ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ. ЯЗЫК PROLOG

Список рекомендуемой литературы

1. *Шрайнер П.А. Основы программирования на языке Пролог. Курс лекций. Учебное пособие — М.: нтернет-Ун-т Информ. Технологий, 2005.*
2. А.Н. Адаменко, А.М. Кучуков. Логическое программирование и Visual Prolog — СПб.: БХВ-Петербург, 2003.
3. Братко И. Программирование на языке Пролог для искусственного интеллекта. - М.: Мир, 1990.

В октябре 1981 года Японское министерство международной торговли и промышленности объявило о создании исследовательской организации — Института по разработке методов создания компьютеров нового поколения (Institute for New Generation Computer Technology Research Center). Целью данного проекта было создание систем обработки информации, базирующихся на знаниях и переходу к ЭВМ пятого поколения. Основная суть качественного перехода к пятому поколению ЭВМ заключалась в переходе от обработки данных к обработке знаний, отойдя при этом от фон-Неймановской архитектуры компьютеров. В 1991 году предполагалось создать первый прототип компьютеров пятого поколения (полностью задача не реализована). Этот проект послужил импульсом к развитию нового витка исследований в области искусственного интеллекта и вызвал взрыв интереса к логическому программированию. Думается, что и в настоящее время Prolog остается наиболее популярным языком искусственного интеллекта в Японии и Европе (в США, традиционно, более распространен другой язык искусственного интеллекта — язык функционального программирования Лисп).

Цель создания языка логического программирования – попытка автоматизировать выполнение логического вывода. Очевидно, что логическое программирование основано на математической логике, в которой используется язык формул. Формула в матлогике строится из высказываний и предикатов, объединенных знаками операций – конъюнкция, дизъюнкция, отрицание (базис) с кванторами: всеобщности и существования. (**Вопрос**: что такое высказывание и предикат?) Упрощение формул матлогики по определенным законам позволяет понять является ли формула тавтологией или нет, т.е. сделать логический вывод.

Математическая база Prolog – языка логического программирования это принцип резолюции, опубликованный (доказанный) лишь в 1965 г. (Дж. Робинсон). Принцип резолюции - это метод автоматического поиска доказательства теорем в исчислении предикатов первого порядка. Первая версия языка Prolog была написана на яз.Fortran в 1973г. в Марсельском универ-те.

Логическое программирование, которое принципиально работает не с данными, а со **знаниями**, поддерживается декларативными языками программирования. Prolog – это декларативный язык программирования. Он основан на системе правил, декларированных в программе, - знаниях. В отличие от императивного программирования, порядок действий в системе логического программирования определяется системой правил (т.е. особым способом обработки совокупности правил – алгоритмом **унификации**), а не определяется порядком записи правил в тексте программы.

Программируя в **императивном** стиле, программист должен объяснить компьютеру **как** нужно решать задачу (порядок действий). Программируя в декларативном стиле, программист должен описать **что** нужно решать, т.е. предметную область. Программа на **декларативном языке** представляет собой совокупность утверждений, описывающих

фрагмент предметной области (знания о предметной области) или сложившуюся ситуацию, а не порядок поиска решения.

Основным элементом языка является **терм. Терм – это:**

1. **Константа:**
   - Число (целое, вещественное),
   - Символьный атом (комбинация символов латинского алфавита, цифр и символа подчеркивания, начинающаяся со строчной буквы: aA, ab_2), используется для обозначения конкретного объекта предметной области или для обозначения конкретного отношения,
   - Строка: последовательность символов, заключенных в кавычки,
2. **Переменная:**
   - Именованная – обозначается комбинацией символов латинского алфавита, цифр и символа подчеркивания, начинающейся с прописной буквы или символа подчеркивания (X, A21, _X),
   - Анонимная - обозначается символом подчеркивания (_),
3. **Составной терм:**
   - Это средство организации группы отдельных элементов знаний в единый объект, синтаксически представляется: **f(t1, t2, …,tm)**, где f - функтор (функциональный символ) , t1, t2, …,tm – термы, в том числе и составные (их называют аргументами), (например: likes(judy, tennis) – знание о том, что judy любит tennis_ или еще, например: book(author(tolstoy, liev), war_and_peace) и т.д.). Аргументом или параметром составного терма может быть константа, переменная или составной объект. Число аргументов предиката называется его **арностью** или **местностью**. Составные термы с одинаковыми функторами, но разной арности, обозначают разные отношения.

С помощью термов и более сложных конструкций языка Prolog – **фактов** и **правил** «описываются» знания о предметной области, т.е. **база знаний**. Используя базу знаний, система Prolog будет делать логические выводы, отвечая на наши **вопросы**. Таким образом, **программа на Prolog представляет собой базу знаний и вопрос**.

База знаний состоит из предложений - CLAUSES (отдельных знаний или утверждений**): фактов и правил**. Каждое предложение заканчивается точкой.
Предложения бывают двух видов: факты и правила. Предложение более общего вида – **правило** имеет вид:

$$A :- B_1, \ldots , B_n.$$

A называется **заголовком правила**, а B_1,..., B_n – **телом правила**.

Факт – это частный случай правила. Факт – это предложение, в котором отсутствует тело (т.е. тело пустое).
Причем, A, B_1,... , B_n – это термы; символ "**:-**" это специальный символ-разделитель.

Заголовок содержит отдельное **знание** о предметной области (составной терм), а тело содержит условия истинности этого знания. Правило называют условной истиной, а факт, не содержащий тела – безусловной истиной.

Заголовок, как составной терм f(t1, t2, …,tm) , содержит **знание о том**, что между

аргументами: t1, t2, …,tm существует **отношение** (взаимосвязь, взаимозависимость). А **имя** этого **отношения** – это **f**. Например: likes(judy, tennis)

Если это факты (или правила) программы, то это записывается в разделе - CLAUSES:

likes(judy, tennis).
mother("Наташа", "Даша").

И далее другие факты. Программа может состоять только из фактов. Программа может состоять из фактов и правил – знаний. Одно знание может быть записано с помощью нескольких предложений.

Третьим специфическим видом предложений Prolog можно считать **вопросы**. Вопрос состоит только из тела – составного терма (или нескольких составных термов). Вопросы используются для выяснения выполнимости некоторого отношения между описанными в программе объектами. Система рассматривает вопрос как цель, к которой (к истинности которой) надо стремиться. Ответ на вопрос может оказаться логически положительным или отрицательным, в зависимости от того, может ли быть достигнута соответствующая цель.

Если составные термы, факты, правила и вопросы не содержат переменных, то они называются основными. Составные термы, факты, правила и вопросы в момент фиксации в программе могут содержать переменные, тогда они называются неосновными. Переменные в момент фиксации утверждений в программе, обозначая некоторый неизвестный объект из некоторого множества объектов, не имеют значения. Значения для переменных могут быть установлены Prolog-системой только в процессе поиска ответа на вопрос, т.е. реализации программы.

Программа на Prolog может содержать вопрос в программе (так называемая **внутренняя цель GOAL**). Если программа содержит внутреннюю цель, то после запуска программы на выполнение система проверяет достижимость заданной цели, исходя из базы знаний. Например:

GOAL

likes(judy, reading)

Или:
GOAL

likes(judy, tennis)

Ответ на поставленный вопрос система дает в логической форме – «Да» или «нет». Цель системы состоит в том, чтобы на поставленный вопрос найти возможность, исходя из базы знаний, ответить «Да». Вариантов ответить «Да» на поставленный вопрос может быть несколько. Система может (в нашем случае обучения - должна) быть настроена в режим получения всех возможных вариантов ответа «Да» на поставленный вопрос.

Факты, правила, и вопросы могут содержать переменные. Вообще – переменные предназначены для передачи значений «во времени и в пространстве». Переменные в факты и правила входят только с квантором всеобщности. А в вопросы переменные входят только с квантором существования. Поэтому символы кванторов в программе на Prolog не пишутся. Отметим, что именованные переменные уникальны в рамках предложения, а анонимная переменная – любая уникальна. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов.

В процессе выполнения программы переменные могут связываться с различными объектами – **конкретизироваться**. Это относится только к именованным переменным.

Анонимные переменные не могут быть связаны со значением.

Говорят, что именованная переменная конкретизирована, если имеется объект, который в данный момент обозначает данная переменная (переменная связанная). Иначе переменная – не конкретизированная (переменная свободная). Конкретизация не связана с распределением памяти и присваиванием переменной значения. В логическом программировании все переменные рассматриваются как безтиповые, т.е. в процессе вычисления любая переменная может быть связана с объектом произвольной природы. В логическом программировании поддерживается механизм деструктивной конкретизации переменной. Т.е. используется идея **ре-конкретизации** переменной путем «отката» вычислительного процесса и отказа от выполненной ранее конкретизации. Это реализовано для возможности поиска нового значения для именованной переменной.

Поиск содержательного ответа на поставленный вопрос, с помощью имеющейся базы знаний, фактически заключается в поиске нужного знания, но какое знание понадобится – заранее неизвестно. Этот поиск осуществляется формально с помощью механизма **унификации**, встроенного в систему и не доступного программисту. Упрощенно, процесс унификации можно представить как формальный процесс сравнивания (сопоставления) терма вопроса с очередным термом знания. При этом, знания по умолчанию просматриваются сверху вниз, хотя такой порядок и не очевиден. В процессе сравнивания для переменных «подбираются», исходя из базы знаний, значения (для именованных переменных). И эти подобранные для переменных значения возвращаются в качестве побочного эффекта ответа на поставленный вопрос. Остается только один вопрос: как установить, что с помощью конкретного знания можно подтвердить истинность вопроса и как «подбираются» значения для переменных. Это выполняется по формальным правилам с помощью сопоставления термов

Ведем определения:

Пусть дан терм: $\mathbf{A(X_1, X_2, X_n)}$

Подстановкой называется множество пар, вида: $\{ X_i = t_i \}$,

где X_i – переменная, а t_i – терм.

Пусть $\Theta = \{ X_1 = t_1, X_2 = t_2, \dots, X_n = t_n \}$ – подстановка, тогда результат применения подстановки к терму обозначается: $A\Theta$. Применение подстановки заключается в замене **каждого** вхождения переменной X_i на соответствующий терм.

Терм **B** называется **примером** терма **A**, если существует такая подстановка Θ, что $B=A\Theta$.

Терм **C** называется **общим примером** термов **A** и **B**, если существуют такие подстановки Θ_1 и Θ_2, что $C = A\,\Theta_1$ и $C=B\,\Theta_2$

В Prolog существует понятие процедуры. **Процедурой** называется совокупность правил, заголовки которых имеют одно и то же имя и одну и ту же арность (местность), т.е. это совокупность правил, описывающих одно определенное отношение. Отношение, определяемое процедурой, называется **предикатом**.

Логический вывод. Простейшие правила логического вывода, когда программа состоит только из фактов.

Цель работы программы – определить является ли вопрос логическим следствием программы или нет, что выполняется с применения правил вывода. Правила вывода – это утверждения о взаимосвязи между допущениями и заключениями, которые с позиции

исчисления предикатов верны всегда.

Если программа состоит только из фактов, то может быть только 4 случая:

1. Факты в базе и вопрос – основные. В этом случае используется простое правило: совпадение - вопрос из программы выводится, если в программе есть тождественный факт;
2. Факты в базе основные, а вопрос – неосновной. В этом случае используется простое правило: обобщение (факта) – если есть такая подстановка, что вопрос QΘ логически следует из программы, то и вопрос Q следует из программы.
3. Факты в базе неосновные, а вопрос – основной. В этом случае используется простое правило: конкретизация (факта) – из утверждения P с квантором всеобщности выводится пример PΘ, которым является вопрос.
4. Факты в базе и вопрос – неосновные. Это наиболее общий случай. Для ответа необходимо найти общий пример для вопроса и факта. В этом случае используется два правила: : конкретизация (факта) – строится пример, а затем, с помощью правила – обобщение, из примера выводится вопрос. Например
   Факт: P(Y , b).

   конкретизация➔ P(a, b). обобщение➔ P(a, X)

   Вопрос: P(a , X)

Существенным недостатком в использовании этих правил вывода является необходимость «угадать» подстановку или пример терма. Кроме этого, переменные в факте и в вопросе могут стоять на одной позиции. Поэтому для выполнения логического вывода используется **механизм (алгоритм) унификации**, встроенный в систему.

Унификация – операция, которая позволяет формализовать процесс логического вывода (наряду с правилом резолюции). С практической точки зрения - это основной вычислительный шаг, с помощью которого происходит:

- Двунаправленная передача параметров процедурам,
- Неразрушающее присваивание,
- Проверка условий (доказательство).

В процессе работы система выполняет большое число унификаций. Процесс унификации запускается автоматически, но пользователь имеет право запустить его принудительно с помощью утверждения (немного нарушает форму записей): T1 = T2 . Унификация – попытка "увидеть одинаковость" – сопоставимость двух термов, может завершаться успехом или тупиковой ситуацией (неудачей). В последнем случае включается механизм отката к предыдущему шагу.

Итак, два терма **T1 и T2 унифицируемы успешно, если**:

1) T1 и T2 – одинаковые константы;

2) T1 – не конкретизированная переменная, а T2 - константа или составной терм, не содержащий в качестве аргумента T1. Тогда унификация успешна, причем T1 конкретизируется значением T2;

3) T1 и T2 не конкретизированные переменные. Их унификация успешна всегда, причем T1 и T2 становятся сцепленными переменными – т.е. двумя именами одного "объекта", возможно пока и не установленного. В этом случае при конкретизации одной переменной вторая получает такое же значение.

4) T1 и T2 составные термы. Унификация успешна при выполнении трех условий:

а) T1 и T2 имеют одинаковые главные функторы,

б) T1 и T2 имеют одинаковые арности (количество аргументов),

в) успешно унифицируется каждая пара их соответствующих компонент (аргументов).