Министерство образования Российской Федерации

Московский Государственный Технический Университет им. Н.Э. Баумана

Отчет по лабораторной работе №5

По курсу «Функциональное и логическое

программирование»

Студент Медведев А.В.

Группа ИУ7-62 Преподаватель Толпинская Н.Б.

# Теоретическая часть

Алгоритм интерпретации программы (выполнения цели) в системе ПРОЛОГ.

Вход: цель p(A1,A2,…,An)

1. Найти в памяти процедуру по имени p с арностью n.
2. Унифицировать входные параметры цели с заголовком найденной процедуры.
3. Если унификация параметров прошла, выполнить первую цель из тела найденной процедуры.
4. Если унификация не прошла, найти следующий вариант процедуры P/n.  
   5. Если следующий вариант процедуры P/n не найден, закончить работу с результатом FAIL.
5. Если первая цель выполнилась, перейти к выполнению следующей цели.
6. Если следующая цель не выполнилась, вернуться к предыдущей цели и выполнить следующий вариант ее реализации.
7. Если текущий вариант реализации процедуры P/n не выполнился, перейти к следующему варианту.
8. Если все цели из тела процедуры выполнены, закончить работу с результатом TRUE.

Унификация – это подстановка параметров в паттерн и применяется не только для селекции входных параметров, но и для автоматического выделения нужных элементов списка и манипулирования ими – добавления и удаления головного элемента списка.

Унификация–это процесс сопоставления вопроса с фактами и правилами базы знаний.

PREDICATES

t(integer,integer)

CLAUSES

t(1,2).

Для Пролога вопрос есть цель, которую необходимо достичь. Пролог берет вопрос t(X,Y) и начинает последователь-но сверху-вниз сравнивать его с фактами и правилами базы знаний. Там, где обнаруживается предикат с таким же идентификатором, как и у вопроса и с таким же количеством аргументов происходит сопоставление. Если в вопросе аргументы ничем не означены (то есть X и Y свободные переменные), то значения из найденного факта присваиваиваются соответствующим переменным. Зададим другой вопрос: t(1,Y).Получим ответ: Y=2.Процесс унификации проходил следующим образом. Пролог взял вопрос t(1,Y) и последовательно сверху-вниз сравнивая его с фактами из базы знаний обнаружил, что существует предикат с таким же идентификатором как и у во-проса и с таким же количеством аргументов. Так как в вопросе первый аргумент имеет конкретное значение, то Пролог проводит сопоставление сравнивая значение из вопроса и из факта. В данном случае они совпали, поэтому процесс сопоставления переходит на вторые аргументы. В вопросе второй аргумент ничем не означен, поэтому ему передается значение из факта.

В Прологе знак "=" не есть знак присвоения. Его действие зависит от контекста. Только если с одной стороны от знака равенства стоит свободная переменная (то есть не имеющая никакого значения), а с другой стороны расположено какое-либо значение или означенная переменная (имеющая конкретное значение), тогда действие знака "=" можно считать эквивалентным присвоению. По итогам его выполнения свободная переменная получит значение с противоположной стороны от знака равенства (сторона не имеет значение) и цель признается выполненной.

Итак, унификация–это встроенный базовый механизм Пролога, представляющий собой процесс последовательного сопоставления вопроса с фактами и правилами базы знаний с целью доказательства его реализуемости. В ходе сопоставления проверяется совпадение следующих сущностей: идентификаторы предикатов, количество аргументов, их значения. Если в вопросе есть свободные переменные, то Пролог ищет значения переменных, при которых цель достижима. Если в вопросе нет свободных переменных, то Пролог ищет возможность унификации цели и в качестве ответа выдает результативность такой унификации.

# Практическая часть

**Если запустить интерпретатор и ввести**

(setf a 2)  
(setf b 3)  
(setf c 5)

Что будет на экране, если вводить:

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 2 |
| b | 3 |
| c | 5 |
| `a | A |
| `(+a c) | (+ A C) |
| (a) | Undefined function |
| (eval `a) | 2 |

**Добавив (defun a() ‘b) и (defun b() 4) Что будет на экране, если вводить:**

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 2 |
| b | 3 |
| c | 5 |
| `a | A |
| `(+a c) | (+ A C) |
| (a) | B |
| (eval `a) | 2 |
| (a) | B |
| (+ a a) | 4 |
| (+ (b) b) | 7 |
| (b b b) | invalid number of arguments |

**Добавив (setf a b) Что будет на экране, если вводить:**

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 3 |
| b | 3 |
| c | 5 |
| `a | A |
| `(+a c) | (+ A C) |
| (a) | B |
| (eval `a) | 3 |

**Добавив (setf c b) Что будет на экране, если вводить:**

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 3 |
| b | 3 |
| c | 3 |
| `a | A |
| `(+a c) | (+ A C) |
| (a) | B |
| (eval `a) | 3 |

**Добавив (defun a(x y) (+ x y)) Что будет на экране, если вводить:**

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 3 |
| b | 3 |
| c | 3 |
| `a | A |
| `(+a c) | (+ A C) |
| (a) | invalid number of arguments: 0 |
| (eval `a) | 3 |
| (a a a) | 6 |
| (a b a) | 6 |

**Добавив (defun b(x y) (setf b (+ b 1)) (\* x y b) ) Что будет на экране, если вводить:**

|  |  |
| --- | --- |
| **Команда** | **Результат** |
| a | 3 |
| b | 3 |
| (b b b) | 36 |

**Напишите функцию, которая вычисляет катет по гипотенузе и другому катету.**

(defun catet (cat hyp) (sqrt(- (\* hyp hyp) (\* cat cat))))