|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 18**

|  |  |
| --- | --- |
| **Тема** Prolog  **Студент** Белоусова Ю.С.  **Группа** ИУ7-61Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Толпинская Н.Б. |  |

Москва.

2020 г.

### Задание

Используя хвостовую рекурсию, разработать программу, позволяющую найти 1. n!,

2. n-е число Фибоначчи. Убедиться в правильности результатов.

### Ответы на вопросы

1. **Что такое рекурсия? Как организуется хвостовая рекурсия в Prolog? Как организовать выход из рекурсии в Prolog?**

Рекурсия – один из способов организации повторых вычислений. В логическом программировании – способ заставить систему многократно использовать одну и ту же процедуру. При этом из нее должен быть выход.

Организация хвостовой рекурсии:

* + Рекурсивный вызов единственен и расположен в конце тела правила.
  + До вычисления рекурсивного вызова не должно быть возможности сделать откат (т.е точки отката отсутствуют). Этого можно добиться, например, с помощью предиката отсечения.

Для выхода из рекурсии используется отдельное правило, в конце которого будет находиться предикат отсечения.

### Какое первое состояние резольвенты?

На первом шаге в резольвенте находится заданный вопрос (цель).

### В каком случае система запускает алгоритм унификации? Каково назначение использования алгоритма унификации? Каков результат работы алгоритма унификации?

Система запускает алгоритм унификации, если есть еще на какие вопросы отвечать (какие цели доказывать). То есть если резольвента (совокупность целей) не пуста.

Назначение алгоритма унификации - подобрать знание, которое позволит на поставленный  
вопрос ответить да.  
Результатом алгоритма унификации является «успех» (цель доказана) или  
«неудача» (на основе данной базы знаний невозможно доказать цель). В случае «успеха» в  
качестве побочного эффекта формируется подстановка, содержащая значения  
переменных, при которых вопрос станет примером программы.

### В каких пределах программы переменные уникальны?

Именованная переменная уникальна в рамках предложения, в котором она используются. Любая анонимная переменная является уникальной.

### Как применяется подстановка, полученная с помощью алгоритма унификации?

Полученная с помощью алгоритма унификации подстановка применяется к целям в резольвенте.

Подстановкой называется множество пар, вида: { 𝑋𝑖= 𝑡𝑖 } , где 𝑋𝑖 – переменная, а 𝑡𝑖 – терм. Т.е происходит конкретизация переменной термом. Применение подстановки заключается в замене каждого вхождения переменной 𝑋𝑖 на соответствующий терм (𝑡𝑖). В результате применения подстановки переменные конкретизируются значениями, которые будут далее использованы при доказательстве истинности тела выбранного правила то есть значения переменных переходят на следующих шаг доказательства.

### Как изменяется резольвента?

Для хранения резольвенты используется стек. Меняется она в ходе доказательства. Преобразования выполняются с помощью редукции – замены текущей цели на тело найденного в программе правила (с помощью унификации цели и заголовка правила программы).

Преобразование резольвенты разделено на два этапа:

1. Берется верхняя из подцелей резольвенты (по стековому принципу) и заменяется на тело правила, найденного в программе.
2. Затем к полученной конъюнкции целей применяется подстановка (наибольший общий унификатор цели и сопоставленного с ней правила).

### В каких случаях запускается механизм отката?

Во время работы системы, в случае, если решение не найдено, и из данного состояния невозможен переход в новое состояние (тупиковое состояние), применяется механизм отката. Также для поиска альтернативных решений (резольвента пуста, но не все правила были рассмотрены).

### domains

int = integer

### predicates

fact(int, int, int, int) factorial(int, int)

fibonacci(int, int) fib(int, int, int, int)

### clauses

**/\*** Обертка **\*/**

factorial(N,Res) :- fact(N, Res,1,1).

/\*Условие выхода \*/ fact(N,Res,N, Res) :- !.

/\*Основной шаг процедуры\*/

fact(N, Res,TmpN,TmpF) :- NewN = TmpN + 1, NewF = TmpF \* NewN, fact(N,Res,NewN,NewF).

**/\*** Проверка на корректный запрос и обертка **\*/**

fibonacci(N, Res):- N < 1, !, fail.

fibonacci(N, Res):- fib(N, 1, 0, Res).

/\*Условие выхода \*/ fib(1, Res, \_, Res) :- !.

/\*Основной шаг процедуры\*/

fib(N, Prev1, Prev2, Res) :- NewN = N - 1,

NewPrev2 = Prev1, NewPrev1 = Prev1 + Prev2,

fib(NewN, NewPrev1, NewPrev2, Res).

### Описание параметров программы

* + Factorial(**N, Res**)
    - N – аргумент, факториал которого нужно найти
    - Res – аргумент-результат.
  + fact(**N, Res,TmpN,TmpF**)
    - N – аргумент, факториал которого нужно найти
    - Res – аргумент-результат.
    - TmpN – счетчик, принимающий значения от 1 до N. Нужен для выхода из рекурсии и для вычисления факториала.
    - TmpF – аргумент для хранения накапливаемого значения факториала.
  + fibonacci(**N, Res**)
    - N – аргумент-номер числа фибоначчи, которое нужно вычислить
    - Res – аргумент-результат.

### fib(N, Prev1, Prev2, Res)

* + - N – аргумент-номер числа фибоначчи, которое нужно вычислить
    - Res – аргумент-результат
    - Prev1, Prev2 – аргументы, соответсвующие значениям чисел фибоначчи на двух предыдуших шагах. (нужно для вычисления).

### Примеры целей и результатов работы программы

1. **Goal** factorial(3, Res).

### Result Res=6

1. **Goal** fibonacci(3, Res).

### Result Res=2

1. **Goal** fibonacci(3, 2).

### Result yes

1. **Goal** factorial(3, 6).

### Result yes

**Описание порядка поиска объектов**

**Цель:** factorial(2, Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты, и вывод: дальнейшие действия (почему?) | Для каких термов запускается алгоритм унификации: Т1=Т2 и каков **результат** (и подстановка) | Дальнейшие действия: прямой ход или откат (почему и к чему приводит?) |
| 1 | factorial(2, Res)  В качестве первого состояния в резольвенту помещается вопрос | Попытка унификации factorial(2, Res) = factorial(N,Res)  Результат: успех  Подстановка: {N=2, Res=Res} | Прямой ход.  Замена цели телом правила, применение подстановки. |
| 2 | fact(2, Res,1,1) | Попытка унификации:  fact(2, Res,1,1) = factorial(N,Res)  Результат: неудача  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 3 | fact(2, Res,1,1) | Попытка унификации: fact(2, Res,1,1) = fact(N,Res,N, Res) Результат: неудача  Разные константы. | Прямой ход, переход к следующему предложению в БЗ. |
| 4 | fact(2, Res,1,1) | Попытка унификации: fact(2,Res,1,1) = fact(N, Res,TmpN,TmpF)  Результат: успех Подстановка: {N=2, Res=Res,  TmpN=1, TmpF=1 } | Прямой ход.  Замена цели телом правила, применение подстановки. |
| 5 | NewN = 1 + 1,  NewF = 1 \* NewN, fact(2,Res,NewN,NewF). | Попытка унификации: NewN = 1 + 1  Результат: успех Подстановка: { NewN =2} | Прямой ход, преобразование резольвенты применением подстановки |
| 6 | NewF = 1 \* 2,  fact(2,Res, 2, NewF). | Попытка унификации:  NewF = 1 \* 2  Результат: успех Подстановка {NewF=2} | Прямой ход, преобразование резольвенты применением подстановки |
| 7 | fact(2, Res, 2, 2). | Попытка унификации: fact(2, Res, 2, 2) = factorial(N,Res)  Результат: неудача  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 8 | fact(2, Res, 2, 2). | Попытка унификации:  fact(2, Res, 2, 2) = fact(N,Res,N, Res) Результат: успех  Подстановка: {N=2, Res=Res, N=2, Res=2} | Прямой ход, замена цели телом правила, применение подстановки. |
| 9 | !. | Выполнение отсечения, запрет отката | Прямой ход. |
| 10 | Резольвента пустая | Завершение работы | Вывод результата  Res = 2 |

**Цель:** fibonacci(2, Res).

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Состояние резольвенты, и вывод: дальнейшие действия (почему?) | Для каких термов запускается алгоритм унификации: Т1=Т2 и каков **результат** (и подстановка) | Дальнейшие действия: прямой ход или откат (почему и к чему приводит?) |
| 1 | fibonacci(2, Res)  В качестве первого состояния в резольвенту помещается вопрос | Попытка унификации: fibonacci(2, Res) = factorial(N,F)  Результат: неудача  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 2 | fibonacci(2, Res) | Попытка унификации: fibonacci(2, Res) = fact(N,F,N,F)  Результат: неудача  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 3 | fibonacci(2, Res) | Попытка унификации: fibonacci(2, Res) = fact(N,F,TmpN,TmpF)  Результат: неудача  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 4 | fibonacci(2, Res) | Попытка унификации: fibonacci(2, Res) = fibonacci(N, \_) Результат: Успех Подстановка: {N=2} | Прямой ход, замена цели телом правила, применение подстановки. |
| 5 | 2 < 1,  !,  fail. | 2 < 1 - ложь | Откат |
| 6 | fibonacci(2, Res) | Попытка унификации: fibonacci(2, Res) = fibonacci(N, Res) Результат: успех  Подстановка: {N=2, Res=Res} | Прямой ход, замена цели телом правила, применение подстановки. |
| 7 | fib(2, 1, 0, Res) | Неуспешные унификации  fib(2, 1, 0, Res) с каждым из следующих термов  factorial(N,Res) fact(N,Res,N, Res)  fact(N, Res,TmpN,TmpF)  fibonacci(N, Res)  fibonacci(N, Res)  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 8 | fib(2, 1, 0, Res) | Попытка унификации: fib(2, 1, 0, Res) =  fib(1, Res, \_, Res)  Результат: неудача.  Разные константы. | Прямой ход, переход к следующему предложению в БЗ. |
| 9 | fib(2, 1, 0, Res) | Попытка унификации: fib(2, 1, 0, Res) =  fib(N, Prev1, Prev2, Res) Результат: успех Подстановка: {N=2, Prev1=1, Prev2=0, Res=Res} | Прямой ход, замена цели телом правила, применение подстановки. |
| 10 | NewN = 2 - 1,  NewPrev2 = 1,  NewPrev1 = 1 + 0, fib(NewN, NewPrev1, NewPrev2, Res). | Попытка унификации: NewN = 2 – 1  Результат: успех Подстановка: {NewN=1} | Прямой ход, преобразование резольвенты применением подстановки. |
| 11 | NewPrev2 = 1,  NewPrev1 = 1 + 0,  fib(1, NewPrev1, NewPrev2, Res). | Попытка унификации: NewPrev2 = 1 Результат: успех  Подстановка: {NewPrev2 = 1  } | Прямой ход, преобразование резольвенты применением подстановки |
| 12 | NewPrev1 = 1 + 0,  fib(1, NewPrev1, 1, Res) | Попытка унификации: NewPrev1 = 1 + 0 Результат: успех  Подстановка: {NewPrev1 = 1  } | Прямой ход, преобразование резольвенты применением подстановки. |
| 13 | fib(1,1, 1, Res) | Неуспешные унификации  fib(1,1, 1, Res) с каждым из следующих термов  factorial(N,Res) fact(N,Res,N, Res)  fact(N, Res,TmpN,TmpF)  fibonacci(N, Res)  fibonacci(N, Res)  Функторы не совпадают | Прямой ход, переход к следующему предложению в БЗ. |
| 14 | fib(1, 1, 1, Res) | Попытка унификации:  fib(1, 1, 1, Res) =  fib(1, Res, \_, Res) Результат: успех  подстановка: {Res=1, Res=Res=1} | Прямой ход, замена цели телом правила, применение подстановки. |
| 15 | ! | Выполнение отсечения, запрет отката | Прямой ход. |
| 16 | Резольвента пуста | Завершение работы | Вывод результата  Res = 1 |

# Выводы

Для повышения эффективности программы на пролог, можно использовать отсечения, чтобы ограничить количество вычислений, в случае, если они избыточны (например, как в случае с взаимоисключающими правилами), также можно использовать хвостовую рекурсию, ее главное отличие от стандартной реализации рекурсии в том, что при вычислениях, не возникает истощения памяти.