СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В АПК

приоритет2030
лидерами становятся

ТЕМА:
Основные принципы логического программирования

КОНСПЕКТ ЛЕКЦИИ

Преподаватель:

Аникуев Сергей Викторович
к.т.н., доцент, доцент кафедры электротехники, автоматики и метрологии.

Тема 2.3. Основные принципы логического программирования

Вопрос 1. Логическая программа. Основные конструкции

Вопрос 2. Логические переменные, подстановки и примеры

1. Логическая программа: основные конструкции

Логическая программа – это множество аксиом и правил, задающих отношения между объектами. Вычислением логической программы является вывод следствий из программы. Программа задает множество следствий, которое и представляет собой значение программы. Искусство логического программирования состоит в построении ясных и изящных программ с требуемым значением.

Основные конструкции логического программирования – термы и утверждения – заимствованы из логики. Имеются **три основных вида утверждений: факты, правила и вопросы.** Имеется единственная **структура данных: логический терм.**

Факты

Простейшим видом утверждения является **факт**. Факты используются для констатации того, что выполнено некоторое отношение между объектами. Например:

отец(авраам, исаак).

Этот факт утверждает, что Авраам является отцом Исаака или, что выполнено отношение отец между индивидами, названными авраам и исаак. Другое название для отношения – **предикат.** Имена индивидов называются **атомами**. Аналогично факт *плюс*(2,3,5) означает, что имеет место отношение «2 плюс 3 равно 5» Известно отношение *плюс* может быть задано с помощью множества фактов определяющих таблицу сложения. Вот начальный фрагмент таблицы:

плюс(0,0,0),	плюс(0,1,1),	плюс(0,2,2),	плюс(0,3,3).
плюс(1,0,1),	плюс(1,1,2),	плюс(1,2,3),	плюс(1,3,4).

```
отец(фарра, авраам).        мужчина(фарра).
отец(фара,нахор).           мужчина(авраам).
отец(фарра,аран).           мужчина(нахор).
отец(авраам,исаак).         мужчина(аран).
отец(аран,лот)              мужчина(исаак).
отец(аран,милка).           мужчина(лот).
отец(аран,иска).

                            женщина(сара).
мать(сара,исаак).           женщина(милка).
                            женщина(иска).
```

Программа 1.1. База данных библейской семьи.

Конечное множество фактов образует программу. Это простейший вид логической программы. Множество фактов, с другой стороны, составляет описание ситуации. Такой подход лежит в основе

программирования баз данных, рассматриваемого в следующей главе. Пример базы данных семейных отношений в Библии приведен в программе 1.1. Предикаты отец, мать, мужчина и женщина выражают очевидные отношения.

Вопросы

Второй формой утверждения в логической программе является **вопрос.** Вопрос – это средство извлечения информации из логической программы. С помощью вопроса выясняется, выполнено ли некоторое отношение между объектами. Например, с помощью вопроса *отец (авраам,исаак)?* выясняется, верно ли, что выполнено отношение *отец* между объектами *авраам и исаак?* Исходя из фактов программы 1.1, ответ на этот вопрос – да.

Синтаксически вопросы и факты выглядят одинаково, однако их можно различить по контексту. В тех случаях, когда возможна неоднозначность, конец факта будет обозначаться точкой, а конец вопроса – вопросительным знаком. Объект без точки и вопросительного знака мы называем целью. факт *P.* утверждает, что цель *P* является истинной. В вопросе *P?* спрашивается, является ли истинной цель *P.* **Простой вопрос** состоит из одной цели.

Что касается программы, то поиск ответа на вопрос состоит в том, чтобы определить, является ли вопрос логическим следствием программы. Логические следствия выводятся путем применения правил. Простейшее правило вывода **совпадение:** из P выводимо P. Вопрос является логическим следствием тождественного с ним факта.

Если используется программа, содержащая факты, подобно программе 1.1, то процедура поиска ответа на простой вопрос выполняется непосредственно. В программе ищется факт, предполагаемый в вопросе. Если факт, тождественный вопросу, найден, то ответ – *да*.

Ответ *нет* дается в том случае, когда факт, тождественный вопросу, не найден, поскольку данный факт не является логическим следствием программы. Такой ответ не подвергает сомнению истинность вопроса, он просто показывает, что нам не удалось доказать истинность вопроса с помощью программы. На оба вопроса – *женщина (авраам)?* и *плюс(1,0,2)?* программа 1.1 ответит *нет*.

2. Логические переменные, подстановки и примеры

Логические переменные служат для обозначения неопределенных объектов и соответственно этому используются. Рассмотрим их применение в вопросах. Предположим, мы хотим выяснить, чьим отцом был авраам. Для этого можно задавать вопросы: *отец (авраам, лот)?*, *отец (авраам, милка)?...*, *отец (авраам, исаак)?,...* до тех пор, пока не будет получен ответ да. С помощью переменной реализуется более удобный способ поиска ответа. Вопрос формулируется в виде *отец (авраам, X)?*. Ответом на вопрос

является *X = исаак. Переменные в данном случае служит для представления совокупности вопросов.* Как будет объяснено в дальнейшем, с помощью вопроса, содержащего переменную, выясняется, имеется ли такое значение переменной, при котором вопрос будет логическим следствием программы.

Использование переменных в логических программах отличается от использования переменных в традиционных языках программирования. В логических программах переменная обозначает неопределенный, но единственный объект, а не некоторую область памяти.

Введя переменные, мы можем определить единственную структуру данных в логических программах – **термы**. Определение термов индуктивно. Константы и переменные являются термами. Кроме того, термами являются составные термы, или структуры. **Составной терм** содержит **функтор** (называемый главным функтором терма) и последовательность из одного или более аргументов, являющихся термами. Функтор задается своим именем, которое суть атом, и своей **арностью**, или числом аргументов. Синтаксически составные термы имеют вид $f(t_1, t_2, \ldots t_n)$,где f – имя n-арного функтора, а t_i – аргументы. Примеры составных термов: *s(0), горячий (молоко), имя (джон, доу), list(a, list (b, nil)), foo(X) и tree(tree(nil,3,nil), 5,R).*

Вопросы, цели и вообще термы, в которые не входят переменные, называются **основными.** Термы, содержащие переменные, называются **неосновными**. Так, *foo (a,b)* – основной терм, а *bar(X)* – нет.

• **Определение. Подстановкой** называется конечное (возможно, пустое) множество пар вида $X_i = t_i$ где X_i – переменная и t_i – терм; $X_i \neq X_j$, при $i \neq j$ и X_i не входит в t_j при любых i и j. •

Примером подстановки, состоящей из одной пары, может служить {X = *исаак*}. Подстановки могут применяться к термам. Результат применения подстановки Θ к терму A, обозначается AΘ, есть терм, полученный заменой каждого вхождения переменной X в A на t для каждой пары вида $X = t$ из Θ.

Результат применения {X = *исаак*} к терму *отец (авраам, X)* есть терм *отец(авраам, исаак).*

• **Определение**: A называется **примером** B, если найдется такая подстановка Θ, что A = BΘ. •

По определению цель *отец (авраам, исаак)* является примером терма *отец (авраам, X).* Аналогично *мать(сара, исаак)* – пример терма *мать (X, Y)* при подстановке *{X = сара, Y= исаак}.*

Экзистенциальные вопросы

В логических терминах переменные в вопросах связаны квантором существования это означает на интуитивном уровне, что вопрос *отец(авраам, X)?* следует читать: «Существует ли такое X, что *авраам* является отцом X». В общем случае вопрос $P(T_1 T_2, \ldots, T_n)$?, содержащий

переменные $X_1, X_2, ..., X_k$. означает следующее: «Существуют ли такие $X_1, X_2, ..., X_k$, что $p(T_1, T_2, \ldots, T_n)$». Для удобства квантор существования обычно не пишется.

Введем еще одно правило вывода – **обобщение**: при любой подстановке Θ экзистенциальный вопрос P логически следует из примера PΘ. Из факта *отец(авраам, исаак)* следует существование такого X, что истинно *отец(авраам, X)*, а именно *X = исаак*.

Процедура поиска ответа на экзистенциальный неосновной вопрос при использовании программы, состоящей из фактов, сводится к поиску факта, являющегося примером вопроса. Ответом, или *решением* , будет такой пример. Ответ *нет* дается, если подходящего факта нет в программе.

Поиск ответа на неосновной вопрос представляет собой вычисление, выходом которого является пример вопроса. Иногда мы изображаем этот пример с помощью подстановки, применение которой к вопросу и дает решение.

Экзистенциальный вопрос в общем случае может иметь несколько решений. Из программы 1.1 ясно, что Аран – отец троих детей. Следовательно, вопрос *отец(аран, X)*? имеет решения {*X=лот*}, {*X=милка*}, {*X=иска*}. Другим вопросом, обладающим множеством решений, является *плюс(X,Y,4)*?, в котором ищутся числа, дающие в сумме 4. Решениями, например, будут {X = 0, У = 4} и {X = 1, Y= 3}. Обратите внимание, что различным переменным X и Y могут соответствовать различные объекты.

Интересным вариантом последнего вопроса является *(плюс X, X, 4)?*, в котором требуется, чтобы два числа, дающие в сумме 4, совпадали. Имеется единственный ответ – {X = 2}.

Универсальные факты

Переменные также полезны и в фактах. Предположим, что все библейские персонажи любят гранаты. Вместо того чтобы включать соответствующий факт для каждого индивида:

любит(авраам, гранаты).

любит(сара, гранаты).

все это можно выразить универсальным фактом *любит(X, гранаты)*. В данном случае *переменные позволяют выразить совокупность многих фактов*. Факт *умножить(0, X, 0)* объединяет все факты, утверждающие, что 0, умноженный на любое число, дает 0.

Переменные в фактах неявно связаны квантором общности; это на интуитивном уровне означает, что факт *любит(X, гранаты)* утверждает, что для всех X справедливо: X любит гранаты. В общем случае факт $p(T_1, T_2, \ldots, T_n)$. следует понимать так. что при любых значениях переменных $X_1, ..., X_k$, где X_i – переменные, входящие в факт $p(T_1, T_2, \ldots, T_n)$. выполнено.

Естественно, из факта с квантором общности можно вывести любой пример факта. Например, из *любых(X, гранаты)* следует *любит(авраам, гранаты)*.

Это – третье правило вывода, называемое **конкретизацией**, из утверждения *P* с квантором общности выводится пример *P*Θ при любой подстановке Θ.

Как и в случае вопроса, можно добиться, чтобы два неопределенных объекта, обозначенных переменными, совпадали – для этого нужно использовать имя одной и той же переменной. Факт *плюс(0, X, X)* означает, что 0 является левой единицей по сложению. Это следует понимать так, что при всех значениях X, 0 плюс X равно X. Аналогичное использование переменных возникает при переводе фразы «каждый любит себя» в факт *любит(X, X)*.

Поиск ответа на основной вопрос при использовании факта с квантором общности происходит непосредственно. Ищется факт, для которого вопрос является примером. Например, ответом на вопрос *плюс(0, 2, 2)* на основе факта *плюс(0, X, X)* будет *да*. Поиск ответа на неосновной вопрос при использовании неосновных фактов требует нового понятия: общий пример двух термов.

•**Определение.** C называется **общим примером** термов A и B. если C есть пример A и C есть пример B. Иными словами, если найдутся такие подстановки Θ_1 и Θ_2, что $C = A\Theta_1$ синтаксически совпадает с $B\Theta_2$.•

Например, цели *плюс(0, 3, Y)* и *плюс(0, X, X)* имеют общий пример *плюс(0, 3, 3)*. Применения подстановки {Y = 3} к *плюс(0, 3, Y)* и подстановки {X = 3} к *плюс(0, X, X)* приводят к *плюс(0, 3, 3)*.

В общем случае при поиске ответа на вопрос с использованием факта ищется общий пример вопроса и факта. Если общий пример существует, то он и будет, ответом. В противном случае ответ – *нет*.

Ответ на экзистенциальный вопрос на основе универсального факта с использованием общего примера требует двух логических выводов. Факт выводится из примера с помощью правила конкретизации, а пример выводится из вопроса с помощью правила обобщения.

Конъюнктивные вопросы и общие переменные

Важным обобщением обсуждавшихся до сих пор вопросов являются *конъюктивные вопросы* . Конъюнктивные вопросы – это конъюнкция целей, поставленная в виде вопроса, например:*отец(фара, X), отец(X, Y)* или в общем виде: $Q_1 \ldots Q_n$. Простые вопросы – это частный случай конъюнктивных вопросов с единственной целью. Логически вопрос состоит в выводимости конъюнкции из программы. Мы всюду используем «,» для обозначения логического «и». Не следует путать запятую, отделяющую аргументы цели, с запятой, отделяющей цели и обозначающей конъюнкцию.

В простейшем конъюнктивном вопросе все цели простые, например,

отец (авраам, исаак), мужчина(лот)? Ясно, что ответ на этот вопрос, исходя из программы 1.1, – *да*, так как обе цели являются фактами программы. Вообще на вопрос $Q_1 ... Q_n$ *?* в котором каждое Q_i, – основная цель, ответом, исходя из программы *P* будет *да*, если каждое Q_i выводится из *P*. Таким образом, конъюнкция основных вопросов не очень интересна.

Конъюнктивные вопросы представляют интерес в тех случаях, когда имеются одна или более **общие переменные**, т.е. переменные, входящие в две различные цели вопроса. Примером служит вопрос *отец (аран, X), мужчина(X)?* Областью переменной в конъюнктивном вопросе является вся конъюнкция. Таким образом, вопрос *p(X),q(X)?* означает: «Существует ли такое X, что *p(X) и q(X) одновременно?*» Как и в простых вопросах, переменные в конъюнктивных вопросах неявно связаны квантором существования.

Общие переменные используются для ограничения простых вопросов путем сужения области значения переменных. Мы уже рассматривали пример с вопросом *плюс(X, X, 4)*?, в котором поиск чисел, дающих в сумме 4, был ограничен тем, что числа должны быть равными. Рассмотрим вопрос *отец(аран, X), мужчина(X)*? В этом случае решения вопроса *отец(аран, X)?* ограничены детьми мужского пола. Программа 1.1 показывает, что существует единственное решение {X = лот}. С другой стороны, данный вопрос можно рассматривать как ограничение решений вопроса *мужчина(X)*? множеством индивидов, отцом которых является Аран.

Несколько иное использование общих переменных демонстрируется в вопросе *отец(фарра, X), мужчина(X, Y)*?. С одной стороны, вопрос ограничивает сыновей Фарры теми, кто сам является отцом. С другой стороны, ищутся индивиды, чьи отцы были сыновьями Фарры. Имеется несколько решений, например: {X = авраам, Y = исаак} и {X = *аран*, Y= *лот*}.

Конъюнктивный вопрос логически следует из программы P, если все цели – следствия P, причем общим переменным в разных целях сопоставлено одно и то же значение. Достаточное условие выводимости состоит в существовании основного примера вопроса, являющегося следствием *P*. Из такого примера вопрос может быть получен с помощью обобщения.

Процедура поиска решения конъюнктивного вопроса $A_1, A_2, ..., A_n$ с помощью программы P состоит в нахождении такой подстановки Θ, что $A_1\Theta$ и $...A_n\Theta$ будут основными примерами фактов из P. То, что одна и та же подстановка применяется ко всем целям, обеспечивает единое соответствие переменным в вопросе. Рассмотрим, например, вопрос *отец(аран,X)?мужчина(X)?* при использовании программы 1.1. Применение подстановки {*X = лот*} к вопросу дает основной пример

отец(аран,лот),мужчина(лот), который является следствием программы.

Правила

Интересные конъюнктивные вопросы сами по себе определяют отношения. Вопрос *отец(аран,X),мужчина(X)?* выражает свойство быть сыном Арана. Вопрос *отец(фарра,X),отец(X,Y)*? относится к внукам Фарры. Так мы приходим к третьему и самому важному виду утверждения в логическом программировании – к правилу, позволяющему определять новые отношения в терминах существующих отношений.

Правила – это утверждения вида

$A \leftarrow B_1, B_2, \ldots B_n$

где $n \geq 0$. A называется **заголовком** правила, а последовательность B_i – **телом** правила. Как A, так и B_i; должны быть целями. Правила, факты и вопросы называются также **хорновыми предложениями** или просто **предложениями.** Заметим, что факт – это частный случай правила при n=0. Факты называются также **единичными предложениями**. Кроме того, имеется специальное название для правил, тело которых состоит из одной цели, т.е. для случая n = 1. Такие предложения называются **итерационными предложениями**. В правилах, как и в фактах, переменные связаны кванторами общности, область действия кванторов – все правило.

Правило, выражающее свойство быть сыном:

сын(X,Y)←отец(Y,X),мужчина(X)

Аналогично можно определить свойство быть дочерью:

дочь(X,Y) ← отец(Y,X),женщина(X).

Правило для отношения быть дедушкой:

дедушка(X,Z) ← отец(X,Y),отец(Y,Z).

Возможны два способа интерпретации правил. При первом правила используются для построения новых и сложных вопросов в терминах простых вопросов. Вопрос *сын(X,аран)?* в программе, содержащей приведенное выше правило для отношения *сын*, переводится в соответствии с этим правилом в вопрос *отец(аран,X),мужчина(X)?* и решается как и раньше. Новый вопрос, включающий отношение *сын*, строится из простых вопросов, содержащих отношения *отец* и *мужчина*. Такая интерпретация правил сводится к процедурному истолкованию правил. Процедурное истолкование правила для отношения *дедушка* состоит в следующем: «Для ответа на вопрос, *является ли X дедушкой Y* ответьте на конъюнктивный вопрос, является .ли X *отцом* Z и Z *отцом* Y».

Второй способ основан на интерпретации правил как логических аксиом. Обратная стрелка ← используется для обозначения логического следования. Правило для отношения *сын* понимается так: «X – сын Y если Y – отец X и X – мужчина». Здесь правила используются для построения

новых или сложных отношений на основе других, более простых отношений. Предикат *сын* определяются через предикаты *отец* и *мужчина*. Соответствующее понимание правила известно как декларативное понимание. Декларативное понимание правила для отношения дедушка состоит в следующем: «Для всех X, Y и Z, если X – отец Z и Z – отец Y, то X – дедушка Y».

Хотя формально все переменные в предложении связаны квантором общности, мы будем иногда говорить о переменных, входящих лишь в тело, а не в заголовок предложения, как если бы они были связаны в теле квантором существования. Например, правило для отношения *дедушка* можно понимать так: «Для всех X и Y, X – дедушка Y, если найдется такое Z, что X – отец Z и Z – отец Y». Мы не приводим обоснования подобной словесной трансформации и рассматриваем ее как некоторое удобство чтения.

Для введения правил в наше рассмотрение логического вывода потребуется закон modus ponens. Modus ponens утверждает, что из B и A ← B следует A.

• **Определение.** Обобщенный закон modus ponens утверждает, что из правила

R=($A \leftarrow B_1, B_2, \ldots B_n$) и фактов

B_1'

B_2'

...

B_n'

выводится A' при условии, что

$A' \leftarrow B_1', B_2', \ldots, B_n'$

является примером R.•

Правила совпадения и конкретизации являются частными случаями обобщенного закона modus popens.

Теперь все готово для полного определения понятия логической программы и соответствующего понятия логического следствия.

• **Определение.** Логическая программа – это конечное множество правил.•

• **Определение.** Цель G с кванторами существования является логическим следствием программы P, если в P найдется такое предложение с основным примером $A \leftarrow B_1, B_2, \ldots B_n$, $n \geq 0$, $B_1, \ldots, B_n$ – логические следствия P и A – пример G. •

Заметим, что цель C логически следует из программы P тогда и только тогда, когда G может быть выведена из Y с помощью конечного числа применений обобщенного правила modus ponens.

Рассмотрим вопрос *сын(S,аран)?* относительно программы 1.1, расширенной правилом для отношения *сын*. Применение подстановки {X = *лот*, Y = *аран*} к данному правилу приводит к примеру *сын(лот,аран)←отец(аран,лот), мужчина(лот)*. Обе цели в теле правила являются фактами программы 1.1. Таким образом, из обобщенного modus ponens следует ответ *S* = *лот*.

Процедура поиска ответа на вопрос отражает определение логического следования. Угадываются основной пример цели и основной пример правила, далее рекурсивно ищется ответ на конъюнктивный вопрос, соответствующий телу этого правила. Доказательство цели A в программе P сводится к выбору правила $A \leftarrow B_1, B_2, \ldots B_n$ в P и указанию такой подстановки Θ, что A = A_1Θ и B_iΘ – основные цели при $1 \le i \le l$. Далее рекурсивно доказывается каждая цель B_iΘ. В такой процедуре могут возникать сколь угодно длинные цепочки доказательств. В общем случае угадать нужный основной пример и выбрать должное правило сложно.

Правило, сформулированное для описания отношения сын, корректно, но не является полным описанием соответствующего понятия. Мы не сможем, например, заключить, что Исаак был сыном Сары. Не было указано, что ребенок является сыном, как отца, так и матери. Можно добавить новое правило, описывающее данное понятие, а именно:

сын(X,Y) ← мать (Y,X),мужчина (X).

Аналогично определение отношения *внук* требует четырех правил, включающих оба случая – отец и мать:

внук (X, Y) ← отец (Y, Z),отец (Z, X).

внук(X, Y) ← отец(Y, Z),мать(Z, X).

внук(X, Y) ← мать(Y,Z),отец(Z, X).

внук(X, Y) ← мать(Y, Z), мать(Z, X).

Имеется лучший, более компактный способ задания этих правил. Нам нужно задать отношение *родитель*, подходящее для отца и для матери. Частично искусство логического программирования и состоит в определении промежуточных предикатов, дающих полную и изящную аксиоматизацию некоторого отношения. Правило, описывающее отношение *родитель*, строится непосредственно, исходя из того, что родитель является или отцом, или матерью. Логические программы могут включать альтернативные определения или, более формально, – дизъюнкцию, применяя альтернативные правила, как в случае отношения *родитель*:

родитель(X, Y) ← отец(X, Y).

родитель(X, Y) ← мать(X, Y).

Правила для отношений *сын* и *внук* теперь запишутся так:

сын(X, Y) ← родитель (Y, X), мужчина (X).

внук(X, Y) ← родитель(Y, Z),родитель(Z, X)

Совокупность правил с одним и тем же предикатом в заголовке, такая, как пара правил для родителей, называется **процедурой.** Позже мы увидим, что при операционной интерпретации этих правил в Прологе подобные наборы правил действительно аналогичны процедурам или подпрограммам в традиционном программировании.

Простой абстрактный интерпретатор

Операционная процедура поиска ответа на вопрос была неформально описана в предыдущих разделах. Углубляясь в детали, рассмотрим абстрактный интерпретатор логических программ. Поскольку обобщенное правило modus ponens ограничено применением к основным целям, интерпретатор отвечает лишь на основные вопросы.

Абстрактный интерпретатор выполняет вычисления с ответами да/нет. Он получает программу P и основной вопрос *Q* и дает ответ *да*, если *Q* выводимо из P, и ответ *нет* в противном случае. Кроме того, если цель невыводима, интерпретатор может вообще не завершить работу. В этом случае не выдается никакого ответа. Рис. 1.1 демонстрирует работу интерпретатора.

Вход: Основной вопрос *Q* и программа P
Результат: *да*, если найден вывод *Q* из программы P,
Алгоритм: *нет* в противном случае.
Положить резольвенту равной *Q*,
пока резольвента $A_1,\ldots A_n$ не пуста начало
выбрать цель A_n, $1 \le i \le n$ и такой основной пример предложения $A \leftarrow B_1, B_2, \ldots B_k, k \ge 0$ в P что $A = A_i$;
(если такого предложения нет, то выйти из цикла);
положить резольвенту равной $A_1, \ldots A_{i-1}, B_1, \ldots B_k, A_{i+1}, \ldots, A_n$
конец
если резольвента пуста, то результат – *да*, иначе результат – *нет*.

Рис. 1.1. Абстрактный интерпретатор логических программ.

На любой стадии вычислений существует текущая цель, называемая **резольвентой. Протоколом** интерпретатора называется последовательность резольвент, которые строятся в процессе вычисления, вместе с указанием сделанного выбора. Рассмотрим вопрос *сын(лот,аран)?* при использовании программы 1.2, являющейся подмножеством фактов программы 1.1, которое дополнено правилами, определяющими отношения *сын* и *дочь*. На рис. 1.2 приводится протокол поиска ответа.

Протокол в неявном виде содержит вывод основного вопроса в программе. Удобнее изображать вывод в виде дерева вывода. Введем необходимые понятия.

Вход: сын (лот, аран)? и программа 1.2
Резольвента не пуста
Выбор: сын (лот, аран) (единственный выбор)
Выбор: сын (лот, аран) ← отец (аран, лот), мужчина (лот).
Новая резольвента: отец (аран, лот), мужчина (лот)?
Резольвента не пуста
Выбор: отец (аран, лот)
Выбор: отец (аран, лот).
Резольвента не пуста
Выбор: мужчина (лот)
Выбор: мужчина (лот).
Новая резольвента пуста
Результат: *да*

Рис.1.2. Протокол интерпретатора

отец(аврам, исаак).	мужчина(исаак).
отец(аран, лот)	мужчина(лот)
отец(аран, милка).	женщина(милка).
отец(аран, иска).	женщина(иска).

сын(X,Y) ← отец(Y,X), мужчина(X).
дочь(X,Y) ← отец(Y,X), женщина(X).

Программа 1.2. Отношения в библейской семье.

• **Определение**: **Основная редукция** цели G с помощью программы P состоит в замене цели G телом того примера правила из P, чей заголовок совпадает с данной целью. •

Редукция является главным вычислительным шагом в логическом программировании. Она соответствует применению обобщенного правила modus ponens, а также одной итерации цикла «пока» в абстрактном интерпретаторе. Замененная при редукции цель называется **снятой,** появившиеся цели называются **производными.**

Применим данные понятия к анализу протокола на рис. 1.2. В протоколе имеются три редукции. Первая редукция снимает цель *сын (лот,аран)* и строит две производные цели – *отец (аран.лот)* и *мужчина (лот)*. Следующая редукция применяется к цели *отец(аран.лот)* и не дает производных целей. Третья редукция также не дает производных целей при снятии **цели** *мужчина (лот)*.

Дерево вывода состоит из вершин и ребер и изображает цели, снимаемые в процессе вычисления. Корень дерева вывода в случае простого вопроса совпадает с этим вопросом. Вершины дерева соответствуют целям, снимаемым в процессе вычисления. Направленное ребро, ведущее из одной вершины в другую, соответствует переходу от снимаемой цели к производной. Деревом вывода для конъюнктивного вопроса является просто совокупность деревьев вывода для отдельных целей в конъюнкции. Рис. 1.3 изображает дерево вывода для протокола, приведенного на рис. 1.2.

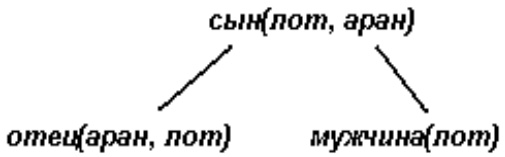

Рис. 1.3. Простое дерево вывода.

В интерпретаторе имеются два неопределенных выбора: выбор цели из резольвенты для снятия и выбор предложения (и подходящего основного примера) для редукции. Эти два выбора имеют различную природу.

Выбор цели для снятия произволен. В любой имеющейся резольвенте все цели должны быть сняты. Можно показать, что порядок снятия при построении доказательства не имеет значения. Иными словами, если существует доказательство цели, то оно является доказательством независимо от того, в каком порядке применяются редукции. В терминах дерева вывода это означает, что порядок ветвей несущественен.

Наоборот, выбор предложения и подходящего основного примера является решающим. В общем случае существует несколько вариантов предложений и бесконечно много основных примеров. Выбор производится недетерминировано. Понятие недетерминированного выбора применяется весьма плодотворно в определении многих моделей вычислений, например, в конечных автоматах и машинах Тьюринга. Недетерминированный выбор состоит в неопределенном выборе из некоторого числа альтернатив, который предположительно производится методом «предвидения»; если только некоторые из альтернатив приводят к успешному вычислению (в нашем случае – к нахождению доказательства), то одна из них и будет выбрана. Формально понятие определяется следующим образом: вычисление, содержащее недетерминированный выбор по определению заканчивается успешно, если существует последовательность недетерминированных выборов, приводящая к успеху. Конечно, ни одна из реальных машин не может непосредственно реализовать это определение.

Интерпретатор, представленный на рис. 1.1, может быть расширен так, чтобы он давал ответы и на неосновные экзистенциальные вопросы. Для

этого следует ввести дополнительный шаг: угадывание основного примера вопроса. Этот шаг подобен тому, на котором интерпретатор угадывает основные примеры правил. Угадать правильный основной пример в общем случае непросто, так как это означало бы знание результата вычисления до его выполнения.

Чтобы избавиться от ограничений, налагаемых основными примерами, и бремени угадывания, потребуются новые понятия.

Деревья вывода позволяют использовать важную меру сложности – число вершин в дереве. Она показывает, сколько шагов редукции выполнялось при вычислении.

Значение логической программы

Откуда известно, что логическая программа вычисляет то, что мы хотим? Корректна она или некорректна? Для ответа на подобные вопросы мы должны определить, что же является значением логической программы. Дав такое определение, можно будет проверять, совпадает ли значение программы с предположительным значением.

• **Определение: Значением** логической программы P, обозначаемым M(P), называется множество выводимых из P основных единичных целей. •

Из этого определения следует, что значением логической программы, построенной просто из основных фактов, такой, как программа 1.1, является сама программа. Другими словами, для простых программ программа «значит ровно то, что написано». Рассмотрим программу 1.1, расширенную двумя правилами определения отношения *родитель*. Что будет ее значением? Оно содержит кроме фактов об отцах и матерях, явно указанных в программе, также все факты вида *родитель* (X, Y) Для каждой пары X и Y, такой, что факт *отец*(X,Y) или *мать*(X,Y) присутствуют в программе. Этот пример показывает, что значение программы в явном виде содержит все то, что программа утверждает неявно.

Если предположить, что подразумеваемое значение программы также задано в виде множества основных единичных целей, то можно спросить, как соотносятся действительное и подразумеваемое значения программы. Мы можем проверить, все ли утверждения программы корректны или вычисляет ли программа все, что нам требуется.

Неформально мы называем программу корректной относительно некоторого заданного значения M, если значение M (P) программы P является подмножеством M. Иными словами, корректная программа не вычисляет того, что не требуется. Программа полна относительно M, если M есть подмножество M(P), т.е. полная программа вычисляет все, что задано. Следовательно, программа P корректна и полна относительно заданного значения M, если M = M(P).

В тех случаях, когда из имен предикатов или констант интуитивно

ясно, что является их значением, будем считать в данной книге, что подразумеваемое значение определяется в программе смыслом имен.

Например, если в программе для отношения сын содержится только первая аксиома, которая ссылается на отношение отец, то программа неполна относительно интуитивно понимаемого значения отношения сын, так как цель сын(исаак,сара) невыводима. Если к программе добавить правило

сын(X, Y) ← мать(X, Y), мужчина (Y).

то получим программу, некорректную относительно подразумеваемого значения ввиду выводимости утверждения сын (сара, исаак).

Будем называть основную цель истинной относительно подразумеваемого значения, если цель входит в данное значение, и ложной в противном случае. В тех случаях, когда подразумеваемое значение определяется именами предикатов и констант, содержащихся в программе, цели, входящие в такие значения, будем называть просто истинными.

Резюме

Основная структура в логических программах – это терм. Терм есть константа, переменная или составной терм. Константы обозначают конкретные объекты, такие, как целые числа и атомы, в то время как переменная обозначает единственный, но неопределенный объект. Символом атома может служить любая последовательность букв, которая берется в кавычки, если ее можно спутать с другими символами (такими, как переменные или целые числа). Символы переменных отличаются начальной прописной буквой.

Составной терм строится из функтора (называемого главным функтором терма) и последовательности из одного или более термов, называемых аргументами. Функтор задается своим именем, т.е. некоторым атомом, и арностью, или числом аргументов. Константы рассматриваются как 0-арные функторы. Синтаксически составной терм записывается как $f(t_1,t_2,\ldots t_n)$,где f – имя n-арного функтора, а t_i – аргументы.. Для n-арного функтора f используется запись f/n. Функторы с одинаковыми именами, но различными арностями различны. Терм является основным , если в нем не содержится переменных; в противном случае терм неосновной.

Цели – это атомы или составные, в общем случае неосновные термы.

Подстановка – это конечное (возможно, пустое) множество пар вида X = t, где X – переменная, t – терм, причем переменные в левых частях пар различны. Для любой подстановки $\Theta = \{X_1 = t_1, X_2 = t_2, \ldots, X_n = t_n\}$ и терма S терм SΘ обозначает результат одновременной замены каждого вхождения в S переменной X_i на t_i, $1<i<n$. Терм SΘ называется примером терма S.

Логическая программа – конечное число предложений. Предложением, или правилом, является замкнутое квантором общности утверждение вида

$A \leftarrow B_1, B_2, \ldots B_k$, $k \geq 0$,

где A и B_i – цели. Декларативное понимание такого утверждения – «A следует из конъюнкции целей B_i;», процедурная интерпретация – «для ответа на вопрос A ответь на конъюнктивный вопрос $B_1, B_2, \ldots B_k$». A называется заголовком предложения, последовательность B_i – телом предложения. При k = 0 предложение называется фактом или единичным предложением и имеет запись A., декларативное понимание – «A истинно», процедурная интерпретация – «цель A выполнена». При k = 1 предложение называется итерационным.

Вопросом называется конъюнкция вида

$A_1, \ldots, A_n$?$n>0$.

где A_i – цели. Считается, что переменные в вопросе связаны квантором существования.

Вычисление логической программы P строит пример заданного вопроса, логически выводимый из P. Цель G выводима из P, если существует такой пример A цели G, что $A \leftarrow B_1, \ldots B_n$, $n \geq 0$, – основной пример предложения в P и каждое B_i, выводимо из P. Выводимость цели из тождественного факта рассматривается как особый случай.

С помощью логической выводимости индуктивно определяется значение программы P. Множество основных примеров фактов из P принадлежит значению P. Основная цель G входит в значение, если существует такой основной пример $G \leftarrow B_1, \ldots B_n$ правила из P, что $B_1, \ldots B_n$ входят в значение P. Таким образом, значение состоит из тех основных примеров, которые выводятся из программы.

Подразумеваемое значение M программы P также задается в виде множества основных единичных целей. Программа P корректна относительно подразумеваемого значения M, если M(P) образует подмножество M. Программа P полна относительно M, если M – подмножество M(P). Ясно, что программа корректна и полна относительно ее подразумеваемого значения (наиболее желательный случай), когда M = M(P).

Основная цель называется истинной относительно подразумеваемого значения, если она принадлежит этому значению. В противном случае цель называется ложной.