Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 18

Формирование эффективных программ на Prolog

Дисциплина	Функциональное и логическое программирование
Студент	Сиденко А.Г.
Группа	ИУ7-63Б
Преподаватель	Толпинская Н.Б., Строганов Ю.В.

Москва, 2020 г.

Задание

Используя хвостовую рекурсию, разработать программу, позволяющую найти

1. n!
2. n-е число Фибоначчи

Убедиться в правильности результатов.

Для одного из вариантов вопроса и каждого задания составить таблицу, отражающую конкретный порядок работы системы

Программа

```
predicates
  factorial(integer, integer).
  fibonachi(integer, integer).
clauses
  factorial(0, 1):- !.
  factorial(Next, FactorialNext):-
    Cur = Next - 1,
    factorial(Cur, FactorialCur),
    FactorialNext = FactorialCur * Next.

  fibonachi(0, 0):- !.
  fibonachi(1, 1):- !.
  fibonachi(Next, FibonachiNext):-
    Cur = Next - 1,
    Prev = Cur - 1,
    fibonachi(Cur, FibonachiCur),
    fibonachi(Prev, FibonachiPrev),
    FibonachiNext = FibonachiCur + FibonachiPrev.
```

Приведем таблицу для задания 1.

```
goal
  factorial(5, Rez).
```

№ шага	Состояние резольвенты	Сравниваемые термы; результат; подстановка, если есть	Дальнейшие действия: прямой ход или откат
1	factorial(5, Rez)	По factorial(5, Rez) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(5, Rez), прямой ход

2	Cur = 5 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 5	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо Cur = Next - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * Next	Прямой ход
3	Cur = 5 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 5	Cur = 5 - 1	Значение утверждения true, Cur = 4, переход к следующему определению
4	factorial(4, FactorialCur), FactorialNext = FactorialCur * 5	По factorial(4, FactorialCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(4, FactorialCur), прямой ход
5	Cur = 4 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	Cur = 4 - 1	Значение утверждения true, Cur = 3, переход к следующему определению
6	factorial(3, FactorialCur), FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	По factorial(3, FactorialCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(3, FactorialCur), прямой ход
7	Cur = 3 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	Cur = 4 - 1	Значение утверждения true, Cur = 2, переход к следующему определению

8	factorial(2, FactorialCur), FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	По factorial(2, FactorialCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(2, FactorialCur), прямой ход
9	Cur = 2 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	Cur = 4 - 1	Значение утверждения true, Cur = 1, переход к следующему определению
10	factorial(1, FactorialCur), FactorialNext = FactorialCur * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	По factorial(1, FactorialCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(1, FactorialCur), прямой ход
11	Cur = 1 - 1, factorial(Cur, FactorialCur), FactorialNext = FactorialCur * 1, FactorialNext = FactorialCur * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	Cur = 4 - 1	Значение утверждения true, Cur = 0, переход к следующему определению

12	factorial(0, FactorialCur), FactorialNext = FactorialCur * 1, FactorialNext = FactorialCur * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	По factorial(0, FactorialCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек factorial(0, FactorialCur), прямой ход
13	FactorialNext = 1 * 1, FactorialNext = FactorialCur * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	Унификация factorial(0, FactorialCur) и factorial(0, 1)	FactorialCur = 1, извлечение из стека factorial(0, FactorialCur)
14	FactorialNext = 1 * 2, FactorialNext = FactorialCur * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	FactorialNext = 1	Извлечение из стека factorial(1, FactorialCur)
15	FactorialNext = 2 * 3, FactorialNext = FactorialCur * 4, FactorialNext = FactorialCur * 5	FactorialNext = 2	Извлечение из стека factorial(2, FactorialCur)
16	FactorialNext = 6 * 4, FactorialNext = FactorialCur * 5	FactorialNext = 6	Извлечение из стека factorial(3, FactorialCur)
17	FactorialNext = 24 * 5	FactorialNext = 24	Извлечение из стека factorial(4, FactorialCur)
18		FactorialNext = 120	Резольвента пуста, вывод результата
19			Стек пуст, завершение программы

Приведем таблицу для задания 2.

```
1 goal
2   fibonachi(4, Rez).
```

№ шага	Состояние резольвенты	Сравниваемые термы; результат; подстановка, если есть	Дальнейшие действия: прямой ход или откат
1	fibonachi(4, Rez)	По fibonachi(4, Rez) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(4, Rez), прямой ход
2	Cur = 4 - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо Cur = Next - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Прямой ход
3	Prev = 3 - 1, fibonachi(3, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Cur = 4 - 1	Значение утверждения true, Cur = 3, переход к следующему определению
4	fibonachi(3, FibonachiCur), fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Prev = 3 - 1	Значение утверждения true, Prev = 2, переход к следующему определению

5	fibonachi(3, FibonachiCur), fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	По fibonachi(3, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(3, FibonachiCur), прямой ход
6	Cur = 3 - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо Cur = Next - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Прямой ход
7	Prev = 2 - 1, fibonachi(2, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Cur = 3 - 1	Значение утверждения true, Cur = 2, переход к следующему определению

8	fibonachi(2, FibonachiCur), fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Prev = 2 - 1	Значение утверждения true, Prev = 1, переход к следующему определению
9	fibonachi(2, FibonachiCur), fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	По fibonachi(2, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(2, FibonachiCur), прямой ход
10	Cur = 2 - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо Cur = Next - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Прямой ход

11	Prev = 1 - 1, fibonachi(1, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Cur = 2 - 1	Значение утверждения true, Cur = 1, переход к следующему определению
12	fibonachi(1, FibonachiCur), fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Prev = 1 - 1	Значение утверждения true, Prev = 0, переход к следующему определению

13	fibonachi(1, FibonachiCur), fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	По fibonachi(1, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(1, FibonachiCur), прямой ход
14	fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Унификация fibonachi(1, FibonachiCur) и fibonachi(1, 1)	FibonachiCur = 1, извлекается из стека fibonachi(1, FibonachiCur), прямой ход

15	fibonachi(0, FibonachiPrev), FibonachiNext = 1 + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	По fibonachi(0, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(0, FibonachiCur), прямой ход
16	FibonachiNext = 1 + FibonachiPrev, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Унификация fibonachi(0, FibonachiCur) и fibonachi(0, 0)	FibonachiPrev = 0, извлекается из стека fibonachi(0, FibonachiCur), прямой ход
17	FibonachiNext = 1 + 0, fibonachi(1, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	FibonachiNext = 1 + 0	FibonachiCur = 1, прямой ход

18	fibonachi(1, FibonachiPrev), FibonachiNext = 1 + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	По fibonachi(1, FibonachiPrev) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(1, FibonachiPrev), прямой ход
19	FibonachiNext = 1 + FibonachiPrev, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Унификация fibonachi(1, FibonachiPrev) и fibonachi(1, 1)	FibonachiPrev = 1, извлекается из стека fibonachi(1, FibonachiCur), прямой ход
20	FibonachiNext = 1 + 1, fibonachi(2, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	FibonachiNext = 1 + 1	FibonachiCur = 2, прямой ход
21	fibonachi(2, FibonachiPrev), FibonachiNext = 2 + FibonachiPrev	По fibonachi(2, FibonachiPrev) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(2, FibonachiPrev), прямой ход
22	Cur = 2 - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	Начинает «раскрываться» правило, т.е. доказывается каждое целевое утверждение в теле правила последовательно слева направо Cur = Next - 1, Prev = Cur - 1, fibonachi(Cur, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev	Прямой ход

23	Prev = 1 - 1, fibonachi(1, FibonachiCur), fibonachi(Prev, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	Cur = 2 - 1	Значение утверждения true, Cur = 1, переход к следующему определению
24	fibonachi(1, FibonachiCur), fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	Prev = 1 - 1	Значение утверждения true, Prev = 0, переход к следующему определению
25	fibonachi(1, FibonachiCur), fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	По fibonachi(1, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(1, FibonachiCur), прямой ход
26	fibonachi(0, FibonachiPrev), FibonachiNext = FibonachiCur + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	Унификация fibonachi(1, FibonachiCur) и fibonachi(1, 1)	FibonachiCur = 1, извлекается из стека fibonachi(1, FibonachiCur), прямой ход
27	fibonachi(0, FibonachiPrev), FibonachiNext = 1 + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	По fibonachi(0, FibonachiCur) ищется системой определение отношения (по имени предиката и списку (числу) аргументов)	Определение отношения найдено, заносится в стек fibonachi(0, FibonachiCur), прямой ход

28	FibonachiNext = 1 + FibonachiPrev, FibonachiNext = 2 + FibonachiPrev	Унификация fibonachi(0, FibonachiCur) и fibonachi(0, 0)	FibonachiPrev = 0, извлекается из стека fibonachi(0, FibonachiCur), прямой ход
29	FibonachiNext = 1 + 0, FibonachiNext = 2 + FibonachiPrev	FibonachiNext = 1 + 0	FibonachiPrev = 1, извлекается из стека fibonachi(3, FibonachiCur), прямой ход
30		FibonachiNext = 2 + 1	FibonachiNext = 3, извлекается из стека fibonachi(4, FibonachiCur), Резольвента пуста, вывод результата
31			Стек пуст, завершение программы

Вывод

Эффективный способ организации рекурсии – хвостовая рекурсия. Эффективность рекурсивной процедуры повышается благодаря отсечению неперспективных путей поиска решения. Используя «!» – отсечение. Которое сократит количество выполняемых унификаций для достижения максимальной эффективности работы системы.

Ответы на вопросы

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

   Рекурсия позволяет использовать в процессе определения предиката его самого.

   Хвостовая рекурсия: Для ее осуществления рекурсивный вызов определяемого предиката должен быть последней подцелью в теле рекурсивного правила и к моменту рекурсивного вызова не должно остаться точек возврата (непроверенных альтернатив).

   Параметры должны изменяться на каждом шаге так, чтобы в итоге либо сработал базис рекурсии, либо условие выхода из рекурсии, размещенное в самом правиле.

2. Какое первое состояние резольвенты?

   Вопрос.

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

   Пролог выполняет унификацию в двух случаях: когда цель сопоставляется с заголовком предложения или когда используется знак равенства, который является инфиксным предикатом (предикатом, который расположен между своими аргументами, а не перед ними).

   Унификация двух термов – это основной шаг доказательства. В процессе работы система выполняет большое число унификаций. **Унификация** – операция, которая позволяет формализовать процесс логического вывода.

   Унификация представляет собой процесс сопоставления цели с фактами и правилами базы знаний. Цель может быть согласована, если она может быть сопоставлена с заголовком какого-либо предложения базы.

   Результатом его работы является последняя из построенных подстановок.

4. В каких пределах программы уникальны переменные?

   Областью действия переменной в Прологе является одно предложение. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов. Исключением является анонимная переменная. Каждая анонимная переменная – это отдельный объект.

5. Как применяется подстановка, полученная с помощью алгоритма унификации?

   При согласовании переменные получают значения, указанные с другой стороны от знака «=», если переменные еще не были связаны. Переменные становятся связанными и после успешного согласования всех целевых утверждений, будет напечатано значение связанных переменных.

6. Как изменяется резольвента?

   Резольвента - текущая цель, существующая на любой стадии вычислений. Резольвенты порождаются целью и каким-либо правилом или фактом, которые просматриваются последовательно сверху вниз. Если резольвента существует при наиболее общей унификации, она вычисляется. Если пустая резольвента с помощью такой стратегии не найдена, то ответ на вопрос отрицателен.

7. В каких случаях запускается механизм отката?

   Откат дает возможность получить много решений в одном вопросе к программе.

   Во всех точках программы, где существуют альтернативы, в стек заносятся точки возврата.

Если впоследствии окажется, что выбранный вариант не приводит к успеху, то осуществляется откат к последней из имеющихся в стеке точек программы, где был выбран один из альтернативных вариантов.

Выбирается очередной вариант, программа продолжает свою работу. Если все варианты в точке уже были использованы, то регистрируется неудачное завершение и осуществляется переход на предыдущую точку возврата, если такая есть.

При откате все связанные переменные, которые были означены после этой точки, опять освобождаются.