Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

ОТЧЁТ
по лабораторной работе №2
по дисциплине

ЛОГИЧЕСКИЕ ОСНОВЫ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Студент гр. 121701 Р. В. Липский
Руководитель В. П. Ивашенко

Минск 2023

СОДЕРЖАНИЕ

1 ХОД ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Вариант 8

Тема: обратный нечёткий логический вывод.

Цель: ознакомиться и получить навыки реализации методов обратного нечёткого логического вывода.

Задание: Реализовать систему обратного нечёткого логического вывода на основе операции нечёткой композиции $max(\{max\{0\} \cup \{x_i + y_i - 1\})|i\})$

1.1 Описание

Задача заключается в написании алгоритма обратного нечёткого вывода, используя нечёткую композицию $max(\{max\{0\} \cup \{x_i + y_i - 1\})|i\})$. Входом программы является файл, содержащий два нечётких предиката – правило и заключение. Выходом программы должен является файл, содержащий множество посылок, которые, при применении данного правила, должны привести к указанному заключению. Стоит заметить, что задача обратного нечёткого вывода сложнее, чем задача прямого нечёткого вывода и не всегда имеет решение).

Для реализация программы использовался язык программирования C++.

Были использованы следующие структуры данных:

– Список

– Множество

– Словарь

– Кортеж

Структура приложения выглядит следующим образом:

– src – каталог, содержащий исходный код программы:

– src/fuzzy_set_types.h и src/fuzzy_set_types.cpp – файлы, содержащие описание предметной области программы;

– src/main.cpp – точка входа программы, содержащая алгоритм решения поставленной задачи.

1.2 Теоретические сведения

Правило – импликация, которая выражает зависимость между наблюдаемыми причинами и следствиями.

Прямой нечеткий логический вывод – композиция между двумя нечеткими предикатами, один из которых рассматривается как унарный (посылка), а второй бинарный (импликация фактов по заданному правилу).

Обратный нечеткий логический вывод – обратное по отношению к прямому нечёткому логическому выводу действие.

Нечеткое высказывание – утверждение, в котором истинность оценивается с использованием степени принадлежности к нечеткому множеству.

Нечеткий предикат – это нечеткое множество, значения которого интерпретируются как значения истинности.

Импликация – бинарная логическая связка, по своему применению приближенная к союзам «если..., то...».

Нечеткая импликация нечетких высказываний - это операция, которая определяет отношение между двумя нечеткими высказываниями.

1.3 Формат базы знаний

```
<база знаний>::= <заключение>  <правило>

<заключение>::= <имя нечёткого множества>=<нечёткое множество>

<правило>::= <имя нечёткого множества>=<матрица>

<матрица>::= (<степень принадлежности> <степень принадлежности>)

<нечёткое множество>::={ <список пар нечёткой принадлежности> }

<список пар нечёткой принадлежности>::=
<пара нечёткой принадлежности>, <пара нечёткой принадлежности>

<пара нечёткой принадлежности>::=(<элемент>,<степень принадлежности>)

<элемент>::=<имя> | <множество>

<множество>::=<ориентированное множество> |
              <неориентированное множество>

<неориентированное множество>::={ <список элементов> }

<ориентированное множество>::=(<элемент>,<список элементов>)

<список элементов>::=<элемент> ,<элемент> ,<элемент>

<имя нечёткого множества>::=<имя>

<имя>::=<символ> <символ>
```

```
<символ>::=<буква>|<цифра>

<цифра>::=0|...|9

<буква>::=A|...|z

<степень принадлежности>::=<действительное число с 0 по 1>

<действительное число с 0 по 1>::=<единица>|
<действительное число с 0 до 1>

<действительное число с 0 до 1>::=0 . <цифра>

<единица>::=1 . 0
```

1.4 Реализация

fuzzy_set_types.h

Name:

Класс `Name` представляет абстракцию имени и имеет следующие члены:

– `ch`: символ имени (тип `char`), инициализирован значением -1 по умолчанию.

– `num`: числовая часть имени (тип `int32_t`), инициализирована значением -1 по умолчанию.

Публичные методы:

– `Name()`: Конструктор по умолчанию.

– `explicit Name(const std::string &str)`: Конструктор, принимающий строку в качестве параметра.

– `std::string toString() const`: Метод, возвращающий строковое представление имени.

– `bool operator==(const Name &rhs) const`: Перегруженный оператор сравнения на равенство.

– `bool operator!=(const Name &rhs) const`: Перегруженный оператор сравнения на неравенство.

– `bool operator<(const Name &rhs) const`: Перегруженный оператор меньше.

SetName:

Класс `SetName` является производным от класса `Name` и используется для представления имени множества. Конструкторы и методы в этом классе аналогичны тем, что определены в классе `Name`.

RelationName:

Класс `RelationName` также является производным от класса `Name` и представляет имя отношения. Конструкторы и методы в этом классе аналогичны тем, что определены в классе `Name`.

SetElemName:

Класс `SetElemName` также является производным от класса `Name` и используется для представления имени элемента множества. Конструкторы и методы в этом классе аналогичны тем, что определены в классе `Name`.

FuzzySetElem:

Класс `FuzzySetElem` представляет элемент нечеткого множества и имеет следующие члены:

– `var`: имя элемента (тип `SetElemName`).

– `val`: значение элемента (тип `double`), инициализировано значением 0.0 по умолчанию.

Публичные методы:

– `FuzzySetElem()`: Конструктор по умолчанию.

– `explicit FuzzySetElem(const SetElemName &inVar, double inVal)`: Конструктор, принимающий имя элемента и значение в качестве параметров.

– `explicit FuzzySetElem(std::string str)`: Конструктор, принимающий строку в качестве параметра.

– `SetElemName getVar() const`: Метод, возвращающий имя элемента.

– `double getVal() const`: Метод, возвращающий значение элемента.

– `std::string toString() const`: Метод, возвращающий строковое представление элемента.

– `bool operator<(const FuzzySetElem &el) const`: Перегруженный оператор меньше.

FuzzySet:

Класс `FuzzySet` представляет нечеткое множество и является производным от `std::vector<FuzzySetElem>`. Конструкторы и методы в этом классе аналогичны тем, что определены в классе `FuzzySetElem`. Дополнительно класс `FuzzySet` имеет член `name` типа `SetName`, который представляет имя нечеткого множества.

Relation:

Класс `Relation` представляет отношение и имеет следующие члены:

– `matrix`: матрица отношения (тип `RelationMatrix`).

– `name`: имя отношения (тип `RelationName`).

Публичные методы:

– `Relation()`: Конструктор по умолчанию.

– `explicit Relation(const std::string &inStr)`: Конструктор, принимающий строку в качестве параметра.

– `const RelationMatrix &getMatrix() const`: Метод, возвращающий матрицу отношения.

– `const RelationName &getName() const`: Метод, возвращающий имя отношения.

ReverseConclusion:

Класс `ReverseConclusion` представляет обратный вывод и имеет следующие члены:

– `sols`: вектор решений (тип `Solutions`).

– `xSetName`: имя множества X (тип `SetName`).

– `ySetName`: имя множества Y (тип `SetName`).

– `relName`: имя отношения (тип `RelationName`).

Публичные методы:

– `ReverseConclusion()`: Конструктор по умолчанию.

– `explicit ReverseConclusion(std::string str)`: Конструктор, принимающий строку в качестве параметра.

– `std::string toString() const`: Метод, возвращающий строковое представление обратного вывода.

– `void setSolutions(Solutions inSols)`: Метод для установки вектора решений.

– `const SetName &getXSetName() const`: Метод, возвращающий имя множества X.

– `const SetName &getYSetName() const`: Метод, возвращающий имя множества Y.

– `const RelationName &getRelationName() const`: Метод, возвращающий имя отношения.

main.cpp

`Solutions calculateSolution(const FuzzySet &ySet, const RelationMatrix &rel);`

Функция, которая вычисляет решения на основе переданных нечетких множеств и матрицы отношений.

`bool addRange(Solutions &sols, size_t varNum, const Range &newRange);`

Функция, которая добавляет диапазон для указанного номера переменной в решения.

`bool addSolutions(Solutions &sols, const std::map<size_t, Range> &newRanges);`

Функция, которая добавляет новые диапазоны в решения.

`Solutions uniteSolutions(const Solutions &sols);`

Функция, которая объединяет решения в одно множество.

```
void read(std::ifstream &input, std::vector<FuzzySet>
&sets, std::vector<Relation> &rels,
  std::vector<ReverseConclusion> &concls);
```

Функция, которая считывает данные из входного файла и заполняет переданные векторы нечетких множеств,
матриц отношений и обратных выводов.

```
template <typename T>
std::vector<T> readSets(std::ifstream &input);
```

Функция-шаблон, которая считывает нечеткие множества из входного файла и возвращает вектор.

```
std::vector<Relation> readRelations(std::ifstream &input);
```

Функция, которая считывает матрицы отношений из входного файла и возвращает вектор.

```
template <typename T>
void validateNames(const std::vector<T> &vect);
```

Функция-шаблон, которая проверяет уникальность имен в переданном векторе.

```
template <typename T>
const T &findByName(const std::vector<T> &vect, const Name
&name);
```

Функция-шаблон, которая находит элемент по имени в переданном векторе и возвращает его ссылку.

1.5 Демонстрация результатов работы программы

1.5.1 Тест 1

Входной файл input1.kb:

```
C={(y1,0.1),(y2,0.3),(y3,0.2)}

n=(
0.1 0.3 0.2
0.1 0.3 0.2
0.1 0.3 0.2
0.1 0.3 0.2
)

{C1,n}=C
```

Рисунок 1.1 – Входной файл input1.kb для теста 1

Результат работы программы для файла input1.kb:

```
⟨A1(x1),A1(x2)⟩∈([1.0]×[1.0])
```

Рисунок 1.2 – Результат работы программы для файла input1.kb

1.5.2 Тест 2

Входной файл input2.kb:

```
B={(y1,0.0),(y2,0.0),(y3,0.0)}

l=(
0.7 1.0 1.0
0.7 0.9 0.7
1.0 0.8 0.2
0.1 0.6 0.4
0.1 0.2 0.3
)

{B1,l}=B
```

Рисунок 1.3 – Входной файл input2.kb для теста 2

Результат работы программы для файла input2.kb:

```
⟨B1(x1),B1(x2),B1(x3),B1(x4),B1(x5)⟩∈([0.0]×[0.0,0.1]×[0.0]×[0.0,0.4]×[0.0,0.7])
```

Рисунок 1.4 – Результат работы программы для файла input2.kb

1.5.3 Тест 3

Входной файл input3.kb:

```
C={(y1,0.1),(y2,0.3),(y3,0.2)}

n=(
0.1 0.3 0.2
0.1 0.3 0.2
0.1 0.3 0.2
0.1 0.3 0.2
)

{C1,n}=C
```

Рисунок 1.5 – Входной файл input3.kb для теста 3

Результат работы программы для файла input3.kb:

```
(LOIS) rlipski@rlipski-pc:~/opt/LOIS$ python src/main.py --kb example/input3.kb
I2 = B/~\(A~>B) = {<x4, 0.7>,<x1, 1>,<x3, 1>,<x2, 0.7>}
I3 = I2/~\(A~>B) = {<x2, 1>,<x1, 1>,<x4, 1>,<x3, 1>}
```

Рисунок 1.6 – Результат работы программы для файла input3.kb

1.6 Личный вклад

2 ВЫВОД

В процессе выполнения лабораторной работы, были получены навыки реализации нечёткой логики, а именно обратного нечёткого логического вывода при помощи программирования. В рамках данный работы были представлены модули, отвечающие за анализ исходного текста базы знаний, а так же алгоритм обрантого нечёткого логического вывода.

При помощи разработанного программного продукта нам удалось построить корректные выводы для нескольких случаев.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Голенков, В. В. Логические основы интеллектуальных систем. Практикум: учеб.-метод. пособие / В. В. Голенков. — БГУИР, 2011.