Лекция 4.

Раздел 4. Основы логического программирования на языке Prolog

Тема 12-16. Язык PROLOG: история появления и диалекты. Синтаксис и структура программы. Основные виды предложений: факты, правила и целевые утверждения (запросы). Семантический дуализм Пролога. Выполнение пролог-программы. Согласование целей, унификация. Режим возвратов. Примеры использования языка Пролог в задачах прикладного искусственного интеллекта.

План лекции.

1. Язык программирования PROLOG.
2. Синтаксис и структура программы.
3. Пролог в задачах прикладного искусственного интеллекта.

Основная часть.

Основная суть качественного перехода к пятому поколению ЭВМ заключалась в переходе от обработки данных к обработке знаний. Этот проект послужил импульсом к развитию нового витка исследований в области искусственного интеллекта и вызвал взрыв интереса к логическому программированию.

Логический язык программирования ПРОЛОГ появился в начале 70-х годов в качестве экспериментальной разработки лабораторий искусственного интеллекта университетов Марселя и Эдинбурга.

В настоящее время Пролог остается наиболее популярным языком искусственного интеллекта в Японии и Европе.

На сегодня существует довольно много реализаций Пролога. Наиболее известные из них следующие: BinProlog, AMZI-Prolog, Arity Prolog, CProlog, Micro Prolog, МПролог, Prolog-2, PDC Prolog 3.31, Visual Prolog 4-6.0, Quintus Prolog, SICTUS Prolog, Silogic Knowledge Workbench, Strawberry Prolog, SWI Prolog, UNSW Prolog и т. д.

В нашей стране были разработаны такие версии Пролога как Пролог-Д (Сергей Григорьев), Акторный Пролог (Алексей Морозов), а также Флэнг (А. Манцивода, Вячеслав Петухин).

№ этапа	Период развития	Направление развития	Авторы
I	1950 г.	Публикация в журнале «Mind» статьи «Вычислительные машины и интеллект». В этой статье были сформулированы идеи, которые получили название «Тест Тьюринга».	А. Тьюринг
II	1973 г.	Создание программы для доказательства теорем, которая получила название Prolog (от Programmation en Logique). Эта программа использовалась при построении систем обработки текстов на естественном языке. Она послужила прообразом Пролога.	«группа искусственного интеллекта» во главе с Аланом Колмероэ
III	1974 г.	Выпущена работа «Логика предикатов как язык программирования».	Р. Ковальский
	1976 г.	Предложено два подхода к прочтению текстов логических программ: процедурный и декларативный.	Р. Ковальский и Маартен Ван Эмден
	1977 г.	Создан эффективный компилятор языка Пролог для ЭВМ DEC-10, который послужил прототипом для многих последующих реализаций Пролога, так называемая «Эдинбургская версия», фактически ставшая первым и единственным стандартом языка.	Уоррен и Перейра
	1980 г.	Разработана версия Пролога для персональных компьютеров.	Кларк и Маккейб
	1986-1992 гг.	Выходит первая версия Турбо Пролога. PDC Prolog 3.31.	Borland International и Prolog Development Center (PDC)
IV	1991 г.	Проект создания компьютеров пятого поколения, работа которых должна приблизиться к человеческому мышлению (отход от фон Неймановской архитектуры). Были созданы специализированные компьютеры логического программирования PSI и PIM. Этот проект вызвал взрыв интереса к логическому программированию.	Японские программисты
V	1996-2007 гг.	Выпуск системы Visual Prolog 4.0.- Visual Prolog 6.0.	PDC совместно с российскими программистами

Основной принцип использования языка Пролог состоит в том, что нужно подробно, на логически точном языке, описать условие задачи. Решение ее получается в результате определенного рутинного процесса, который исполняется компьютером. В этом заключается принципиальное отличие Пролога от традиционных языков

программирования, которые требуют описания того, как должен быть вычислен результат, или другими словами, требуют описания процедуры решения задачи.

При программировании на Прологе усилия программиста должны быть направлены на описание логической модели фрагмента предметной области решаемой задачи в терминах объектов предметной области, их свойств и отношений между собой, а не деталей программной реализации. Фактически Пролог представляет собой не столько язык для программирования, сколько язык для описания данных и логики их обработки. Программа на Прологе не является таковой в классическом понимании, поскольку не содержит явных управляющих конструкций типа условных операторов, операторов цикла и т. д. Она представляет собой модель фрагмента предметной области, о котором идет речь в задаче. И решение задачи записывается не в терминах компьютера, а в терминах предметной области решаемой задачи, в духе модного сейчас объектно-ориентированного программирования.

Еще одним применением Пролога является интерпретация трансляторов с языков программирования. С помощью выразительных средств синтаксиса этого языка удается достичь высокой степени лаконичности изложения, это делает возможным описать логику работы интерпретатора с помощью нескольких строк текста.

В Прологе очень компактно, по сравнению с *императивными языками*, описываются многие алгоритмы. По статистике, строка исходного текста программы на *языке Пролог* соответствует четырнадцати строкам исходного текста программы на *императивном языке*, решающем ту же задачу. Пролог-программу, как правило, очень легко писать, понимать и отлаживать. Это приводит к тому, что время разработки приложения на *языке Пролог* во многих случаях на порядок быстрее, чем на *императивных языках*. В Прологе легко описывать и обрабатывать сложные структуры данных.

Прологу присущ ряд механизмов, которыми не обладают традиционные языки программирования: сопоставление с образцом, вывод с поиском и возвратом. Еще одно существенное отличие заключается в том, что для хранения данных в Прологе используются списки, а не массивы.

Декларативная семантика языка ПРОЛОГ. Изучать Пролог без привязки к конкретной его версии, мне кажется, не совсем целесообразно. Как уже было сказано выше, версий Пролога очень много, и нужно выбрать какую-нибудь одну из них, чтобы привязать к этой версии разбираемые в приложении примеры. Мы остановимся на наиболее эффективной версии Пролога у нас в стране – *Прологе-Д.*

Текст на Прологе-Д содержит сообщения двух типов: факты и правила. Факт - это синтаксическая конструкция, которая позволяет накапливать информацию, а правило -

конструкция, с помощью которой можно делать заключение или вывод. Таким образом, факты и правила связывают объекты и отношения между ними.

В качестве объектов, которые связывают правила и факты, могут быть элементы любой природы: целые числа, переменные, графические образы.

Совокупность фактов и правил образуют базу знаний. В отличие от процедурных языков, порядок фактов и правил не имеет, существенного значения для правильности результата, за исключением нескольких случаев, о которых будет сказано особо.

Для запуска системы необходимо задать вопрос. Вопрос - это факт, которому предшествует символ "?". База знаний и вопрос образуют программу на языке Пролог-Д.

Теоретические основы языка. Математическая логика является теоретической основой логического программирования. Для решения задачи с помощью Пролога-Д достаточно описать знания об этой задаче, а процесс построения решения при этом сводится к некоторой рутинной процедуре.

Описание знаний возможно осуществить с помощью совокупности дискретных объектов и отношений между ними. Объекты, если их соотнести с решаемой задачей, образуют ее предметную область. Например, если задача состоит в описании родственных отношений, то предметная область-множество людей, а если задача вычислительная, то предметной областью будет множество целых чисел.

Объекты, при описании их средствами математической логики, должны иметь имена. За определением имен следует описание соотношений между объектами и выражение свойств этих отношений. Построение решения задачи производится на основе логического вывода, манипуляцией предложениями, описывающими задачу.

Объекты, используемые в описании задачи, могут быть произвольными. Например, числовые или литеральные **константы**, **переменные**, принимающие значения из некоторого множества, **функции**.

В математической логике константы, переменные и функции объединены общим названием - терм. **Терм** - это переменные, константы и функции вида:

$f(t_1,t_2,...,t_n),\ (t_1,t_2,...,t_n) \in \rightarrow D^n$

где каждое t_i - терм, а f - n-арный **функциональный символ** или **функтор** (арность - это число аргументов). Особенность функции в том, что она принимает значение элемента предметной области D^n , или иными словами представляет собой некоторое отображение совокупности из n (n - ки) элементов из предметной области в элемент предметной области.

Отношения, определяемые над объектами, отличаются от функций. Отношение определяет совокупность элементов из предметной области и представляет собой отображение из D^n в множество {ИСТИНА, ЛОЖЬ}. Например, отношение мама(x, y)

определяет совокупность пар (x, y), таких, что элементы множества людей x и y находятся в отношении родства мама. Множество пар (x, y) это множество матерей и детей. В математической логике отношениям даются имена, называемые предикатными символами, а сами отношения называются предикатами в последнем примере предикатный символ-мама.

Предикат - это выражение вида $P(t_1,t_2,...,t_m)$,

где каждое t_i - терм, а P - m-арный **предикатный символ**. Формально предикат $P(t_1,t_2,...,t_m)$ можно читать либо как "m - ка $(t_1,t_2,...,t_m)$ принадлежит отношению P", либо как высказывание P справедливо для m-ки $(t_1,t_2,...,t_m)$.

В логике имеется набор связок: &, $\vee$, $\neg$, <- , <->, которые читаются как "и", "или", "не", "если", "тогда и только тогда", соответственно. Связки можно применить для образования логических **формул**, объединяя предикаты и другие формулы. Необходимо отметить, что логические связки:, &, $\vee$, $\neg$, <- , <-> могут быть выражены друг через друга с помощью следующих соотношений, называемых формулами сокращения:

(A & B) означает $\neg$ (A <- B),

(A $\vee$ B) означает ($\neg$ A) <- B,

(A <-> B) означает (A <- B)&(B <- A),

где A и B - суть формулы. Наряду со связками существуют кванторы общности ($\forall$) и существования ($\exists$). Кванторы определяют пределы изменения переменных. Формула, стоящая после квантора называется областью действия этого квантора. Для кванторов тоже существуют формулы сокращения:

$\exists$x A означает $(\neg(\forall x\ (\neg A)))$.

Формула - это либо предикат, либо выражение, составленное из формул с помощью логических связок и кванторов.

Предложение - это формула, в которой каждая переменная находится в области действия квантора общности.

Предложения, построенные в соответствии с введенными выше правилами, образуют **язык логики первого порядка**. В этом языке термы представляют собой объекты, а предикаты отношения между объектами. С помощью этого языка можно описать все задачи, решаемые на ЭВМ. На основе языка логики первого порядка можно построить различные языки логического программирования, различающиеся по правилам формирования предложения. Среди множества всевозможных предложений особое значение имеют предложения, содержащие связки "и" (&) и одну связку "если" (<-). Такие предложения называются дизъюнктами. Более точное определение. **Дизъюнкт** - это предложение вида:

$A_1\&A_2\&...\&A_k$<-$B_1\&B_2\&...\&B_n$,

где $A_1,A_2,...,A_k,B_1,B_2,...,B_n$ - предикаты или **литеры**.

Особое значение для Пролога-Д имеет понятие дизъюнкта Хорна.

Дизъюнктом Хорна называется такой дизъюнкт, у которого k = 0 или k = 1. При k = n = 0 получается **пустой дизъюнкт,** обозначаемый символом **Л**.

Среди множества всех дизъюнктов особую роль играют те, для которых k=1. Если при этом еще и n=0, то такой дизъюнкт называется **фактом**:

A_1.

Если же n>0, дизъюнкт называется **правилом**:

A_1<- $B_1\&B_2\&...\&B_n$,

где A - **голова**, а $B_1\&B_2\&...\&B_n$, - **цели**, образующие **тело** правила.

Базой знаний будет называться всякое (непустое) множество фактов и правил.

Записанная в базе знаний информация представлена в некоторой формально - логической системе и предназначена для изучения информации посредством применения определенного процесса рассуждений, базирующегося на логическом выводе. Эти рассуждения основаны на понятии логического следствия.

Для интерпретации множества предложений S выбирается множество объектов D, называемое **областью интерпретации**.

Интерпретация множества предложений S над D состоит из следующих трех соответствий:

1. Каждой константе (числовой или литеральной) ставится в соответствие некоторый элемент из D.

2. Каждому n-арному функтору из S сопоставляется отображение n-ки из D^n в D.

3. Каждому n-арному предикатному символу из S сопоставляется отображение n-ки из D^n в множество {ИСТИНА, ЛОЖЬ}.

Точное определение логического следствия, основано на понятии выполнимости формулы (предложения). Формула (предложение) выполняется в интерпретации над множеством D тогда и только тогда, когда это предложение в данной интерпретации принимает значение ИСТИНА. Соответственно множество предложений S выполняется в данной интерпретации тогда и только тогда, когда выполняется любое предложение из S. S **логически влечет** некоторое предложение s, тогда и только тогда, когда для любой интерпретации над любой областью D из выполнимости множества S следует выполнимость в ней s.

Смысл отношения $\models$ состоит в том, что если S $\models$ s, то это происходит из-за внутренней структуры предложений множества S, независимо от того, что могут

обозначать составляющие их части в той или иной интерпретации. Использование интерпретаций это один способ получения понятия логического следствия, которое, в конечном счете, не зависит от этих интерпретаций.

Основная задача, стоящая перед пользователем при логическом программировании, состоит в том, чтобы описать условия задачи и критерии ее решения. Таким образом, логическая сущность задачи в этом случае становится более ясной. Существуют методы построения логического следствия, не зависящие от интерпретации. Одним из таких методов является метод резолюций. Он основан на следующем заключении.

Пусть S - некоторое множество дизъюнктов и U - дизъюнкт. Необходимо определить является ли U логическим следствием из S. Для этого множество S дополняется отрицанием ¬U. Если окажется, что объединение множеств S и {¬U} противоречиво, то U логически следует из S.

Предположим, что даны два дизъюнкта C_1 и C_2 из S, а L_1 и L_2 - комплементарные литеры из C_1 и C_2 соответственно (литеры U и ¬U называются **комплементарными**). Дизъюнкция C_1 и C_2 ($C_1 \vee C_2$), в которой предварительно вычеркнуты литеры L_1 и L_2, называется **резольвентой**.

Принцип резолюции основан на том, что резольвента дизъюнктов C_1 и C_2 является логическим следствием C_1 и C_2. Если в результате многократного применения процесса построения резольвент получается пустой дизъюнкт, то множество дизъюнктов S противоречиво. Более того, для противоречивого множества дизъюнктов применениями процесса построения резольвент всегда можно получить пустой дизъюнкт (**Л**).

Подстановка - это конечное множество вида:

$(t_1/v_1,...,t_n/v_n)$,

где v_i - переменные, а t_i - термы.

В приведенном примере (3.2.9'), (3.2.4') использована подстановка (F(a)/x).

В процедуре доказательства по методу резолюций для того, чтобы отождествить комплементарные пары литер необходимо унифицировать выражения, иными словами, необходимо найти подстановку, которая может сделать несколько выражений тождественными.

Унификацией двух выражений E_1 и E_2 с переменными $v_1,...,v_n$ (или без них) называется нахождение такой подстановки (может быть и пустой), что выражения E_1 и E_2 совпадут. Такая подстановка называется унификатором. Если оказалось, что для данных термов не существует унификатора, то термы не унифицируемы.

Процедурная семантика Пролога-Д или исполнение программы. База знаний и следующий за ней вопрос представляют собой **программу** на Прологе-Д.

Независимо от решаемой задачи программа всегда выполняется по одной и той же схеме, определяемой **методом резолюции** - тем методом поиска решения, который использует система логического программирования Пролог-Д.

Поиск решения осуществляется следующим образом. Среди всех элементов множества предложений S отыскивается первое предложение, которого имеет такое же имя, как и первая цель в вопросе.

Если предложение - правило, то это имя должно быть именем головы, или если это предложение факт, имя факта. Проверяется существование подстановки унифицирующей цель вопроса и это предложение. Если такая подстановка не существует, то делается попытка найти следующее по порядку предложение с таким же именем. Если такая подстановка найдена, то резольвентой оказывается тело правила, если предложение - правило или пустой дизъюнкт, если предложение факт. Далее просматриваются цели в теле правила, так как если бы это были цели в вопросе. Просмотр целей осуществляется слева направо.

Арифметика и другие встроенные предикаты в Прологе-Д. Системы логического программирования, к числу которых относится и Пролог-Д, не предназначены для вычислений. Для определения всех математических действий памяти компьютера будет недостаточно, поэтому традиционные действия, связанные с выполнением арифметических операций осуществляются посредством специальных встроенных предикатов. В системе Пролог-Д для выполнения арифметических действий предусмотрен один встроенный арифметический предикат: УМНОЖЕНИЕ(Арг1,Арг2,Арг3,Арг4) Встроенный предикат УМНОЖЕНИЕ имеет четыре аргумента: целых, переменных, конкретизированных целыми, не конкретизированных переменных, допускает обратимость всех аргументов, однако, он может быть использован только в качестве цели в предложении. Предикат УМНОЖЕНИЕ предусматривает реализацию формулы: Арг1*Арг2+Арг3=Арг4.

Предикат предусматривает обратимость аргументов и полностью покрывает арифметические операции в области целых чисел, предусмотренных синтаксисом входного языка (-32767 <число> 32767). Следующая база знаний на языке Пролог-Д показывает, что с помощью данного предиката можно описать любые арифметические операции:

СЛОЖЕНИЕ(X,Y,Z)<-УМНОЖЕНИЕ(1,X,Y,Z);

ВЫЧИТАНИЕ(X,Y,Z)<-УМНОЖЕНИЕ(1,X,Z,Y);

УМНОЖЕНИЕ(X,Y,Z)<-УМНОЖЕНИЕ(X,Y,0,Z);

ДЕЛЕНИЕ(X,Y,Z)<-УМНОЖЕНИЕ(Y,Z,0,X);

Во всех четырех случаях X,Y - суть операнды операций, а Z - результат. Например, СЛОЖЕНИЕ(X,Y,Z) реализует арифметическую операцию сложение: Z=X+Y. Более

подробное описание синтаксиса встроенного предиката УМНОЖЕНИЕ приведено в описании синтаксиса. Предикат УМНОЖЕНИЕ позволяет описывать вычислительные задачи.

Наряду с арифметическим предикатом существуют два предиката БОЛЬШЕ и НЕ. Встроенный предикат БОЛЬШЕ(Арг1,Арг2) предназначен для сравнения двух целых или переменных. Он имеет два аргумента: целых или переменных, конкретизированных целыми. Оба аргумента к моменту выполнения должны быть определены. Если эти требования не выполнены, то появится сообщение об ошибке: "Функция не может быть выполнена.". Предикат выполнен, если Арг1 > Арг2, иначе - не выполнен. Несмотря на то, что предикат БОЛЬШЕ один, его достаточно для описания всех возможных предикатов для сравнения числовой информации: равенство - РАВНО; меньше - МЕНЬШЕ; меньше и равно - МИР и так далее. Это показывает база знаний, приведенная ниже:

РАВНО(X,X);

МЕНЬШЕ(X,Y)<-БОЛЬШЕ(Y,X);

МИР(X,Y)<-НЕ(БОЛЬШЕ(X,Y);

В последнем предложении использован встроенный предикат НЕ, его синтаксис: НЕ(Арг1); Этот встроенный предикат имеет один аргумент, он обязательно должен быть предикатом. Предикат НЕ выполнен тогда и только тогда, когда предикат - аргумент не выполнен. А теперь несложный пример, иллюстрирующий применение БОЛЬШЕ и НЕ.

Рекурсия. Существует огромное количество задач, в которых отношения между объектами можно определить, только используя сами определяемые соотношения. При этом получаются правила, называемые **рекурсивными.** Этот принцип состоит в том, чтобы разбить задачу на случаи, относящиеся к двум группам:

1) тривиальные, или «граничные» случаи;

2) «общие» случаи, в которых решение получается из решений для (более простых) вариантов самой исходной задачи.

Применение рекурсии для описания задач при работе с системами логического программирования широко распространенный прием и используется в Прологе постоянно.

Семантический дуализм Пролога. Создавая пролог программы всегда надо помнить о процедурном и декларативном смысле. Декларативный смысл касается отношений, определенных в программе, т.е. декларативный смысл определяет, что должно быть результатом программы. По сути дела, в программе оказываются объединенными и данные, и способ их обработки. Решение задачи интерпретируется как доказательство логической теоремы. С другой стороны, **процедурный смысл** определяет, как этот результат может быть достигнут, т.е., как реально отношения обрабатываются прологом.

Наилучшим образом Пролог показал себя в следующих областях применения:

- быстрая разработка прототипов прикладных программ;
- автоматический перевод с одного языка на другой;
- создание естественно-языковых интерфейсов для существующих систем;
- символьные вычисления для решения уравнений, дифференцирования и интегрирования;
- проектирование динамических реляционных баз данных;
- экспертные системы и оболочки экспертных систем;
- автоматизированное управление производственными процессами;
- автоматическое доказательство теорем;
- полуавтоматическое составление расписаний;
- системы автоматизированного проектирования;
- базирующееся на знаниях программное обеспечение;
- организация сервера данных или, точнее, сервера знаний, к которому может обращаться клиентское приложение, написанное на каком-либо языке программирования.

Основные приложения, которые связывают с языком Пролог это, как правило, ***разработки в области искусственного интеллекта и экспертных систем.***

Еще одним применением Пролога является интерпретация трансляторов с языков программирования. С помощью выразительных средств синтаксиса этого языка удается достичь высокой степени лаконичности изложения, это делает возможным описать логику работы интерпретатора с помощью нескольких строк текста.

Еще одной возможностью Пролога является интерпретация типов алгоритмов обработки информации: линейного, разветвляющегося циклического и подпрограммы.

Области, для которых Пролог не предназначен: большой объем арифметических вычислений (обработка аудио, видео и т.д.); написание драйверов.

Список литературы:

Основная литература

1. Учебно-методическое пособие по дисциплине Логическое и функциональное программирование [Электронный ресурс]/ - Электрон. текстовые данные.- М.: Московский технический университет связи и информатики, 2016.- 23 с.- Режим доступа: http://www.iprbookshop.ru/61490.html.- ЭБС «IPRbooks».

2. **Городняя Л.В.** Основы функционального программирования [Электронный ресурс]/ Городняя Л.В.- Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 246 с.- Режим доступа:

http://www.iprbookshop.ru/73703.html.- ЭБС «IPRbooks».

3. **Городняя Л.В.** Введение в программирование на Лиспе [Электронный ресурс]/ Городняя Л.В., Березин Н.А.- Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 134 с.- Режим доступа: http://www.iprbookshop.ru/73668.html.- ЭБС «IPRbooks».

4. **Шрайнер П.А.** Основы программирования на языке Пролог [Электронный ресурс]/ Шрайнер П.А.- Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 213 с.- Режим доступа: http://www.iprbookshop.ru/52194.html.- ЭБС «IPRbooks»

Дополнительная литература

1. **Бубнов В.А.** Информатика и информация: знаково-символьный аспект: монография / В.А. Бубнов. - эл. изд. - Москва: БИНОМ. Лаборатория знаний, 2015. - 323 с.: ил. - Библиогр. в кн. - ISBN 978-5-9963-2782-9; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=363359 (12.04.2019).

2. **Давыдова Н.А.** Программирование: учебное пособие / Н.А. Давыдова, Е.В. Боровская. - 3-изд. (эл.). - Москва: БИНОМ. Лаборатория знаний, 2015. - 241 с. - (Педагогическое образование). - ISBN 978-5-9963-2647-1; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=120218 (12.04.2019).

3**. Душкин Р.В.** 14 занимательных эссе о языке Haskell и функциональном программировании / Р.В. Душкин. - Москва: ДМК Пресс, 2011. - 142 с: ил. - ISBN 978-5-94074-691-1; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=129727 (12.04.2019).

4. **Рублев В.С.** Языки логического программирования [Электронный ресурс]/ Рублев В.С.- Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.- 125 с.- Режим доступа: http://www.iprbookshop.ru/73741.html.- ЭБС «IPRbooks».

5. **Салмина Н.Ю.** Функциональное программирование и интеллектуальные системы [Электронный ресурс]: учебное пособие/ Салмина Н.Ю.- Электрон. текстовые данные.- Томск: Томский государственный университет систем управления и радиоэлектроники, 2016.- 100 с.- Режим доступа: http://www.iprbookshop.ru/72216.html.- ЭБС «IPRbooks»

Рекомендуемая литература

1. Бен-Ари М. Языки программирования. Практический сравнительный анализ: Пер. с англ. – М.: Мир, 2008. – 366 с.

2. Себеста У. Основные концепции языков программирования, 3-е изд.: Пер. с англ. - М.: «Вильямс», 2010. – 672 с.

3. Филд А., Харрисон П. Функциональное программирование. - М.: Мир, 1993.

4. Хендерсон П. Функциональное программирование. Применение и реализация. - М.: Мир, 1983.

5. Маурер У. Введение в программирование на языке Лисп. - М.: Мир, 1976.

4. Лавров С., Силагадзе Г. Автоматическая обработка данных. Язык Лисп и его реализация. - М.: Наука, 1978.

6. Хювенен Э., Сеппянен Й. Мир Лиспа. В 2-х т. - М.: Мир, 1990.

7. Ковальски Р. Логика в решении проблем. - М.: Наука,1990.

8. Хоггер К. Введение в логическое программирование. - М.: Мир, 1988.

9. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. - М.: Мир, 1990.

10. Клоксин У., Меллиш К. Программирование на языке Пролог. - М.: Мир, 1987.

11. Братко И. Программирование на языке Пролог для искусственного интеллекта. - М.: Мир, 1990.

12. Малпас Дж. Реляционный язык Пролог и его применение. - Новосибирск: Наука,1990.

13. Ин Ц., Соломон Д. Использование Турбо-Пролога. - М.: Мир, 1993.

14. Янсон А. Турбо-Пролог в сжатом изложении. - М.: Мир, 1991.

15. Путькина Л.В., Пискунова Т.Г. Интеллектуальные информационные системы. Учебное пособие. – СПб.: Изд-во СПбГУП, 2008. – 228 с.