# Prolog

Пролог, как следует из самого названия, представляет собой краткую форму ЛОГического ПРОграммирования. Это логический и декларативный язык программирования. Логическое программирование — это одна из парадигм компьютерного программирования, в которой операторы программы выражают факты и правила о различных проблемах в рамках системы формальной логики. Здесь правила написаны в виде логических выражений, где присутствуют заголовок и тело. Например, H — это голова, а B1, B2, B3 — элементы тела. Теперь, если мы утверждаем, что «H истинно, когда все B1, B2, B3 истинны», это правило. С другой стороны, факты подобны правилам, но без тела. Итак, пример факта — «H истинно». Некоторые языки логического программирования, такие как Datalog или ASP, известны как чисто декларативные языки. Эти языки допускают утверждения о том, что программа должна выполнить. Нету таковых пошаговых инструкций, как выполнить данное задание. Однако другие языки, такие как Пролог, имеют как декларативные, так и императивные свойства. Это может также включать процедурные утверждения, такие как «Чтобы решить проблему H, выполните B1, B2 и B3». Пролог используется в различных областях. Он играет жизненно важную роль в Интеллектуальный поиск в базе данных , Понимание естественного языка, Язык спецификации, Машинное обучение, Планирование роботов, Системы автоматизации , Решение проблем

Рассмторим различия между логическим программированием и традиционными языками функционального программирования. Из иллюстрации видно, что в функциональном программировании мы должны определить процедуры и правила их работы. Эти процедуры работают шаг за шагом, чтобы решить одну конкретную проблему на основе алгоритма. С другой стороны, для логического программирования мы предоставим базу знаний. Используя эту базу знаний, машина может найти ответы на заданные вопросы, что полностью отличается от функционального программирования. В функциональном программировании мы должны упомянуть, как может быть решена одна проблема, но в логическом программировании мы должны указать, для какой проблемы мы действительно хотим получить решение. Затем логическое программирование автоматически находит подходящее решение, которое поможет нам решить эту конкретную проблему.

посмотрим еще некоторые различия: Функциональное программирование следует архитектуре фон Неймана или использует последовательные шаги, синтаксис представляет собой последовательность выражений, таких как (a, s, I), вычисление происходит за счет последовательного выполнения выражений, логика и элементы управления смешаны вместе. Логическое программирование использует абстрактную модель или имеет дело с объектами и их отношениями, синтаксис в основном представляет собой логические формулы (предложения Хорна), он вычисляется путем вычитания предложений, логика и элементы управления могут быть разделены.

В Прологе нам не нужно упоминать, как может быть решена проблема, просто нужно указать, в чем проблема состоит, чтобы Пролог решил ее автоматически. Однако мы должны давать подсказки в качестве метода решения. Язык Пролог в основном состоит из трех различных элементов. Факты — это предикат, который верен, например, если мы говорим: «Том — сын Джека», то это факт. Правила — это вымирание фактов, содержащих условные предложения. Для выполнения правила эти условия должны выполняться. Например, если мы определяем правило как grandfather(X, Y) :- father(X, Z), parent(Z, Y), Это означает, что для того, чтобы X был дедушкой Y, X должен быть отцом Z, Z должен быть родителем Y. Вопросы. И для запуска программы на прологе нам нужны некоторые вопросы, и на эти вопросы можно ответить с помощью данных фактов и правил.

Мы можем определить факт как явную связь между объектами и свойствами, которыми эти объекты могут обладать. Таким образом, факты безусловно истинны по своей природе. Предположим, у нас есть некоторые факты (на слайде). Итак, вот некоторые факты, которые безусловно верны. На самом деле это утверждения, которые мы должны считать истинными. Рекомендации по написанию фактов. Имена свойств/отношений начинаются со строчных букв. Имя отношения отображается как первый термин. Объекты отображаются как аргументы, разделенные запятыми, в круглых скобках. "." должна заканчивать факт. Объекты также начинаются со строчных букв или цифр (например, 1234) и могут быть строками символов, заключенными в кавычки.

Мы можем определить правило как неявную связь между объектами. Так что факты условно истинны. Таким образом, когда одно связанное условие истинно, предикат также истинен. Предположим, у нас есть некоторые правила —Том голоден, если он ищет еду. Джек и Били — друзья, если они оба любят играть в футбол. Это некоторые правила, которые верны условно, поэтому, когда правая часть истинна, левая часть также верна. Здесь символ ( :- ) будет произноситься как «Если» или «подразумевается». Это также известно как символ шеи, левая сторона этого символа называется Головой, а правая сторона называется Телом. Здесь мы можем использовать запятую (,), известную как соединение, и мы также можем использовать точку с запятой, известную как дизъюнкция.

Мы знаем, что в логическом программировании есть три основных компонента: факты, правила и запросы. Среди этих трех, мы можем собрать факты и правила в нечто целое, что формирует базу знаний. Таким образом, мы можем сказать, что база знаний представляет собой набор фактов и правил. Предположим, у нас есть самая первая база знаний под названием KB1. Здесь, в KB1, у нас есть некоторые факты. Факты используются для утверждения вещей, которые безоговорочно верны в интересующей области. Предположим, что мы знаем, что Паша, Таня и Даша — три девушки, среди которых Паша умеет готовить. Теперь мы можем использовать эту базу знаний, задав несколько запросов. «Паша — девочка?», — ответит «да», «Серега — девочка?» тогда он ответит «Нет», потому что не знает, кто такой Серега. Наш следующий вопрос «Может ли Паша готовить?», он ответит «да», но если мы зададим тот же вопрос Тане, он ответит «Нет».

Это была только базовая, малая чать возможностей Prolog, Вот например две программы, одна для поиска максимума и минимума, вторая для вычисления сопративления цепи.