Логическое программирование

Занятие 2 / 26.04.22

Посмотрим на домашнюю задачу

Лабораторная работа 1

Все лабораторные работы выполняются по предметной области, определяемой номером студенческого билета

Вариант = последняя цифра номера студенческого билета

Лабораторная работа – творческое задание! Необычные факты, правила и запросы приветствуются! В таблице информация приведена только для справки.

Лабораторная работа 1. Критерии оценки

Работа выполняется в PIE

Работа выполняется по вашей предметной области

В программе реализовано не менее 10 фактов

В программе реализовано не менее 3 правил

Результат – файл .PRO, который отправляется в ТУИС

Утверждения в Prolog

Утверждения:

Факты

<имя отношения>(<список аргументов>)

Правила

<заголовок-предикат> :- <тело>

<имя отношения>(<список аргументов>) :- <предикат 1>, <предикат 2>, …, <предикат N>.

Стандартные типы

Symbol

String

Char

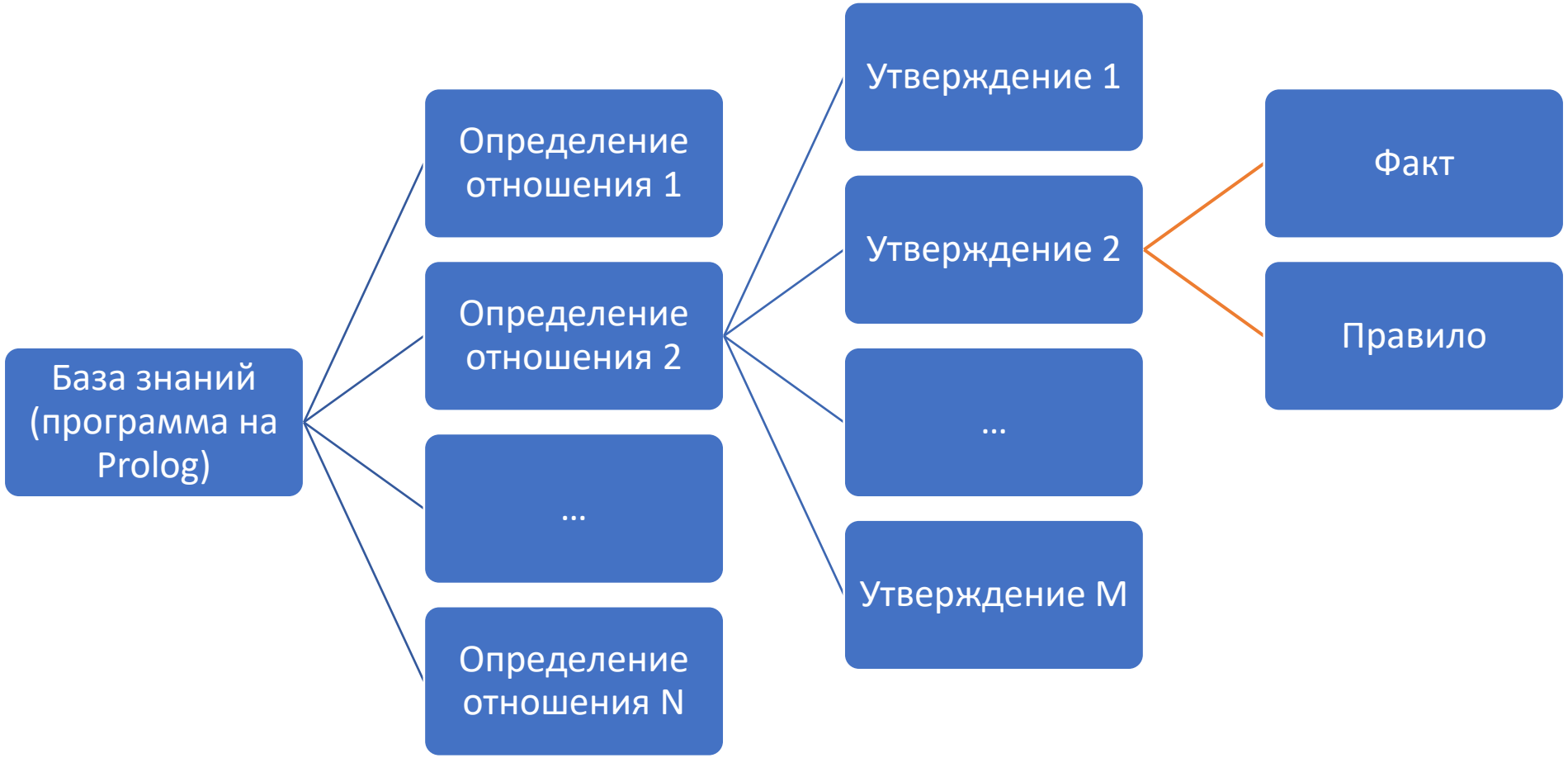
Integer

Byte

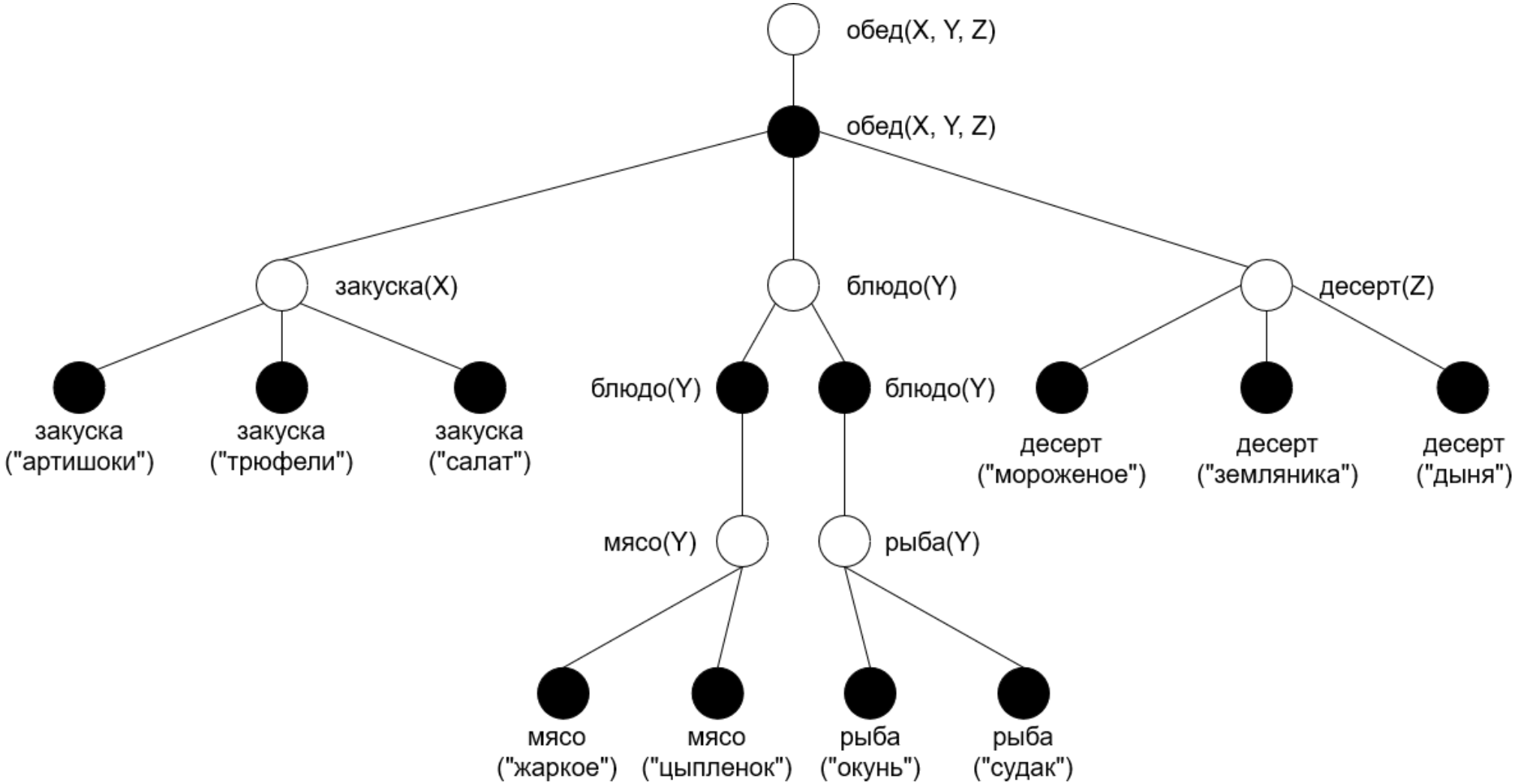
Word

Real

Структура раздела утверждений программы



И-ИЛИ дерево


обед(X, Y, Z)
обед(X, Y, Z)
закуска(X)
блюдо(Y)
десерт(Z)
блюдо(Y)
блюдо(Y)
закуска ("артишоки")
закуска ("трюфели")
закуска ("салат")
десерт ("мороженое")
десерт ("земляника")
десерт ("дыня")
мясо(Y)
рыба(Y)
мясо ("жаркое")
мясо ("цыпленок")
рыба ("окунь")
рыба ("судак")

Алгоритм работы интерпретатора

1. Сопоставление с образцом
2. Унификация аргументов
3. Проверка: факт или правило
4. Процесс возврата

Унификация аргументов

Сопоставление аргументов целевого утверждения (среди которых могут быть переменные) с аргументами отношения базы знаний

Переменные, до момента присваивания им значений, имеют ***неконкретизированное значение*** (ссылается на адрес в памяти, в котором пока ничего не хранится)

Утверждения базы знаний рассматриваются в том порядке, в котором они встречаются в базе знаний

Унификация аргументов

Аргумент предиката X	Аргумент цели Y	Условия унификации
Константа, или конкретизированная переменная	Константа, или конкретизированная переменная	Только если значения одинаковые
Константа, или конкретизированная переменная	Неконкретизированная переменная	Y принимает значение X
Неконкретизированная переменная	Константа, или конкретизированная переменная	X принимает значение Y
Неконкретизированная переменная	Неконкретизированная переменная	Унификация успешна; переменные связываются

Унификация аргументов

джек(«личность») $\leftrightarrow$ джек(«человек»)

человек(«Джек») $\leftrightarrow$ человек(«Джек»)

человек(X) $\leftrightarrow$ человек(«Джек»)

человек(X) $\leftrightarrow$ человек(Y)

размер(X, X) $\leftrightarrow$ размер(23, 23)

размер(X, Y) $\leftrightarrow$ размер(23, 23)

размер(X, 12) $\leftrightarrow$ размер(23, Y)

размер(X, X, 23) $\leftrightarrow$ размер(Y, 18, Y)

Процесс возврата

Инициируется в случаях:

- Текущая цель оказалась ложной: унификация аргументов закончилась неудачей, так как не нашлось ни одного утверждения в определении, аргументы которого были бы сопоставимы с аргументами цели
- Программист намеренно создает процесс возврата (используя встроенный предикат `fail`)

Арифметические выражения в Prolog

+

-

*

/

div

mod

Предикаты сравнения значений в Prolog

X = Y	X унифицируется с Y
X <> Y	X не унифицируется с Y
X >< Y	X не унифицируется с Y
X > Y	X больше Y
X >= Y	X больше или равен Y
X < Y	X меньше Y
X <= Y	X меньше или равен Y

Управление выполнением программы. Цепочка

Конъюнкция предикатов в теле правила может рассматриваться как последовательность вызовов логических функций с параметрами – аргументами предикатов

Управление выполнением программы. Выбор среди альтернатив (if-else)

Несколько утверждений в определении могут рассматриваться как альтернативные варианты

В теле каждого утверждения ставится условие его выполнения

```
function(Value) :- Value < 20, write(“less than 20”).
function(Value) :- Value >= 20, write(“greater than or equal to 20”).
```

Управление выполнением программы. Цикл. Предикат `fail`

`fail` используется для инициирования процесса возврата и перебора возможных решений, получаемых в И-ИЛИ дереве

```
person(4, "Maggy", female).
person(3, "Matt", male).
…
print() :-
 person(_, Person, _), write(Person + "; "), fail.

[in :] print()
[out:] Maggy; Matt; Tom; Pam; Julia; Daemon; Tony;
```