Вопросы теории и практики логического программирования.

Н. Пелин – Государственный университет Тирасполь(UST),
С. Пелин - Университет прикладных знаний Молдовы (UŞAM),

Резюме

В обзорной работе проанализированы этапы развития теории и практика использования логического программирования. Особое внимание уделено дизъюнктам хорна, которые легли в основу компьютерных программ написанных в логике. Проводится пошаговый анализ преобразований от логической модели реального мира до конъюктивно-нормальной формы формулы логики предикатов, состоящая из множества дизъюнктов хорна, которая практически и есть текст реальной программы написанной на языке логического программирования Пролог. Отмечается, что полученная форма интерпретируема с помощью механизма логического вывода автоматически.

Ключевые слова: логика, логическое программирование, Пролог, интерпретация.

Введение

Логическое программирование базируется на основах формальной логики. Начало исследованиям в области формальной логики было положено Аристотелем в IV в. до н. э. Он создал труд по логике, разработал теорию мышления и его формы, учение о силлогизме, сформулировал логические законы. Логика Аристотеля, тесно связана с естественным языком. В XIX веке появился первый формальный язык логики - исчисление высказываний, а несколько позднее более развитый язык – исчисление предикатов.

В рамках этих формализмов идеи Робинсона, Ковальского ряда других, преимущественно британских ученых и специалистов, в 60-е годы прошлого столетия заложили основы теории логического программирования. А вот первая версия языка логического программирования Пролог разработана французом Аленом Колмероо, лингвистом по специальности. В 1971 году, в Марселе он продемонстрировал работу первого интерпретатора языка программирования Пролог. Само название языка Пролог образовано соединением первых трех букв каждого из слов словосочетания "**ПРО**граммирование **ЛОГ**ическое". Отождествлять язык Пролог с самим Логическим программированием не следует. Пролог -

лишь малая часть теории логического программирования, нашедшая в таком виде свое практическое применение.

Итак, что представляет собой логическое программирование как теория. В основе языка логического программирования лежат элементы логики. Программа, написанная на таком языке представляет собой множество дизъюнктов (фраз) Хорна. Процесс вывода умозаключения на основе некоторых предпосылок называют логическим выводом. В нем класс предикатов одного суждения согласуется с классом субъектов другого суждения. В ходе вывода на основе субъекта одного суждения и предиката другого суждения формируется новое суждение. Таким образом, логический вывод можно определить и как некий процесс рассуждения.

В данной статье мы ограничимся рассмотрением лишь элементами теории языка логического программирования. Логическое программирование - это универсальный, абстрактный язык, предназначенный для представления знаний и решения задач. Это один из подходов к информатике, при котором в качестве языка высокого уровня используется логика предикатов первого порядка (ветвь формальной логики) в форме фраз Хорна [1, с.17].

Пролог более приспособлен для решения задач искусственного интеллекта. Однако считать его таковым, в смысле единственности, было бы неверно. ЛИСП имеет не меньшую известность в сфере искусственного интеллекта. Пролог достаточно хорошо приспособлен и к решению других задач, и все это делает его весьма привлекательным и перспективным. Все же, его изящность при построении элементов систем искусственного интеллекта не удержала авторов монографий [2,3] от желания обратить внимание читателей даже в названиях своих книг на то, что Пролог - язык искусственного интеллекта.

Язык программирования Пролог привлекателен своей простотой. Пролог – программу легко читать. Универсальные формализмы фраз Хорна, меньше подвергают текст Пролог – программ влиянию машинно-зависимых особенностей, нежели программы, написанные на других языках [1, с.18].

В настоящее время Пролог - широко известный язык. Над развитием Пролога работают известные научные центры и крупнейшие компании мира, среди которых Ассоциация Логического Программирования (Лондон) [14], Центр Развития Пролога (Копенгаген) [15] и другие [16].

При использовании языка логического программирования основное внимание уделяется описанию структуры прикладной задачи, а не выработке предписаний компьютеру о том, что ему следует делать. С помощью логических программ можно описать и реализовать также реляционные базы данных, задачи программной инженерии и представления знаний. На

Прологе можно реализовать и многие другие задачи, – вопрос стоит только в степени эффективности и целесообразности.

Язык программирования навязывает пользователям определенный взгляд на окружающий мир. Это проявляется в том, что программист, длительное время пользующийся некоторым языком и усвоивший соответствующий этому языку взгляд на мир, будет стремиться находить новые сферы применения для тех типов вычислений, к которым данный язык приспособлен, и будет избегать задач, для решения которых этот язык не годится. Целесообразно оценить взгляды на мир, навязываемые различными языками программирования, в соответствии с тем, насколько хорошо они согласуются с поставленными целями. Данные рассуждения относятся к мета программной инженерии: задана цель программирования - следует определить, какие языки и программные средства наилучшим образом удовлетворяют ей [1, с.13].

О логическом программировании как теории

Как уже отмечалось выше, логическое программирование зародилось в недрах формальной логики и без некоторых важных элементов логики высказываний и логики предикатов нам не обойтись.

Логика высказываний – простая теория, но она широко используется в разных областях. Она лежит в основе практически любой логико-математической теории**,** ее простота не препятствует быть высоко содержательной и широко применимой для задач как теоретического, так и прикладного характера. Информатикам необходимо особо старательно отнестись к изучению логики, чтобы впоследствии было легче разобраться во многом, включая логическое программирование, принцип работы интерпретаторов языка логического программирования и конструкций языков типа Пролог.

Как известно, логика высказываний это язык, который изучает элементарные высказывания, т.е. высказывания (предложения), которые нельзя разбить на компоненты. Они могут быть обозначены, например символами *p,g,r* и называться далее как элементарные формулы. Интерпретируются как ***истина*** или ***ложь [И, Л].***

Логика предикатов позволяет войти во внутреннюю структуру высказывания, т.е. имея представление вида *p(a,b),p(b,c),g(a,c)* и называться уже атомарными формулами. Но интерпретируются, также как в логике высказываний, только воедино как ***[И, Л].***

Сложные высказывания формируются с помощью связок, аналогичных союзам «и», обозначаемого в формуле с помощью символа - $\wedge$**,** «или» - $\vee$, «если …, то» - $\supset$,«тогда и только тогда, когда» - $\equiv$ ***и*** отрицания «не» обозначаемого - $\neg$. Приоритетность связки в сложном

высказывании устанавливается, также как и в математике, с помощью скобок. Иногда, для представления совокупности высказываний, образующих некий единый текст, используют знак «,». В логике предикатов оперируют еще кванторами общности и кванторами существования, понятиями о константе и переменной. Правила построения формул сводятся к установлению базиса, что элементарные высказывания p и q в логике высказываний (или $p(a,b)$ и $q(x,y)$ в логике предикатов), есть формулы (элементарные или атомарные формулы), а сложные высказывания формируются последовательным применением индуктивного шага $\neg p,\ p \wedge q,\ p \vee q,\ p \supset q$ и $p \equiv q$ необходимое число раз.

Истинность формул устанавливается с помощью таблиц истинности, формул более сложных - алгоритмически. Интерпретация, при которой значение истинности формулы есть ***И,*** называют ***моделью*** этой формулы. Формулу, допускающую одну или более чем одну модель, называют выполнимой, ни одну модель – невыполнимой. Если она допускает модель при любом значении, входящих в неё атомарных формул, то ее называют общезначимой.

Для интерпретации формул с помощью алгоритмических методов могут применяться эквивалентные преобразования с целью их приведения к неким нормальным формам.

В логическом программировании оперируют, как правило, конъюнктивно-нормальными или дизъюнктивно-нормальными формами. В данной статье мы ограничимся рассмотрением лишь конъюнктивно-нормальных форм. Для этого уместно ввести понятие о дизъюнкте.

Дизъюнкт. Дизъюнктом (clauses) называется дизъюнкция конечного числа литер, т.е. элементарных высказываний или его отрицаний. Пусть $p_1, p_2, \ldots, p_n$ и $q_1, q_2, \ldots, q_m$ множество литер, тогда формальное представление дизъюнкта может быть следующим: $(p_1 \vee p_2 \vee \ldots \vee p_n \vee \neg q_1 \vee \neg q_2, \ldots, \neg q_m)$.

Понятие о дизъюнкте имеет важное значение для практики. Описание задач и алгоритмов в терминах дизъюнктов, составляют основу логического программирования и языка Пролог в частности. Собственно Пролог оперирует только подклассом дизъюнктов, называемых хорновскими дизъюнктами, речь о которых пойдет ниже.

Пустой дизъюнкт, т.е. дизъюнкт, не имеющий ни одной литеры - единственный из дизъюнктов, который невыполним. Его обозначают через *Л*. Определение понятия пустой дизъюнкт очень важно в логическом программировании. С его введением появляется возможность детерминации цели в автоматическом решателе задач.

Конъюнктивно-нормальные формы (КНФ). Разобравшись с понятием о дизъюнкте, дадим сейчас определение КНФ. КНФ – это конъюнкция конечного числа дизъюнктов. Формально его можно представить следующим образом:

$$(p_1 \vee p_2 \vee \ldots \vee p_n) \wedge (q_1 \vee \ldots \vee q_m) \wedge \ldots \wedge (r_1 \vee r_2 \vee \ldots \vee r_l).$$

В теоретико-множественном представлении КНФ, обозначенная через ***S,*** будет:

$$S=\{(p_1 \vee p_2 \vee \ldots \vee p_n), (q_1 \vee \ldots \vee q_m), \ldots, (r_1 \vee r_2 \vee \ldots \vee r_l)\}.$$

Любая формула может быть преобразована в логически эквивалентную ей КНФ, используя соответствующие эквивалентные формулы [2,4,5,14 и др.] .

Определение хорновского дизъюнкта. В настоящей работе, будем рассматривать только случай для хорновских дизъюнктов, в которых допускается наличие не более одной позитивной литеры. В соответствии с этим определением, дизъюнкт $\neg p \vee \neg q \vee \neg r \vee s$, является хорновским. Он эквивалентен импликации $(p \wedge q \wedge r) \supset s$. Если этот же дизъюнкт переписать в виде *s:-p,q,r.* , то при такой записи хорновский дизъюнкт можно уподобить правилу переписывания, а множество хорновских дизъюнктов – некоторой контекстно-свободной грамматике [8, с. 49]. В ней высказывания и символ *Л* можно рассматривать в качестве нетерминальных символов этой грамматики. Такая форма представления близка и к форме, принятой в языке программирования Пролог если литеры рассматривать как вызываемые процедуры.

Унитарный хорновский дизъюнкт – это дизъюнкт, содержащий одну – единственную положительную литеру. Формат записи: ***s:-.*** Этот дизъюнкт служит для представления некоторого факта.

Хорновский дизъюнкт называется точным, если он содержит одну позитивную литеру и одну или более негативные литеры. Точный дизъюнкт выражает правило, в котором негативные литеры - гипотезы, представленные соответствующими высказываниями, а позитивная литера – заключение. Формат записи: ***s:-p,q,r.*** Это выражение соответствует записи для точного дизъюнкта.

Негативный хорновский дизъюнкт представим в следующем виде: ***Л:-p,q,r.*** Он соответствует понятию «цель, вопрос» или точнее «отрицание цели, вопроса». В приведенном примере цель сложная. Пример простой цели: ***Л:-p.***

Формат записи цели и факта в Турбо-Прологе (наиболее известная версия языка логического программирования для русскоязычной аудитории, в связи с имеющимися многочисленными русскоязычными публикациями) несколько отличается от записи, приведенной выше записи, хотя несет ту же смысловую нагрузку. Запись простой цели в Турбо-Прологе: ***p.*** Запись сложной цели в этом языке: ***p,q,r.*** Запись факта: ***p.*** Однако, чтобы не перепутать, например, факты ***p*** с вопросами ***p***, факты и правила записываются в одном разделе программы, обозначаемом ***clauses***, а цели записываются в другом разделе программы, обозначаемом ***goal***.

Логика предикатов. До настоящего момента, для описания ряда элементов применяемых в логическом программировании, мы обходились формализмом логики высказываний. Для выполнения последующих шагов в логическом программировании понадобится формализм логики предикатов.

Как уже отмечалось выше, в логике предикатов элементарное высказывание так же, как и в логике высказываний, рассматривается как нечто цельное, интерпретирующееся со значением или ***ИСТИНА*** или ***ЛОЖЬ***, но при этом представляется возможным формализовать и внутреннюю структуру. Продемонстрируем это на простом примере. Пусть имеем элементарное высказывание:

«***Язык Пролог эффективнее языка С».***

В логике высказываний мы можем установить лишь истинность этого высказывания. В логике предикатов представляется также возможным представить это высказывание в виде атомарной формулы (атома) с внутренней структурой. Для этого отношение ***«эффективность»*** в приведенном высказывании обозначим его через ***P***, а связанные с этим отношением объекты "***язык Пролог***" и "***язык С"*** обозначим соответственно через q_1 и q_2. Тогда формальным представлением атомарной формулы для

Язык Пролог эффективнее языка С.

объект — отношение — объект

q_1 P q_2

будет выражение вида: $P(q_1, q_2)$. Для того чтобы оперировать понятиями логики предикатов очень важно иметь более четкое представление об атомарной формуле и не запутаться в множественности определений каждого из ее элементов. Эти определения достаточно хорошо даются в разных литературных источниках [1, 4, 7, 13,14]. И все же, как показывает практика, они трудно осваиваются студентами в процессе обучения.

Нам представляется полезным древовидное представление атомарной формулы (рис.1), предложенное в [14] для более легкого осмысления всех ее элементов во взаимосвязях.

В качестве символического обозначения в атомарной формуле отношения вообще и для ***P*** в формуле $P(q_1, q_2)$ используют термин под названием ***«предикат»*** или ***«предикатная константа»*** , а объектов q_1, q_2 –***«термы»***, которые выступают в роли аргументов этого предиката. Количество аргументов в предикате определяет его арность. Например, предикат $P(q_1, q_2)$, имеет арность два, предикат ***P –*** арность нуль, ***P(a,b,c)*** - три**.**

Итак, атомарная формула или атом – это предикатная форма (например, ***p(a,b)*** или ***p(a,b,x)),*** представленная в префиксной форме, или некоторое отношение (например, ***s=t***), представленное в инфексной форме. Инфексная форма может быть заменена на префиксную, если такое соглашение в конкретном случае допустимо. Тогда, для нашего примера, эта атомарная формула предстанет как

=(s, t).

Объектами логики предикатов являются переменные и индивидные константы. В языке предикатов содержится язык высказываний. Высказывание в логике предикатов рассматривается как предикатная константа без аргументов. Ее еще называют нульместной предикатной константой или константа арности нуль.

В естественном языке часто встречаются выражения типа «для всех мужчин», «для некоторых студентов» и множество других, аналогичных им выражений. Выражения общности типа «для всех…» в логике предикатов индентифицируют через квантор общности, «для некоторых…» – квантор существования.

Запись ***∀xP(x)*** означает «для всякого значения ***x P(x)*** – истинное высказывание». Пример: ***∀x(x²+1>0)*** – для всякого значения ***x*** неравенство ***(x²+1>0)*** - истинно.

Запись ***∃xP(x)*** означает «для некоторых значений ***x P(x)*** – истинное высказывание». Например: ***∃x(3+x=3)*** – при ***x=0*** равенство ***(3+x=3)*** – истинно.

Следует помнить, что, если в состав формулы входит переменная, то перед тем, как этой формуле можно будет присвоить истинностное значение, переменную необходимо квантифицировать.

Основные символы языка логики предикатов это переменные, индивидные константы, предикатные константы, связки (отрицание, конъюнкция, дизъюнкция, импликация и эквивалентность), квантор общности и квантор существования.

Переменные и их роль. В математике переменные позволяют указать в структуре математического объекта те места, которые при использовании этого объекта будут заняты другими объектами. Переменные подразделяют на свободные и связные. Например, в формуле

$$S = \sum_{i=1}^{n} 6x,$$

x - свободная переменная, а ***i –*** связная.

В логике из интуитивного понимания квантификации вытекает, что имеется связь между вхождениями переменной, содержащейся в квантификации. Рассмотрим формулу

∀x p (x).

Здесь переменная ***x*** «связана».

Область действия некоторой квантификации есть формула, к которой применяется эта квантификация, а переменная ***x*** в квантификациях ***∀x*** или ***∃x*** называется квантифицированным. Каждое вхождение переменной ***x*** в область действия этой квантификации является связным. Рассмотрим другую формулу: ***∀ x p (x) ^q(x).*** Здесь первое вхождение переменной ***x*** – «связано», второе – «свободно».

Об интерпретации формул в логике предикатов. Так же, как и в логике высказываний, формулы логики предикатов могут быть интерпретированы со значениями ***[И, Л].*** Однако в логике предикатов формулы состоят не только из подформул или литер, но и из термов, т.е. переменных и функциональных форм или, как их еще называют, функциональных констант, соответственно соединенных с подходящим числом термов. Терм интуитивно означает объект. Следовательно, интерпретация должна специфицировать множество объектов, называемых областью интерпретации.

Рассмотрим один из способов построения семантики логики предикатов, который заключается в следующем:

1. Выделяется некоторое непустое множество, называемое областью интерпретации.
2. Определяется функция, сопоставляющая выражениям языка объекты, множества объектов или отношения между объектами области интерпретации.
3. На этой основе определяются важные понятия выполнимости, истинности и общезначимости формул рассматриваемого языка.

Более подробную информацию о механизме интерпретации языка логики предикатов можно найти в [4, с.62-64] и [6, с. 23-24].

В логике предикатов, в дополнении к имеющимся в логике высказываний эквивалентным преобразованиям, есть свои дополнительные схемы формул, описывающих взаимоотношения между квантификациями и связками [4, 9, 11,13].

Предваренная форма и ее связь с формулами логики высказываний. По аналогии с дизъюнктивной и конъюнктивной нормальными формами формул, приведенными в логике высказываний, приводятся к нормальной форме и формулы логики предикатов. Однако проблемы, возникающие при оперировании с квантификациями, и области их действия бывают сложными.

Проблема упрощается для случая предваренных форм. Как известно, предваренная форма есть формула, состоящая из матрицы, перед которой стоит конечная последовательность

квантификаций. Формула имеет вид $Q_1x_1, Q_2x_2, \ldots, Q_nx_nM,$ где символ Q означает либо $\forall$, либо $\exists$, для $i= 1,2,\ldots, n$ и M – формула (матрица), не содержащая квантификаций.

Эти квантификации относятся к различным переменным и их порядок – существенен. В случае повторения квантификаций для одной и той же переменной можно принять, что существенна только самая первая квантификация: это согласуется с общим правилом интерпретации квантифицированных формул [4, с. 69-70].

Интерес к предваренным формам связан с следующей теоремой: ”Для любой логической формулы существует логически эквивалентная ей предваренная форма”. Мы не будем здесь рассматривать алгоритм получения предваренной формы. Этапы получения предваренной формы можно увидеть в [4, с.70-71].

КНФ, выведенные в логике высказываний, распространяются и на логику предикатов. КНФ – это предваренная форма, матрица которой есть конъюнкция дизъюнктов.

О проблемах, возникающих при работе с предваренными формами. Любая формула логики предикатов допускает эквивалентную предваренную форму. С другой стороны, предваренная форма допускает эквивалентную, нормальную форму, хотя иногда и более громоздкую, чем исходная. Предваренные формы интересны тем, что они сохраняют выразительную силу исчисления предикатов и требуют лишь “дисциплинированного” использования механизма квантификации. Но как раз это и является источником сложности логики предикатов.

Более строгие пределы использования механизма квантификации можно установить, хотя при этом выразительная мощь механизма квантификации несколько уменьшится.

Для этого каждой формуле A сопоставляют некоторую формулу S_A - по строению простую, удовлетворяющую лишь условию, что и S_A , и A либо одновременно выполнимы или невыполнимы. Форма S_A известна под названием **сколемовской** формы. Связь между A и S_A строго слабее, чем логическая эквивалентность. Все же доказательство невыполнимости становится эффективнее, если ограничиться только формулами, представленными в сколемовской форме. При оперировании сколемовскими формами рассматривают только предваренные формы формул логики предикатов, эти формулы должны быть замкнутыми.

Как известно, замкнутыми являются те формулы, которые не имеют свободных переменных.

Примеры:

1. $\forall x\,[Q(x) \wedge \forall y\,\exists z\,P(a,y,z)] \wedge R(x)$ - не замкнутая формула.
2. $\forall x\,[Q(x) \wedge \forall y\,\exists z\,P(a,y,z) \wedge R(x)]$ - замкнутая формула.

Предварительные операции по приведению произвольной формулы логики предикатов к сколемовской приводят к замкнутой предваренной форме. Соответствующая ей сколемовская форма получится в результате проведения, так называемого, сколемовского преобразования, предназначенного для исключения $\exists$- квантификаций.

Клаузальная форма. Клаузальной формой называется такая сколемовская форма S_A формулы логики предикатов ***A*,** матрица которой является КНФ, т.е.

S_A =<*последовательность квантификаций*>*и [матрица в КНФ]*.

Любая сколемовская форма допускает эквивалентную клаузальную форму.

Не существует алгоритма, позволяющего распознавать общезначимость, нейтральность или невыполнимость произвольной формулы логики предикатов из-за возможности неисчислимого множества ее интерпретаций. В этой ситуации интересны частные результаты, полученные Эрбраном. Они приводят к упрощенной проверке выполнимости формул.

Определение понятия эрбранова область. Основная идея, подводящая к определению эрбрановой области, состоит в следующем [4, с.76]. Клаузальная форма невыполнима тогда и только тогда, когда она принимает значение ***Л*** при всех интерпретациях. Рассмотреть все возможные интерпретации невозможно, но было бы хорошо, если была бы возможность определить некую специальную область, которая была бы тесно связана с рассматриваемой клаузальной формой и чтобы форма была невыполнима тогда и только тогда, когда она принимает значение ***Л*** при всех интерпретациях этой области. Такая область существует и называется она эрбрановой областью.

Фразовая форма логики предикатов. Фразовая форма логики предикатов – это способ записи формул, при котором употребляются только связки $\neg, \wedge, \vee$. Ранее определенная КНФ и есть представление в фразовой форме. Литера (литерал) – это позитивная или негативная атомарная формула. Каждая фраза – это множество литер, соединенных связкой $\vee$. Это не что иное, как дизъюнкт определенный здесь ранее. Негативные литеры размещаются в конце каждой фразы, а позитивные в начале [1, с.53-54]. Схематический вид фразы:

$$p_1 \vee p_2 \vee \ldots \vee p_n \vee \neg q_1 \vee \neg q_2 \vee \ldots \vee \neg q_m .$$

Как известно, фразу можно рассматривать как обобщение понятия импликации. Например, если p и q атомарные формулы, то формула $p \vee \neg q$ в то же время может быть представлена и как $q \supset p$.

Использование терминов «фраза» и “фразовая форма” также применимо и в дальнейшем изложении материала будет использоваться наряду с терминами “дизъюнкт” и “КНФ”.

Квантификация переменных во фразовой форме. Явная квантификация переменных во фразовой форме не употребляется. Однако неявно все переменные квантифицированы. Так, во фразе ***P(x,y,z)∨Q(x,y,z)*** подразумевается наличие кванторов:

$$\forall x \forall y \forall z\ [P(x,y,z) \vee Q(x,y,z)].$$

Практика построения логических программ.

Логическая программа и ее интерпретация. С точки зрения теории логическая программа есть не что иное, как конъюнкция множества хорновских дизъюнктов. Это модель реального мира, представленная в виде некой конъюктивно-нормальной формы в которой в качестве дизъюнктов используются только:

- унарный хорновский дизъюнкт (для представления некоторого факта представленного в виде простого предложения из естественного языка),
- точный (для представления некоторого правила - условного предложения из естественного языка) и
- негативный хорновский дизъюнкт (для представления некоторого вопросительного предложения из естественного языка - цели).

Это фактически множество предложений (деклараций), которые воедино как-бы образуют некий текст, аналогичный написанному на естественном языке. Читая текст на естественном языке, смысл в нем улавливаем, просматривая его по определенной стратегии, например: сверху вниз, слева направо. И, если мы имеем целью найти ответ на определенный вопрос (цель), мы просматриваем текст по этой стратегии до тех пор, пока не находим, на каком то этапе просмотра, ответ на наш вопрос или же просматриваем его до конца текста и убеждаемся что ответа на данный вопрос в этом тексте мы не имеем. Аналогично вышеизложенному, интерпретатором логических программ «просматривается» текст конкретной программы сверху – вниз, слева – направо. Вопрос сопоставляется с конкретным фактом (утверждением) или правилом (условным предложением). При сопоставлении вопроса с соответствующим фактом, получаем ответ на вопрос без всяких условий. При сопоставлении вопроса с головой соответствующего правила, появляется новая проблема - необходимость сопоставления содержания тела правила (т.е. каждого из множества под вопросов, входящих в тело правила) с соответствующими иными фактами и\или правилами. И так далее до получения полного ответа на поставленный вопрос. Такова, в общих чертах, идея интерпретации логической программы интерпретатором языка логического

программирования. Подробнее о логическом интерпретаторе, его абстрактной модели, можно будет ознакомиться в следующей авторской статье, а также в [10, 11,12] .

Структура стандартной логической программы. Стандартная логическая программа есть целевое утверждение (вопрос) и произвольное количество процедур (факты и правила). Число и длина утверждений не ограничено. В реальных программах написанных, например, на языке Турбо-Пролог, предикатная константа и термы атомарной формулы обозначаются для удобства чтения на естественном языке не только отдельными буквами, но и зачастую словами или группой слов, отделенных друг от друга знаком подчеркивания. Константы в Турбо-Прологе начинаются с прописной буквы или арабской цифры, переменные с заглавной или знака подчеркивания. Пример программы, написанной на языке Турбо-Пролог:

```
CLAUSES
        parinte(ion,nicolae).
        parinte(nicolae,sergiu).
        parinte(sergiu, diana).
        bunei(X,Z):-
                parinte(X,sergiu),
                parinte(sergiu,Z).
GOAL
        bunei(X, diana).
```

Текст данной программы содержит все три типа хорновских дизъюнкта: три факта (унарных дизъюнкта), одно правило (точный хорновский дизъюнкт) и один вопрос или одна цель (негативный дизъюнкт). При согласовании цели ***bunei(X, diana) X*** связывается со значением ***nicolae.***

Порядок (последовательность), в котором процедуры (факты и правила) появляются в программе, логического значения не имеет, т.к. каждая процедура устанавливает некоторый факт о решаемой задаче. Все же принято собирать процедуры для одной и той же предикатной константы в одну группу.

Заключение

В статье дан краткий обзор элементов логического программирования как теории с демонстрацией их применения в одной из реализаций языка Пролог.

С методологической точки зрения и необходимости в ограниченный объем вложить минимально необходимые сведения из теории акценты ставились лишь на главных аспектах, оставив вне рассмотрения многие из альтернативных направлений или направлений второстепенного характера. В то же время в таком представлении материал статьи может представить определенный интерес для тех, кто хотел бы ознакомиться, в общих чертах, с

теорией и практикой логического программирования. Обзор, выполненный в таком представлении, легко подвергаем структуризации в «матрице элементов знаний».

Литература

1. Малпас Дж. Реляционный язык Пролог и его применение: Пер. с англ. Под. Ред. В.Н. Соболева – М.: Наука. Гл. ред. Физ.- мат.лит., 1990-464 с.
2. Макаллистер Дж.Искусственный интеллект и Пролог на микроЭВМ/Пер. с англ.- М.: Машиностроение, 1990.- 240 с.
3. Братко И. Программирование на языке Пролог для искусственного интеллекта. Пер. с англ. – М.: Мир, 1990 – 560 с.
4. Тей А., Грибомон П., Луи Ж. и др. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. Пер. с. франц. Пермякова П.П. Под. ред. Гаврилова Г.П. – М., «Мир», 1990 – 432 с.
5. Хоггер К. Введение в логическое программирование: Пер.с англ.М:-Мир,1998-348 с.
6. Логика и компьютер. Моделирование рассуждений и проверка правильности программ / Н.А. Алешина, А.М. Анисов, П.И. Быстров и др.–М.: Наука,1990– 240 с.
7. Ковальски Р. Логика в решении проблем: Пер. с англ. – М.: Наука. Гл. ред. физ. мат. лит., 1990. – (Пробл. искусств. интеллекта) - 280 с.
8. Лорьер Ж.-Л. Системы искусственного интеллекта: Пер.с франц.- М.: Мир, 1991.- 568 с.
9. Кузнецов О.П., Адельсон – Вельский Г.М. Дискретная математика для инженера – 2 –е изд., перераб. и доп. М.: Энергоатомиздат, 1988. - 480 с.
10. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог: М., Мир, 1987.
11. Пелин Н. Элементы логического программирования. Ch.: Nestor, 2000. - 171 стр.
12. Pelin S., Pelin N. Programarea logică în proiectarea sistemelor informaționale.- Ch.: UST, 2011.- 221 p.
13. Pelin S. Prezentarea realităţii cu ajutorul logicii. Raport prezentat doctoranturei pe linga Universitatea Alex. I. Cuza în conformitate cu prevederile a planului de lucru. Iasi. 2007.
14. http://www.cs.kuleuven.ac.be/~dtai/projects/ALP/
15. www.pdc.dk
16. www.prolog.md