Лекция 8. Логическое программирование. Введение

Ключевые понятия: ***связь***, ***отношение, атом, переменная, цель, правило***

Вообще говоря, развитие языков программирования происходит на пути перехода от языков низкого уровня, в которых программист определяет, как должны быть выполнены те или иные действия, к языкам высокого уровня, позволяющим просто указать, что должно быть сделано. С появлением языка Prolog, наметился четко выраженный переход к использованию языков декларативного типа, которые побуждают программиста описывать ситуации и формулировать задачи, а не регламентировать во всех подробностях процедуры решения этих задач.

1 Язык Prolog: основные определения

Prolog - это язык программирования, сосредоточенный вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидное представление структур данных и автоматический перебор с возвратами. Этот ограниченный набор средств образует достаточно мощную и гибкую среду программирования. Prolog особенно хорошо подходит для решения задач, в которых рассматриваются объекты (в частности, структурированные объекты) и отношения между ними. Например, на языке Prolog совсем не сложно выразить пространственные связи между объектами, допустим, указать, что синий шар находится за зеленым. Столь же просто определить можно более общее правило: если объект X ближе к наблюдателю, чем объект Y, а Y ближе, чем Z, то X должен быть ближе, чем Z. После этого система Prolog получает возможность формировать рассуждения о пространственных связях и их совместимости с общим правилом. Благодаря таким особенностям Prolog становится мощным языком для искусственного интеллекта и нечислового программирования в целом.

Prolog стал воплощением идеи использования логики в качестве языка программирования, которая зародилась в начале 1970-х годов, и само его название является сокращением от слов "programming in logic" (программирование в терминах логики). Первыми исследователями, которые занялись разработкой этой идеи, были Роберт Ковальски (теоретические основы) и Маартен ван Эмден из Эдинбурга (экспериментальная демонстрационная система), а также Ален Колмероэ (реализация). Популяризации языка Prolog во многом способствовала эффективная реализация этого языка в середине 1970-х годов Дэвидом Д. Г. Уорреном. К числу новейших достижений в этой области относятся средства программирования на основе логики ограничений (Constraint Logic Programming - CLP), которые обычно реализуются в составе системы Prolog.

Средства CLP показали себя на практике как исключительно гибкий инструмент для решения задач составления расписаний и планирования материально-технического снабжения. А в 1996 году был опубликован официальный стандарт ISO языка Prolog.

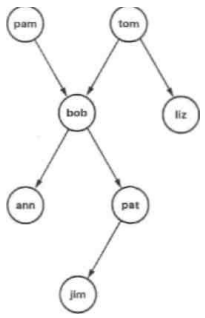
В развитии языка Prolog наблюдаются очень интересные тенденции. Этот язык быстро приобрел популярность в Европе как инструмент практического программирования. В Японии вокруг языка Prolog были сосредоточены все разработки компьютеров пятого поколения. В США этот язык в целом был принят с небольшим опозданием в связи с некоторыми историческими причинами и наличием консервативной школы информатики.

2 Определение отношений на основе фактов

Prolog - это язык программирования для символических, нечисловых вычислений. Он особенно хорошо приспособлен для решения проблем, которые касаются объектов и отношений между объектами. На рис. 1.1 приведен подобный пример: семейные отношения. Тот факт, что Том является одним из родителей Боба, можно записать на языке Prolog следующим образом:

parent (tom, bob).

В данном случае в качестве имени отношения выбрано слово parent; Tom и Bob являются параметрами этого отношения. По причинам, которые станут понятными позже, такие имена, как tom, записываются со строчной буквой в начале. Все дерево семейных отношений, показанное на рис. 1, определено с помощью следующей программы Prolog:

parent(pam, bob) parent(tom, bob). parent(tom, liz) parent(bob, ann) parent(bob, pat) parent(pat, jim) .	

Эта программа состоит из шести предложений, каждое из которых объявляет один факт об отношении parent. Например, факт parent(tom, bob) представляет собой конкретный экземпляр отношения parent. Такой экземпляр называют также *связью*. В целом *отношение* определяется как множество всех своих экземпляров.

После передачи соответствующей программы в систему Prolog последней можно задать некоторые вопросы об отношении parent, например, является ли Боб одним из родителей Пэт? Этот вопрос можно передать системе Prolog в следующем виде:

?- parent(bob, pat).

Обнаружив, что это - факт, о существовании которого утверждается в программе, Prolog отвечает: yes. После этого можно задать еще один вопрос: ?- parent(liz, pat). Система Prolog ответит отрицательно, поскольку в программе нет упоминания о том, что Лиз является одним из родителей Пэт. Система ответит также "no" на вопрос ?- parent(tom, ben), поскольку в программе даже не встречалось имя Бен.

Кроме того, системе можно задать более интересные вопросы. Например, кто является родителями Лиз?

?- parent(X, liz).

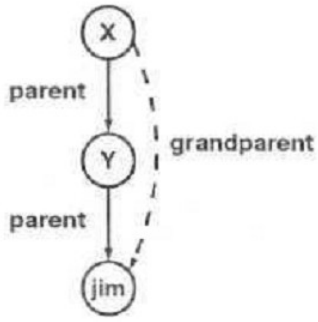
На этот раз Prolog ответит не просто "yes" или "no", а сообщит такое значение X, при котором приведенное выше утверждение является истинным. Поэтому ответ будет таковым: X = tom.

Вопрос о том, кто является детьми Боба, можно сообщить системе Prolog следующим образом: ?- parent(bob, X). На этот раз имеется больше одного возможного ответа. Система Prolog вначале выдаст в ответ одно решение — ann. Теперь можно потребовать у системы сообщить еще одно решение, введя точку с запятой) и Prolog найдет следующий ответ: X = pat. Если после этого будут затребованы дополнительные решения, Prolog ответит "no", поскольку все решения уже исчерпаны.

Этой программе может быть задан еще более общий вопрос о том, кто является чьим родителем? Этот вопрос можно также сформулировать иным образом: Найти X и Y, такие, что X является одним из родителей Y. Этот вопрос может быть оформлен на языке Prolog следующим образом: ?- parent(X, Y).

После этого Prolog начнет отыскивать все пары родителей и детей одну за другой. Решения будут отображаться до тех пор, пока Prolog получает указание найти следующее решение (в виде точки запятой) или пока не будут найдены все решения.

Программе, рассматриваемой в качестве примера, можно задать еще более сложный вопрос, например, спросить о том, кто является родителями родителей Джима (дедушками и бабушками). Поскольку в программе непосредственно не предусмотрено использование соответствующего отношения grandparent, этот запрос необходимо разбить на следующие два этапа (рис.2): 1) кто является одним из родителей Джима? Предположим, что это - некоторый объект Y. и 2) кто является одним из родителей Y. Предположим, что это - некоторый объект X.



Подобный сложный запрос записывается на языке Prolog как последовательность двух простых:

?- parent(Y, jim), parent (X,Y) .

Ответ должен быть следующим: X = bob, Y = pat

Этот составной запрос можно прочитать таким образом: найти такие X и Y, которые удовлетворяют следующим двум требованиям: parent(Y, jim) и parent(X,Y)

Если же будет изменен порядок следования этих двух требований, то логический смысл всего выражения останется тем же:

parent(X, Y) и parent(Y, jim)

Безусловно, такое же действие может быть выполнено и в программе Prolog, поэтому запрос:

?- parent(X, Y), parent(Y, jim) -выдаст тот же результат.

Аналогичным образом, программе можно задать вопрос о том, кто является внуками Тома:

?- parent(tom, Y), parent(Y, X)

Система Prolog ответит следующим образом:
x = bob, y = ann
x = bob, y = pat

Могут быть также заданы и другие вопросы, например, имеют ли Энн и Пэт общих родителей. Эту задачу также можно разбить на два этапа.

1. Кто является одним из родителей Энн (X)?
2. Является ли (тот же) X одним из родителей Пэт?

Поэтому соответствующий вопрос в языке Prolog выглядит следующим образом: ?- parent(X, ann), parent(x, pat)

Ответом на него является: x = bob.

Рассматриваемый пример программы позволяет проиллюстрировать перечисленные ниже важные понятия:

- В языке Prolog можно легко определить отношение, такое как parent, задавая n-элементные кортежи объектов, которые удовлетворяют этому отношению.
- Пользователь может легко запрашивать систему Prolog об отношениях,

определенных в программе.

• Программа Prolog состоит из предложений. Каждое предложение оканчивается точкой.

• Параметрами отношений могут быть (кроме всего прочего) определенные объекты, или константы (такие как torn и ann), а также объекты более общего характера (такие как X и Y). Объекты первого типа, применяемые в рассматриваемой программе, называются ***атомами.*** Объекты второго типа называются ***переменными,***

• Вопросы к системе состоят из одной или нескольких *целей.* Последовательность целей, такая как parent(X, ann), parent(X, pat) означает конъюнкцию целей: «X является одним из родителей Энн» и «X является одним из родителей pat». Слово "цель" (goal) используется для обозначения таких вопросов потому, что система Prolog воспринимает вопросы как цели, которых необходимо достичь.

• Ответ на вопрос может быть положительным или отрицательным, в зависимости от того, может ли быть достигнута соответствующая цель или нет. В случае положительного ответа считается, что соответствующая цель была достижимой и что цель достигнута. В противном случае цель была недостижимой и не достигнута.

• Если вопросу соответствует несколько ответов, Prolog отыскивает столько ответов, сколько потребует пользователь (в пределах возможного).

3 Определение отношений на основе правил

Рассматриваемый пример программ можно легко дополнить с применением многих интересных способов. Вначале введем информацию о мужском или женском поле людей, участвующих в отношении parent. Эту задачу можно решить, добавив следующие факты к программе:

female (pam), male(tom), male(bob), female(liz), female(pat), female(ann), male(jim).

В этом случае введены отношения male и female. Эти отношения являются унарными (или одноместными). Бинарные отношения типа parent определяют связь между парами объектов. С другой стороны, унарные отношения могут использоваться для объявления простых свойств объектов, которые они могут иметь или не иметь. Первое из приведенных выше унарных предложений можно прочитать таким образом: Пэм - женщина. Вместо этого информацию, объявленную в двух унарных отношениях, можно передать с помощью одного бинарного отношения. В таком случае приведенная выше часть программы примет примерно такой вид:

sex(pam, feminine), sex(tom, masculine), sex(bob, masculine)

В качестве следующего дополнения к программе введем отношение offspring (отпрыск), обратное отношению parent. Отношение offspring можно определить таким же образом, как и parent, предоставив список обычных фактов об отношении offspring, и в качестве каждого факта указать такую пару

людей, что один из них является сыном или дочерью другого, например: offspring(liz, tom).

Но отношение offspring можно определить гораздо более изящно, используя то, что оно является противоположным parent и что parent уже было определено. Этот альтернативный способ может быть основан на следующем логическом утверждении:

Для всех X и Y, Y является сыном или дочерью X, если X является родителем Y.

Такая формулировка уже более близка к синтаксису языка Prolog. Соответствующее предложение Prolog, которое имеет тот же смысл, выглядит таким образом:

offspring(Y, X) :- parent(X, Y)

Это предложение можно также прочитать так:

Для всех X и Y, если X является родителем Y, то Y является сыном или дочерью X.

Предложения Prolog, такие как offspring(Y, X) :- parent(X, Y) называются *правилами.* Между фактами и правилами существует важное различие.

Такие факты, как parent(tom, liz) представляют собой логические утверждения, которые всегда и безусловно являются истинными. С другой стороны, правила представляют собой утверждения, которые становятся истинными, если удовлетворяются некоторые условия. Поэтому принято считать, что правила состоят из следующих частей:

- условие (правая часть правила);
- заключение (левая часть правила).

Часть, соответствующая заключению, называется также *головой предложения,* а часть, соответствующая условию, - *телом предложения,*

Если условие parent(X,Y) является истинным, то его логическим следствием становится offspring(Y, X).

Применение правил в языке Prolog иллюстрируется в следующем примере. Зададим программе вопрос, является ли Лиз дочерью Тома: ?- offspring(liz, torn).

Поскольку в программе отсутствуют факты о дочерях и сыновьях, единственным способом поиска ответа на этот вопрос является использование правила с определением отношения offspring. Данное правило является общим в том смысле, что оно применимо к любым объектам X и Y; но его можно также применить и к таким конкретным объектам, как liz и torn. Чтобы применить правило к объектам liz и com, вместо Y необходимо подставить liz, а вместо X - torn. Это действие называется *конкретизацией переменных* (в данном случае - X и Y), которое выполняется следующим образом: X = tom и Y =Iiz

После конкретизации будет получен частный случай общего правила, который выглядит следующим образом: offspring(liz, tom) :- parent(tom, liz).

Часть с обозначением условия принимает вид parent(tom, liz)

После этого система Prolog пытается определить, является ли истинной часть с обозначением условия. Поэтому первоначальная цель: offspring(liz, torn) заменяется подцелью: parent(tom, liz)

Оказалось, что задача достижения этой (новой) цели является тривиальной, поскольку ее можно найти как факт в рассматриваемой программе. Это означает, что часть данного правила с обозначением заключения также является истинной, и Prolog в качестве ответа на вопрос выводит yes.

Теперь введем в рассматриваемый пример программы еще некоторую информацию о семейных отношениях. Определение отношения mother может быть основано на следующем логическом утверждении:

Для всех X и Y, X является матерью Y, если:

X является одним из родителей Y и X - женщина,

Это утверждение можно перевести на язык Prolog в виде следующего правила:

mother(X, Y) :- parent(X,Y), female(X).

Запятая между двумя условиями указывает на конъюнкцию этих условий; это означает, что оба условия должны быть истинными.

Такие отношения, как parent, offspring и mother, можно проиллюстрировать с помощью схем, подобных приведенным на рис. 3. Эти схемы соответствуют следующим соглашениям. Узлы графов относятся к объектам, т.е. параметрам отношений. Дуги между узлами соответствуют бинарным (или двухместным) отношениям. Дуги направлены от первого параметра отношения ко второму. Унарные отношения обозначаются на схемах путем проставления отметки на соответствующих объектах с именем отношения. Отношения, которые определены на основе других отношений, представлены в виде пунктирных дуг. Поэтому каждую схему необходимо интерпретировать следующим образом: если соблюдаются отношения, обозначенные сплошными дугами, то соблюдаются и созданные на их основе отношения, обозначенные пунктирными дугами. Согласно рис. 3, отношение grandparent можно непосредственно записать на языке Prolog следующим образом:

grandparent(X, Z) :- parent(X, Y),parent(Y, Z).

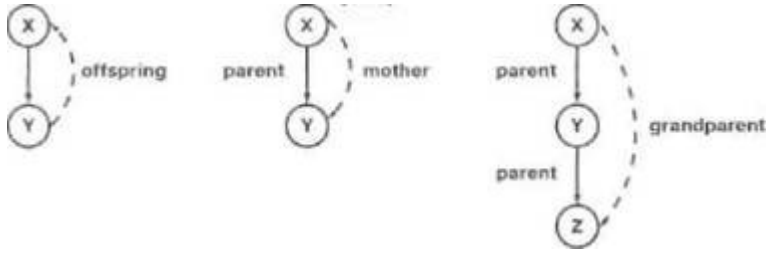


Рис. 3. Графы, которые определяют отношения offspring, mother и grandparent в терминах других отношений

На данном этапе необходимо кратко рассмотреть вопрос о компоновке

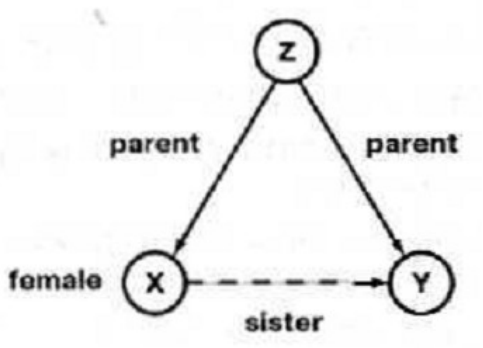
программ. Система Prolog предоставляет почти полную свободу выбора компоновки программ. Однако чаще всего голова предложения и каждая цель в его теле записываются на отдельной строке. Например:

grandparent(X, Z) :- parent(X,Y), parent (Y, Z) .

Схема отношения sister (рис4) имеет следующее определение:

Для любого X и Y: X является сестрой Y, если

1) X и Y имеют общего родителя и 2) X - женщина.



Рис, 4. Определение отношения sister

Граф, представленный на рис. 4, можно перевести на язык Prolog следующим образом:

sister(X,Y) :- parent (Z, X), parent(Z,Y), female(X).

Обратите внимание на то, каким способом было выражено требование "X и Y имеют общего родителя". Для этого использовалась следующая логическая формулировка: некоторый Z должен быть родителем X, и тот же *Z* должен быть родителем Y. Альтернативный, но менее изящный способ мог предусматривать использование следующей цепочки утверждений: Z1 является родителем X, Z2 является родителем Y и Z1 равно Z2.

Теперь системе можно задать вопрос: ?- sister (ann, pat) и ответом должно быть "yes". Поэтому можно сделать вывод, что отношение suster в том виде, в каком оно определено, действует правильно. Но в рассматриваемой программе имеется незаметный на первый взгляд недостаток, который обнаруживается при получении ответа на вопрос о том, кто является сестрой Пэт: ?- sister(X, pat) .

Prolog находит два ответа: X = ann; X = pat. Т.е. pat является сестрой самой себя. Согласно правилу, касающемуся сестер, ответ системы Prolog является полностью обоснованным. В правиле о сестрах нет упоминания о том, что X и Y не должны быть одинаковыми, если X рассматривается как сестра Y. Поскольку это требование не предъявляется, система Prolog (вполне обоснованно) предполагает, что X и Y могут быть одинаковыми, и поэтому приходит к заключению, что любая женщина, имеющая родителя, является сестрой самой себя.

Чтобы исправить приведенное выше правило о сестрах, необходимо дополнительно указать, что X и Y должны быть разными - предположим, что

системе Prolog известно отношение different. Уловие different (X, Y) удовлетворяется, если и только если X и Y не равны. Поэтому усовершенствованное правило для отношения sister может выглядеть следующим образом:

sister (X, Y) :- parent(Z, X), parent(Z,Y), female(X), different(X,Y).

На основании изложенного можно сделать следующие выводы:

- Программы Prolog можно дополнять, вводя новые предложения.
- Предложения Prolog относятся к трем типам: факты, правила и вопросы.
- С помощью фактов можно вводить в программу сведения, которые всегда и безусловно являются истинными.
- С помощью правил можно вводить в программу сведения, которые являются истинными в зависимости от заданного условия,
- Задавая программе вопросы, пользователь может узнавать, какие сведения являются истинными.
- Предложения языка Prolog состоят из головы и тела. Тело представляет собой список целей, разделенных запятыми. Запятые рассматриваются как знаки конъюнкции.
- Факты представляют собой предложения, которые имеют голову и пустое тело. Вопросы имеют только тело. Правила имеют голову и непустое тело.
- В процессе вычисления переменные можно заменять другими объектами. В таком случае переменная становится конкретизированной.
- Предполагается, что на переменные распространяется действие квантора всеобщности, который имеет словесное выражение "для всех". Но если переменные появляются только в теле, их можно трактовать несколькими способами. Например, предложение

hasachild(X) :- parent (X, Y)

можно прочитать двумя приведенными ниже способами:

а) Для всех X и Y, если X является родителем Y, то X имеет ребенка.

б) Для всех X, X имеет ребенка, если существует некоторый Y, такой, что X является родителем Y.