Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра: 806 «Вычислительная математика и программирование»

Факультет: «Прикладная математика и физика»

Дисциплина: «Логическое программирование»

Реферат

Тема:

«Как научить младшего брата/сестру логическому программированию»

Группа: 8О-207Б

Студент: Сорокин Денис Михайлович

Преподаватель: Левинская Мария Александровна

Вариант: №13

Москва

2016

# Как научить младшего брата/сестру логическому программированию

Действительно, как же? Задача не из легких, особенно, когда у тебя только старший брат. Обратился я к нему за советом, как бы он меня учил, на что получил: «Какое логическое программирование? Отвали». Что ж придется придумывать себе младшего брата… Или сестру. Нет, пожалуй, все же брат, после моего «гениального» курса мы бы смогли сходить поиграть в хоккей на улице!

Сколько ему лет? Знаком ли он с программированием? Проявляет ли интерес к программированию вообще? Многие вопросы предстоит выяснить, прежде чем начать его обучать.

Ладно, допустим, ему 10-12 лет, с программированием он не знаком вообще. То есть, я – старший брат, сейчас устрою ему путешествие в неизведанную страну логического программирования! Великолепная перспектива!

С чего же начать… Естественно, грузить его сложными терминами пока еще рано, поэтому придется все показывать и объяснять максимально просто, чтобы не запутать его. Также плюс в том, что, скажем так, голова у него «чистая» в плане программирования, то есть Пролог будет для него первым языком программирования, и ничего не будет искажать его восприятие. Начал бы я с примера (так для меня будет проще всего начать что-то рассказывать и объяснять):

**isBigger(X, Y) :-**

**X > Y.**

И сразу бы показал, как это все работает. Тут мы и познакомились с главной фишкой логического программирования – логикой.

«Правда» или «ложь» вот таким будет вывод этой программы, если давать на вход два числа. И тут главное, чтобы малой прочувствовал, как и почему это происходит. Я ему в этом должен помочь: объясню, что мы задаем правило, а после **:-** пишем условия, в зависимости от которых будет правда или ложь. Таким образом, если **Х** больше **У**, тогда правда, иначе ложь. И вот с такого несложного примера мы и начнем обучение.

Далее начнем знакомится с синтаксисом и основными понятиями пролога. Для этого разберем поближе разберем наш пример.

Итак**, isBigger(X, Y)** называется предикатом, то есть элементом языка, который задает логические взаимоотношения между его объектами. Пролог по большей части состоит из предикатов. Они, в свою очередь, образуют факты и правила, которые вместе называются утверждением. Весь этот маленький пример есть одно утверждение. Факт здесь - **isBigger(X, Y),** он утверждает, что **X > Y.** Далее после **:-** идет набор правил, которые определяют достоверность этого факта, в данном случае правило одно - **X > Y.**

Нужно заметить, что название предикатов пишется с маленькой буквы, а переменные начинаются с больших.

Факты могут быть и без правил, просто утверждающие. Пример:

**favorite\_food(denis, cesar).**

**favorite\_food(misha, chicken).**

Таким образом, я задал факты, из которых можно видеть, что любимая еда Дениса – цезарь, а любимая еда Миши – курица. Теперь я бы показал брату, как сработает программа, если задать ей **favorite\_food(denis, X).** Этот пример поможет понять, как работает Пролог.

Дальше я бы хотел рассказать, что существуют такие переменные, как анонимные. Для этого немного модифицируем наш предыдущий пример:

**favorite\_food(denis, \_).**

Теперь вывод программы даст обычный **true.** Что же сейчас произошло? А произошло то, что мы утверждаем, что у Дениса есть любимая еда, но неизвестно какая, поэтому Пролог выводит нам лишь подтверждение наличия любимого блюда. Символом **“\_”** в Прологе обозначаются анонимные переменные, то есть те, значение которых может быть любым.

Далее поговорим о правилах, точнее о том, как связывать их между собой. Если перечисляемые правила должны выполняться одновременно для подтверждения факта, тогда между ними следует ставить **“,” – логическое «И».** Если же для подтверждения факта достаточно выполнения хотя бы одного правила, то **“;” – логическое «ИЛИ».**

**champion(X, Y, Z) :-**

**wins(X, Y),**

**wins(X, Z).**

Этот пример показывает использование **«И».** Если **Х** обыграл всех своих соперников, то он чемпион. Рассмотрим другой пример с **«ИЛИ»:**

**not\_loser(X, Y, Z) :-**

**wins(X, Y);**

**wins(X, Z).**

Здесь я утверждаю, что **Х** не проиграл, если обыграл хотя бы одного из своих соперников. Стоит заметить, что конец набора правил обозначается **“.”.** Так вот, с помощью данных примеров мы разобрались, как работать с простой логикой на языке Пролог. Едем дальше.

Тут можно рассказать про арифметику на Прологе. К тому же брат наверняка учиться в школе, ему будет интересно практическое применение программирования. Допустим, нужно будет решать линейные уравнения вида (**ax + b = 0)**:

**multiply(X, A, B) :-**

**X is -B \* A.**

Теперь у нас есть программа, которая решает простенькие уравнения, стоит лишь указать коэффициенты. Здесь можно объяснить, что операторы арифметических операций здесь работают, как на калькуляторах, а **is –** оператор присваивания значения переменной. Ребенок сможет вычислять на Прологе, если ему наскучит делать это ручками!

Дальше нужно будет посмотреть на реакцию моего брата. Если он сильно загрузился и не особо понимает, что здесь вообще происходит, то может быть еще рано для Пролога, либо я плохой учитель… В любом случае, на демонстрации логических фокусов можно и остановиться. Но если ученик хочет продолжения, то следующим шагом в рассказе будут списки в Прологе.

Как объяснить ребенку, что такое список в логическом программировании… Придумал! В Прологе есть такой тип данных, как список. Представь, что клюшки, которые у тебя есть, записали друг за другом. Так вот, Пролог понимает такие списки, и ты можешь поспрашивать его на счет них!

**[bauer, reebok, ccm, fisher]**

Теперь нужно объяснить стандартные предикаты обработки списков.

**lenght** – выясняет, сколько в списке элементов, с помощью него можно проверить, все ли клюшки включили в список или какую-то забыли.

**member** – проверяет, есть ли такой элемент в списке, с помощью него можно узнать есть ли такая клюшка в списке или стоит ее добавить.

**append** – с помощью этого предиката можно совместить списки. Например, если бы ты с другом решил объединить свои клюшки.

**remove** – удаляет элемент из списка, допустим, если у тебя к несчастью сломалась клюшка.

Таким образом, проводя аналогию с тем, что интересно брату и, подкрепив данные аналогии примерами, ребенок вполне вероятно поймет, как примитивно пользоваться списками.

Учить детей Прологу, вещь тяжелая, особенно, когда сам не до конца его понимаешь… Но, я все же попробовал! Мой воображаемый брат в совершенстве им овладел и абсолютно превзошел меня в этом вопросе, горжусь! Другое дело, если бы был у меня настоящий младший брат или сестра, вот тогда бы мы намучались…

Я считаю, что для маленьких детей Пролог стоит изучать не дальше примитивных логических задач, ну, или хотя бы не дальше списков на том же уровне. Потому что для ребенка будет тяжеловато ориентироваться в данном вопросе без достаточного развития и опыта. Так что обучать младших Прологу, дело хорошее, главное найти грань, где ребенку интересно и не очень сложно, чтобы не запутать его и навсегда не отбить интерес к программированию.