до 31. Сопоставление – процесс проверки сопоставимости термов.

2.3 Декларативный смысл Пролог-программ

Дано предложение:

$$P : -Q, R.$$

Q и R имеют синтаксис термов. Возможные способы декларативной интерпретации этого предложения:

P истинно, если Q и R истинны.

Из Q и R следует P.

Два варианта процедурного прочтения:

Чтобы решить задачу P нужно сначала решить подзадачу Q, а затем – подзадачу R.

Чтобы достичь P, сначала достигни Q, затем достигни R.

Таким образом, различие между декларативным и процедурным прочтениями заключается в том, что последнее не только определяет логические связи между левой и правой (цели) частями предложения, но еще и порядок, в котором эти цели обрабатываются.

Декларативный смысл программы определяет, является ли данная цель истинной и, если да, то при каких значениях переменных это достигается. Конкретизацией предложения называется результат подстановки в него на место каждой переменной некоторого терма. Вариантом предложения C называется такая конкретизация C, при котрой каждая переменная замененна на другую переменную.

Например:

$$reader(X) : -have\_book(X, Y)$$

Два варианта:

$$reader(X1) : -have\_book(X1, Y2)$$

$$reader(Beta) : -have\_book(Beta, Alpha)$$

Конкретизации:

```
reader("Колмогоров"):- have_book("Колмогоров",Z).
reader("Васильев"):-
have_book("Васильев",book("Овод")).
```

В общем случае, вопрос к Пролог-системе есть список целей, разделенных запятыми. Список целей будет истиннным, если все цели в списке истинны при одинаковых конкретизациях переменных. Запятая между целями обозначает конъюнкцию целей. В Прологе возможна и дизъюнкция целей: должна быть истинной, по крайней мере, одна из целей. Дизъюнкция обозначается точкой с запятой (;):

```
 P:-Q;R.
P истинно, если истинно Q или P.
То же самое можно написать в виде двух правил Пролога:
```

```
P:-Q.
P:-R.
```

Запятая связывает цели сильнее (имеет более высокий приоритет), чем точка с запятой. Предложение

$$P : -Q, R; S, T, U.$$

понимается как

$$P : -(Q, R); (S, T, U).$$

2.4 Процедурная семантика Пролог-программ

Процедурная семантика определяет, как Пролог-система отвечает на вопросы. Ответить на вопрос – это значит удовлетворить список целей. Этого можно добиться, приписав переменным значения таким образом, чтобы цели логически следовали из программы. Процедурная семантика Пролога – процедура вычисления списка целей.

Для пояснения того, как Пролог вычисляет список целей, прямо воспользуемся примером Братко [2], в котором задается вопрос базе знаний: “Кто у нас темный и большой?”

Программа:

```
большой("медведь").
большой("слон").
маленький("кот").
коричневый("медведь").
черный("кот").
серый("слон").
темный(Z):-черный(Z).
темный(Z):-коричневый(Z).
```

Вопрос:

```
темный(X),большой(X).
```

Шаги вычисления:

1) Исходный список целевых утверждений: темный(X),большой(X).

2) Просмотр всей программы от начала к концу и поиск предложения, у которого голова сопоставима с первым целевым утверждением:

темный(X)

Найдено предложение 7:

темный(Z):-черный(Z).

Замена первого целевого утверждения конкретизированным телом предложения 7 – порождение нового списка целевых утверждений:

черный(X):-большой(X)

3) Просмотр программы для нахождения предложения,сопоставимого с черный(X). Найдено предложение 5: черный(кот). У этого предложения нет тела, поэтому список целей при соответствующей конкретизации сокращается до

большой(кот).

4) Просмотр программы в поисках этой цели завершается неуспехом и происходит возврат к шагу 3 и отмены конкретизации X=кот. Список целей вновь:

черный(X),большой(X)

Продолжение просмотра ниже предложения 5. Ни одно предложение не найдено. Возврат к шагу 2 и продолжение просмотра ниже предложения 7. Найдено предложение 8:

темный(Z):-коричневый(Z).

Замена первой цели в списке на коричневый(X) дает:

коричневый(X),большой(X)

5) Просмотр программы для обнаружения предложения, сопоставимого с коричневый(X)дает коричневый(медведь). У этого предложения нет тела, поэтому список целей уменьшается до

большой(медведь)

6) Просмотр программы и обнаружение предложения большой(медведь).

У него нет тела, поэтому список целей становится пустым. Это указывает на успешное завершение, а соответствующая конкретизация переменных:

X=медведь.

Забегая немного вперед, представляю запись этой же программы на Turbo Prolog'e. (Если используете VP v.5.2, можете писать по русски все, за исключением типов встроенных переменных, наименований встроенных предикатов, наименований разделов программы)

/* Раздел объявления предикатов*/

```
predicates
  большой(string)
  черный(string)
  коричневый(string)
  серый(string)
  темный(string)
  маленький(string)
/*Главная цель-вопрос*/
goal
    темный(X),большой(X),
        write(X).
/*Правила*/
clauses
    большой("медведь").
    большой("слон").
    маленький("кот").
    коричневый("медведь").
    черный("кот").
    серый("слон").
    темный(Z):-черный(Z).
    темный(Z):-коричневый(Z).
```

Список литературы

[1] Игорь Швыркин. Пролог. Генезис.
Мир ПК, №5,2000

[2] Братко И. Программирование на языке Пролог для искусственного интеллекта/Пер. с англ. — М.: Мир, 1990.

[3] Доорс Дж., Рейблейн А.Р., Вадера С. Пролог — язык программирования будущего/Пер. с англ. — М.: Финансы и статистика, 1990.

[4] Ин Ц., Соломон Д. Использование Турбо-Пролога /Пер. с англ. — М.: Мир, 1993.

[5] Кларк К., Маккей Ф. Введение в логическое программирование на микро- Прологе /Пер. с англ. — М.: Радио и связь, 1987.

[6] Клоксин У., Меллиш К. Программирование на языке Пролог /Пер. с англ. — М.: Мир, 1987.

[7] Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. /Пер. с англ. — М.: Мир, 1990.

[8] Стобо Дж. Язык программирования Пролог /Пер. с англ. — М.: Радио и связь, 1993.

[9] Дж. Мартин. Надежность программного обеспечения. /Пер. с англ. – М.:Мир, 1986.

[10] Построение экспертных систем./Под ред. Хейеса-Рота, Д. Уотермана, Д. Лената М.: Мир, 1987.

Следующая лекция

Лекция 6. ЯЛП ПРОЛОГ. Часть II