Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Реферат**

**по курсу «Логическое Программирование»**

**Тема:**

**Математическая логика и**

**логическое программирование: сравнение.**

|  |  |
| --- | --- |
| Студент: | Суворова С. А. |
| Группа: | М80-206Б-18 |
| Преподаватель: | Сошников Д.В. |
| Преподаватель: | Левинская М. А. |
| Оценка: |  |
| Дата: |  |

Москва

2019

**Введение**

На нашей планете мы можем объяснить и описать практически любой объект или явление ,используя одну или несколько известных нам наук, к которым также относится и математика. Данное учение имеет очень широкий спектр применения, поэтому её часто делят на составляющие части. Математическая логика является таким же составным куском рассматриваемой науки. Если кратко объяснить, то данная часть разбирает логические суждения используя "математический" язык, то есть различные предикаты и операторы.

В кокой-то момент нашей истории появились вычислительные машины, которые предназначались отчасти для того, чтобы упростить рассмотрение различных процессов и предметов ,и их анализ. В самом начале данные устройства были довольно примитивные, поэтому использовались только для математических исчислений и "общение" с ними происходило довольно незамысловатым способом, то есть с использованием перфокарт- носителей информации из тонкого картона, представляющих информацию наличием или отсутствием отверстий в определённых позициях. Но по мере развития функционал вычислительных машин совершенствовался, и в следствие этого рассматриваемые изобретения стали применятся более широко, то бишь использоваться в различных отраслях науки для различных целей. По этой причини появилась необходимость производить взаимодействия с данными устройствами на довольно высоком уровне. Эту проблему решили путем изобретения различных языков программирования, каждый из которых имел определенный подход для решения задач. Логическое программирование является одним из таких подходов, использующий совокупностью правил без явного указания последовательности их применения для нахождения решения.

Математическая логика и логическое программирование как похожи так и различны в некоторых аспектах. Похожи не только тем ,что оба имеют слово "логика", но и некоторыми особенностями при решении задач. Различны в формате охвата задач(не каждую задачу можно запрограммировать) и возможностью их решения(не каждую задачу можно решить используя только определенный подход).

Мы более подробно рассмотрим математическую логику и логическое программирование, их историю и различные особенности, и на этой основе в конце проведем сравнение и сделаем выводы.

**Математическая логика.**

Математи́ческая ло́гика (теоретическая логика, символическая логика) — раздел [математики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0), изучающий [математические обозначения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_%D0%BE%D0%B1%D0%BE%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [формальные системы](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), [доказуемость](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) математических [суждений](https://ru.wikipedia.org/wiki/%D0%A1%D1%83%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5), природу математического [доказательства](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D0%B0%D0%B7%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) в целом, [вычислимость](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) и прочие аспекты [оснований математики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0).

**История.**

Математическая логика тесно связана с логикой и обязана ей своим возникновением. Основы логики, науки о законах и формах человеческого мышления (отсюда одно из ее названий — формальная логика), были заложены величайшим древнегреческим философом Аристотелем (384—322 гг. до н. э.), который в своих трактатах обстоятельно исследовал терминологию логики, подробно разобрал теорию умозаключений и доказательств, описал ряд логических операций, сформулировал основные законы мышления, в том числе законы противоречия и исключения третьего. Вклад Аристотеля в логику весьма велик, недаром другое ее название — Аристотелева логика. Еще сам Аристотель заметил, что между созданной им наукой и математикой много общего. Он пытался соединить две эти науки, а именно свести размышление, или, вернее, умозаключение, к вычислению на основании исходных положений. В одном из своих трактатов Аристотель вплотную приблизился к одному из разделов математической логики — теории доказательств.

В дальнейшем многие философы и математики развивали отдельные положения логики и иногда даже намечали контуры современного исчисления высказываний, но ближе всех к созданию математической логики подошел уже во второй половине XVII века выдающийся немецкий ученый Готфрид Вильгельм Лейбниц (1646 — 1716), указавший пути для перевода логики «из словесного царства, полного неопределенностей, в царство математики, где отношения между объектами или высказываниями определяются совершенно точно». Лейбниц надеялся даже, что в будущем философы, вместо того чтобы бесплодно спорить, станут брать бумагу и вычислять, кто из них прав. При этом в своих работах Лейбниц затрагивал и двоичную систему счисления.

После Лейбница исследования в этой области вели многие выдающиеся ученые, однако настоящий успех пришел здесь к английскому математику-самоучке Джорджу Булю (1815—1864), целеустремленность которого не знала границ. Материальное положение родителей Джорджа позволило ему окончить лишь начальную школу для бедняков. Спустя какое-то время Буль, сменив несколько профессий, открыл маленькую школу, где сам преподавал. Он много времени уделял самообразованию и вскоре увлекся идеями символической логики. В 1847 году Буль опубликовал статью «Математический анализ логики, или Опыт исчисления дедуктивных умозаключений», а в 1854 году появился главный его труд «Исследование законов мышления, на которых основаны математические теории логики и вероятностей».

Буль изобрел своеобразную алгебру — систему обозначений и правил, применимую ко всевозможным объектам, от чисел и букв до предложений. Пользуясь этой системой, он мог закодировать высказывания (утверждения, истинность или ложность которых требовалось доказать) с помощью символов своего языка, а затем манипулировать ими, подобно тому, как в математике манипулируют числами. Основными операциями булевой алгебры являются конъюнкция (И), дизъюнкция (ИЛИ) и отрицание (НЕ).

Через некоторое время стало понятно, что система Буля хорошо подходит для описания электрических переключательных схем. Ток в цепи может либо протекать, либо отсутствовать, подобно тому, как утверждение может быть либо истинным, либо ложным. А еще несколько десятилетий спустя, уже в XX столетии, ученые объединили созданный Джорджем Булем математический аппарат с двоичной системой счисления, заложив тем самым основы для разработки цифрового электронного компьютера.

Отдельные положения работ Буля в той или иной мере затрагивались и до, и после него другими математиками и логиками. Однако сегодня в данной области именно труды Джорджа Буля причисляются к математической классике, а сам он по праву считается основателем математической логики и тем более важнейших ее разделов — алгебры логики (булевой алгебры) и алгебры высказываний.

Большой вклад в развитие логики внесли и русские ученые П. С. Порецкий (1846−1907), И. И. Жегалкин(1869−1947).

В XX веке огромную роль в развитии математической логики сыграл Д. Гильберт (1862−1943), предложивший программу формализации математики, связанную с разработкой оснований самой математики. Наконец, в последние десятилетия XX века бурное развитие математической логики было обусловлено развитием теории алгоритмов и алгоритмических языков, теории автоматов, теории графов (С.К. Клини, А. Черч, А. А Марков, П. С. Новиков, Гегель и многие другие)

**Основные составляющие логического метода решения задач.**

1.Анализ – разделение объекта на составные части с целью их самостоятельного изучения.

2. Синтез – реальное или мысленное объединение различных сторон, частей предмета в единое целое. Это процесс, обратный анализу.

3 .Абстрагирование – выделение важных признаков и отвлечение от несущественных.

4 .Обобщение – мысленное объединение предметов и явлений по их общим и существенным качествам.

5. Индукция – логический прием исследования, связанный с обобщением

результатов наблюдений и экспериментов и движением мысли от единичного

к общему. Это умозаключение от частных случаев к общему положению.

6. Дедукция – во-первых, переход в процессе познания от общего к единичному (частному), выведение единичного из общего; во-вторых, процесс логического вывода, т.е. перехода по тем или иным правилам логики от некоторых данных предложений к их следствиям (заключениям).

7. Аналогия – метод научного познания, при котором устанавливается сходство в некоторых

сторонах, качествах и отношениях между нетождественными объектами.

8 .Моделирование как метод познания представляет собой воспроизведение характеристик некоторого объекта на другом объекте, специально созданном для их изучения. Последний называется моделью.

9 .Сравнение – сопоставление предметов, явлений, процессов с целью установления сходства и различия в целом или в каких-либо признаках.

**Методы решения логических задач.**

1.Метод рассуждений.

Его идея состоит в том, что мы проводим рассуждения,

используя последовательно все условия задачи, и приходим к выводу, который и будет являться ответом задачи.

2.Метод таблиц.

Это способ решения текстовых логических задач с помощью построения таблиц. Таблицы позволяют наглядно представить условие задачи или ее ответ и помогают делать правильные логические выводы в ходе решения задачи.

3.Метод блок-схем.

Описать последовательность выполнения операций, определить порядок их выполнения и фиксировать состояния.

4.Метод бильярда.

Нарисовать бильярдный стол и интерпретировать действия движениями бильярдного шара, фиксирования состояний в отдельной таблице.

5.Метод графов.

При решении задачи составить граф и ,используя различные способы обхода ,найти решение.

6.Метод кругов Эйлера.

Представить логическое утверждение в виде кругов Эйлера, где различные логические операции(дизъюнкция, конъюнкция и другие) представлены графически.

**Применение математической логики.**

1.Логика оказала влияние на развитие математики, прежде всего теории множеств, функциональных систем, алгоритмов, рекурсивных функций.

2. В гуманитарных науках (логика, криминалистика).

3. Математическая логика является средством для изучения деятельности мозга - для решения этой самой важной проблемы биологии и науки вообще.

4. Идеи и аппарат логики используется в кибернетике, ВТ и электротехнике (построены компьютеры на основе законов математической логики).

1938 г. – американский математик и инженер Клод Шеннон связал Булеву алгебру (аппарат математической логики), двоичную систему кодирования и релейно-контактные переключательные схемы, заложив основы будущих ЭВМ.

5. Идеи и аппарат логики используется в программировании, базах данных и экспертных системах. (PROLOG – язык логического программирования).

**Логическое программирование.**

Логическое программирование представляет собой тип [программирования парадигмы](https://ru.qwertyu.wiki/wiki/Programming_paradigm) , которая в значительной степени основана на [формальной логике](https://ru.qwertyu.wiki/wiki/Formal_logic) . Любая программа , написанная на логическом [языке программирования](https://ru.qwertyu.wiki/wiki/Programming_language) представляет собой набор предложений в логической форме, выражающие факты и правили о некоторой предметной области, которые впоследствии помогают найти ответ на поставленную задачу.

**История.**

Применение математической логики для представления и выполнения компьютерных программ также является особенностью [лямбда - исчисления](https://ru.qwertyu.wiki/wiki/Lambda_calculus) , разработанной [Алонзо церкви](https://ru.qwertyu.wiki/wiki/Alonzo_Church" \o "Алонзо Чёрч) в 1930 - х годах. Тем не менее, первое предложение использовать [смысловую](https://ru.qwertyu.wiki/wiki/Clausal_normal_form) форму логики для представления компьютерных программ было сделано Корделл Грином. Это используется аксиоматизацию подмножества [LISP](https://ru.qwertyu.wiki/wiki/LISP) , вместе с представлением на вход-выход, для вычисления отношения путем имитации выполнения программы в LISP. Фостер и Элкок в [ABSYS](https://ru.qwertyu.wiki/wiki/Absys) , с другой стороны, использовали комбинацию уравнений и лямбда - исчисления в assertional языке программирования , который не накладывает никаких ограничений на порядок , в котором выполняются операции.

Логическое программирование в его нынешнем виде может быть прослежена до дебатов в конце 1960 - х и начале 1970 - х о декларативным и процедурным представления знаний в области искусственного интеллекта. Сторонники декларативных представлений были особенно работающим в [Стэнфорд](https://ru.qwertyu.wiki/wiki/Stanford_University) , связанный с [Джоном Маккарти](https://ru.qwertyu.wiki/wiki/John_McCarthy_(computer_scientist)) , [Бертрам Рафаэлем](https://ru.qwertyu.wiki/wiki/Bertram_Raphael" \o "Bertram Raphael) и [Корделл Грином](https://ru.qwertyu.wiki/w/index.php?title=Cordell_Green&action=edit&redlink=1" \o "Корделл Green (страница не существует)) , и в [Эдинбурге](https://ru.qwertyu.wiki/wiki/University_of_Edinburgh) , с [Джоном Аланом Робинсоном](https://ru.qwertyu.wiki/wiki/John_Alan_Robinson) (академическим приезжим из [Сиракузского университета](https://ru.qwertyu.wiki/wiki/Syracuse_University" \o "Сиракузы университета) ), [Пэт Хейс](https://ru.qwertyu.wiki/wiki/Patrick_J._Hayes) и [Роберт Ковальским](https://ru.qwertyu.wiki/wiki/Robert_Kowalski) . Сторонники процедурного представления были в основном сосредоточены в [MIT](https://ru.qwertyu.wiki/wiki/MIT) под руководством [Марвина Мински](https://ru.qwertyu.wiki/wiki/Marvin_Minsky" \o "Марвин Мински) и [Пейперт](https://ru.qwertyu.wiki/wiki/Seymour_Papert) .

Хотя оно было основано на доказательстве методов логики, [планировщик](https://ru.qwertyu.wiki/wiki/Planner_(programming_language)) , разработанный в Массачусетском технологическом институте, был первый язык , чтобы появиться в этой proceduralist парадигмы. Планировщик признаки шаблона направленного вызова процедурных планов из целей (т.е. цели сокращения или [обратный логический вывод](https://ru.qwertyu.wiki/wiki/Backward_chaining) ) и из утверждений (т.е. [вперед цепочке](https://ru.qwertyu.wiki/wiki/Forward_chaining) ). Наиболее влиятельная реализация Planner было подмножество Planner, которая называется Micro-планировщик, реализованный [Gerry Зуссмана](https://ru.qwertyu.wiki/wiki/Gerald_Jay_Sussman" \o "Джеральд Джей Сассмен) , [Юджин Чарниак](https://ru.qwertyu.wiki/wiki/Eugene_Charniak) и [Терри Винограда](https://ru.qwertyu.wiki/wiki/Terry_Winograd" \o "Терри Винограда) . Он был использован для реализации Винограда на естественном языке понимание программы [SHRDLU](https://ru.qwertyu.wiki/wiki/SHRDLU) , который был знаковым в то время. Чтобы справиться с очень ограниченными системами памяти в то время, планировщик использовал возвраты структуры управления таким образом , что только один из возможных путей вычисления должны были быть сохранены в то время. Планировщик привел к языкам программирования QA-4, Popler, Conniver, QLISP и одновременно язык Ether.

Hayes и Ковальский в Эдинбурге пытались примирить логики на основе декларативного подхода к представлению знаний с процедурным подходом планировщика. Hayes (1973) разработал эквациональный язык, Golux, в котором различные процедуры могут быть получены путем изменения поведения теоремы расстойки. Kowalski, с другой стороны, разработаны [SLD разрешение](https://ru.qwertyu.wiki/wiki/SLD_resolution) , вариант SL-разрешения, и показал , как он относится последствия как процедуры цель сокращения. Kowalski сотрудничал с [Colmerauer](https://ru.qwertyu.wiki/wiki/Alain_Colmerauer" \o "Ален Колмерауэр) в Марселе, который разработал эти идеи в разработке и реализации языка программирования [Prolog](https://ru.qwertyu.wiki/wiki/Prolog" \o "пролог) .

[Ассоциация логического программирования](https://ru.qwertyu.wiki/wiki/Association_for_Logic_Programming) была создана для содействия логического программирования в 1986 году.

Пролог породил языки программирования [ALF](https://ru.qwertyu.wiki/wiki/Algebraic_Logic_Functional_programming_language) , [Fril](https://ru.qwertyu.wiki/wiki/Fril) , [гёделевское](https://ru.qwertyu.wiki/wiki/G%C3%B6del_(programming_language)) , [Mercury](https://ru.qwertyu.wiki/wiki/Mercury_programming_language) , [Oz](https://ru.qwertyu.wiki/wiki/Oz_(programming_language)" \o "Oz (язык программирования)) , [Ciao](https://ru.qwertyu.wiki/wiki/Ciao_(programming_language)" \o "Ciao (язык программирования)) , [Visual Prolog](https://ru.qwertyu.wiki/wiki/Visual_Prolog" \o "Visual Prolog) , [XSB](https://ru.qwertyu.wiki/wiki/XSB) и [λProlog](https://ru.qwertyu.wiki/wiki/%CE%9BProlog" \o "ΛProlog) , а также множество [параллельных логических языков программирования](https://ru.qwertyu.wiki/wiki/Concurrent_logic_programming) , [ограничение логики программирования](https://ru.qwertyu.wiki/wiki/Constraint_logic_programming) языков и [DataLog](https://ru.qwertyu.wiki/wiki/Datalog" \o "Лог данных) .

**Основные концепции.**

1.Логика и управление

Логическое программирование можно рассматривать как контролируемые дедукции. Важное понятие в логическом программировании является разделением программ в их логической составляющую и их компоненты управления. С чистыми языками логического программирования, логика компоненты в одиночку определяют решения, полученные. Компонент управления может быть изменен, чтобы обеспечить альтернативные способы выполнения логической программы. Это понятие захватывается лозунгом

Алгоритм = Логика + Control

где «логика» представляет собой логическую программу и «Control» представляет собой различные стратегии, теорема доказав.

2.Решение проблем

В упрощенном, пропозициональный случае , в котором логика программа и атомная цель верхнего уровня не содержат никаких переменных, назад рассуждение определяет [и-или дерево](https://ru.qwertyu.wiki/wiki/And-or_tree" \o "И или дерево) , которое представляет собой пространство поиска для решения поставленной цели. Цель верхнего уровня является корнем дерева. Учитывая , любой узел в дереве и любое положение, голова которого соответствует узлу, существует множество дочерних узлов , соответствующих подцели в теле статьи. Эти дочерние узлы группируются вместе посредством «и». Альтернативные наборы детей , соответствующих альтернативных путей решения узла сгруппированы по оператору «или».

Любая стратегия поиска может быть использована для поиска этого пространства. Пролог использует последовательный, лифо, возвратов стратегию, в которой только одна альтернатива и одна подцель считается одновременно. Другие стратегии поиска, такие как параллельный поиск, интеллектуальные возвраты, или лучше всего первый поиск, чтобы найти оптимальное решение, также возможны.

В более общем случае, когда могут быть использованы Подцели доля переменных, другие стратегии, такие как выбор подцель , который наиболее инстанцированный или что достаточно инстанцированный так , что применяется только одна процедура. Такие стратегии используются, например, в [параллельной логики программирования](https://ru.qwertyu.wiki/wiki/Concurrent_logic_programming) .

3.Отрицание как отказ

Для большинства практических применений, а также для приложений, требующих немонотонных рассуждений в области искусственного интеллекта, Horn раздел логика программа должна быть расширена до нормальных логических программ, с отрицательными условиями. Положение в нормальной логической программе имеет вид:

H :- A1, …, An, not B1, …, not Bn.

и читаются декларативно как логическое следование:

H if A1 and … and An and not B1 and … and not Bn.где Hи все и атомарные формулы. Отрицание в отрицательных литералов обычно называют как « [отрицание как провал](https://ru.qwertyu.wiki/wiki/Negation_as_failure) », потому что в большинстве реализаций, отрицательное условие показано провести, показывая , что положительное условие не выполнено.

4.Представление знаний

Тот факт , что Хорн может быть дан процедурной интерпретация и, наоборот, что процедуры редукции цели может быть понята как Хорн + назад рассуждение означает , что логика программа сочетает декларативные и процедурные представления [знаний](https://ru.qwertyu.wiki/wiki/Knowledge_representation) . Включение [отрицания как неудачи](https://ru.qwertyu.wiki/wiki/Negation_as_failure) означает , что логическое программирование является своим родом [немонотонной логики](https://ru.qwertyu.wiki/wiki/Non-monotonic_logic) .

Несмотря на свою простоту по сравнению с классической логикой, эта комбинация Хорн и отрицания как неудачи, оказалась удивительно выразительными. Например, он обеспечивает естественное представление для здравого смысла законов причины и следствия, а формализуется как [ситуация исчисления](https://ru.qwertyu.wiki/wiki/Situation_calculus) и [исчисления событий](https://ru.qwertyu.wiki/wiki/Event_calculus) . Кроме того , было показано , что соответствует вполне естественно для Полуформальные языка законодательства. В частности, Праккен и Sartor кредитуют представление Закона о британском гражданстве в качестве логической программы с бытием «огромное влиянием на развитие вычислительных представлений законодательства, показывая , как логическое программирование позволяет интуитивно привлекательные представления , которые могут быть непосредственно развернуты для создания автоматического умозаключения» .

**Различные области применения.**

1.Пролог

Язык программирования [Пролог](https://ru.qwertyu.wiki/wiki/Prolog) был разработан в 1972 году [Ален Колмерауэр](https://ru.qwertyu.wiki/wiki/Alain_Colmerauer) . Она возникла из сотрудничества между Colmerauer в [Марселе](https://ru.qwertyu.wiki/wiki/Marseille) и [Роберта Ковальски](https://ru.qwertyu.wiki/wiki/Robert_Kowalski) в Эдинбурге. Colmerauer работал на [понимание естественного языка](https://ru.qwertyu.wiki/wiki/Natural_language_understanding) , используя логику для представления семантики и используя разрешение для вопроса-ответа. В течение лета 1971 года, Colmerauer и Kowalski обнаружили , что клаузальная форма логики может быть использована для представления [формальных грамматик](https://ru.qwertyu.wiki/wiki/Formal_grammars) и что теорема разрешения испытатели могут быть использованы для анализа. Они отметили , что некоторые доказательства теоремы, как гипер-разрешение, ведут себя как восходящие парсер и другие, как [SL-разрешение](https://ru.qwertyu.wiki/wiki/SLD_resolution) (1971), ведут себя как сверху вниз анализаторы.

Это было следующим летом 1972 года, что Ковальский, снова работать с Colmerauer, разработал процедурную интерпретацию последствий. Эта двойная декларативная / процедурная интерпретация позже стала формализованным в обозначениях Prolog

H :- B1, …, Bn.

которые можно читать (и используется) как декларативно и процедурно. Кроме того , стало ясно , что такие положения могут быть ограничены определенными пунктами или [Хорна](https://ru.qwertyu.wiki/wiki/Horn_clause) , где H, , ..., являются все атомные логики предикатов формулы, и что SL-разрешение может быть ограничено (и обобщенно) для LUSH или [SLD-разрешением](https://ru.qwertyu.wiki/wiki/SLD_resolution) . Процедурная интерпретация Ковальского и LUSH были описаны в 1973 записке, опубликованная в 1974 году.B1Bn

Colmerauer с Филиппом Руссель, использовал эту двойную интерпретацию положений в качестве основы Пролога, который был реализован летом и осенью 1972 года первой программы Prolog, также написана в 1972 году и реализуется в Марселе, был французский вопрос, ответив система , Использование Prolog в качестве практического языка программирования был дан большой импульс в развитии компилятора Дэвид Уоррен в Эдинбурге в 1977 г. Эксперименты показали , что Эдинбург Пролог может конкурировать со скоростью обработки других символических языков программирования , таких как [Lisp](https://ru.qwertyu.wiki/wiki/Lisp_(programming_language)" \o "Lisp (язык программирования)) . Edinburgh Prolog стал де - факто стандартом и сильно повлиял на определение [ISO](https://ru.qwertyu.wiki/wiki/International_Organization_for_Standardization) стандарта Пролога.

# 2. Программирование в ограничениях.

Программирование в ограничениях  ‑ программ­ная технология для декларативного описания и эффективного решения боль­ших комбинаторных задач в областях планирования и календарного планиро­вания.

Цель программирования в ограничениях состоит в разработке языков программирования для задания ограничений и процедур поиска задач удовлетворения ограничений(УО). Целью решения задачи УО является нахождение значений переменных, удовлетворяющих заданным ограничениям.

Процедура поиска в УО неявно описывает обход дерево поиска. При этом она не описывает, как обходить это дерево, это уже ‑ задача стратегий поиска. Обычно дерево поиска обходится с помощью поиска в глубину (depth-first search).

Благодаря успешному решению многих прикладных задач, стратегии поиска стали неотъ­емлемой частью языков программирования в ограничениях.

В частности, программная система OZ впервые ввела спецификацию общих стратегий поиска, которые могут быть заданы независимо от проце­дуры поиска.

Поисковый язык SALSA (Laburthe & Caseau, 1998)19 также содержит ряд заданных общих стратегий поиска.

Основные идеи, лежащие в основе программирования в ограничениях, просты: декларативное представление ограничений задачи, со­вмещенное с общими методами решения типа хронологического поиска с возвратами или локального поиска.

Программирование в ограничениях имеет множество сильных сторон: мощные языки моделирования, позволяющие представить сложные и дина­мические прикладные задачи; быстрые методы вывода общего назначения типа вынуждения дуговой совместимости, для усечения части пространства поиска; быстрые методы вывода специального назначения, связанные с гло­бальными ограничениями; гибридные методы, сочетающие преимущества методов программирования в ограничениях и подходов исследования опера­ций; методы локального поиска, позволяющие быстро находить решения, близкие к оптимальным; большой диапазон расширений типа мягких огра­ничений и распределенного решения ограничений, с помощью которых воз­можно более адекватное моделирование практических задач.

3.Программирование металогики

Поскольку математическая логика имеет долгую традицию различения [объектного языка](https://ru.qwertyu.wiki/wiki/Object_language) и метаязыка, логическое программирование также позволяет [программировать метауровень](https://ru.qwertyu.wiki/w/index.php?title=Metalevel_programming&action=edit&redlink=1) . Самая простая программа металогики является так называемым « [ваниль](https://ru.qwertyu.wiki/wiki/Vanilla_(computing)) » мета-интерпретатор:

solve(true).

solve((A,B)):- solve(A),solve(B).

solve(A):- clause(A,B),solve(B).

где истинные представляет собой пустую конъюнкцию, и п (А, В) означает, что существует положение объекта на уровне вида А: - B.

Программирование металогики позволяет объектно-уровеневое и метауровневое представление объединить, как в естественном языке. Он также может быть использован для реализации какой - либо логики, указанная с помощью [правил вывода](https://ru.qwertyu.wiki/wiki/Inference_rule) . Металогика используется в логическом программировании для реализации метапрограмм, которыми манипулируют другие программы, базы данных, базы знаний или аксиоматические теории ,как данные.

**Сравнение математической логики и логического программирования.**

На основе всего вышеприведенного насчет математической логики и логического программирования не трудно определить какие часки у них совпадают и какие различаются.

Совпадения:

1.Оба решают задачи логического характера.

2. Оба используют некоторый данный набор фактов для определения решения.

3. Имеют схожие методы нахождения решения.

Различия:

1.Логическое программирование следует из математической логики, обратное неверно.

2.Некоторые методы ,использующиеся в математической логике, не удобны в логическом программировании(метод кругов Эйлера).

3.Математическая логика покрывает большее пространство задач нежели логическое программирование.

**Выводы.**

Математическая логика и логическое программирование два очень тесно связных понятия, которые друг друга дополняют: логическое программирование помогает быстро и удобно решать логические задачи, а математическая логика предоставляет теорию для эффективного нахождения решения. Оба понадобились в определенный момент развития человечества и помогли ему усовершенствовать некоторые области жизни. Математическая логика и логическое программирование применяются и по сей день не только в промышленности и науке ,но и в образовании.

**Источники:**

1.https://docviewer.yandex.ru/view/77186188/?page=6&\*=%3D&lang=ru

2.https://www.webkursovik.ru/kartgotrab.asp?id=-51843

3.https://studfile.net/preview/5023847/page:15/

4.https://ru.qwertyu.wiki/wiki/Logic\_Programming#Concepts

5. https://studfile.net/preview/2277966/