Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет

информатики и радиоэлектроники”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине «Логические основы интеллектуальных систем»

на тему

«Реализация процедур неклассического логического вывода и преобразований логических формул в базах знаний и онтологиях»

**Вариант 7**

Выполнил

студент гр. 121702 фио

Проверил Ивашенко В. П.

Минск 2023

**Цель:** Приобрести навыки разработки и программирования прикладных систем логического вывода, включая системы неклассического логического вывода.

**Задача:** Запрограммировать обратный нечеткий логический вывод на основе операции нечеткой композиции max(xi\*yi | i).

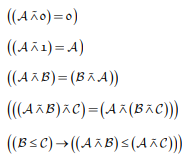
**Описание лабораторной работы:**

В ходе лабораторной работы необходимо произвести обратный нечеткий логический вывод, исходя из правила и посылки, введенных пользователем или загруженных из файла. Для этого необходимо:

1. Проверить введенные, либо загруженные из файла, нечеткое множество и правило на правильность синтаксиса.
2. С помощью полученных на вход нечеткого множества и правила составить систему уравнений.
3. На основе полученной системы уравнений рассчитать возможные значения степеней принадлежности.
4. Составить на основе возможных значений исходное нечеткое множество.[1]

**Дополнительные теоретические сведения:**

1. Под *обратным нечетким логическим выводом* понимается процесс, обратный процессу прямого логического вывода. В качестве исходных данных здесь выступают два нечётких предиката – правило и заключение. Требуется найти множество посылок, которые могут при применении данного правила привести к указанному заключению.[2]
2. *Нечеткое множество* определяется так: , где [0,1] является отрезком на множестве от числа ноль до числа один, элементы которого являются значениями степени нечеткой принадлежности, а S – произвольное множество.
3. *Нечеткий предикат* – это нечеткое множество, значения которого интерпретируются как значения истинности.
4. *Нечеткая конъюнкция* – операция конъюнкции над объектами нечеткой логики удовлетворяющая свойствам операции треугольной нормы.
5. *Треугольная норма*  – бинарная операция на интервале L=[0,1] с нейтральным элементом {1} удовлетворяющая ряду свойств:



# **Описание алгоритма программы:**

Программа включает в себя классы ParseUtils и Utils.

ParseUtils отвечает за чтение строк и преобразование их в структуры в памяти, в данном случае - нечеткие множества, а также за чтение из файла. Содержит методы tryParseToList() и readFromFile():

1. Метод tryParseToList() принимает на вход строку и, проверяя все символы строки с применением определенного набора правил, формирует из этой строки нечеткое множество, которое является результатом работы данного метода.
2. Метод readFromFile() принимает на вход путь к файлу и возвращает массив строк из этого файла.

Utils отвечает за алгоритмы расчета матрицы значений после применения импликации, а также за проведение прямого нечеткого логического вывода.

1. Метод makeEquations() создает систему уравнений для последующего решения.
2. Метод getPossibleSolutions() составляет множество возможных значений переменных в уравнении системы.
3. Метод getPossibleFixedIndices() подбирает все переменные, для которых возможно решение.
4. Метод getPossibleSystemSolutions() применяет getPossibleSolutions() для всех уравнений в системе.
5. Метод getCartesianProduct() составляет декартово произведение найденных возможных значений в уравнениях.
6. Метод uniteVariableSolutions() рассчитывает и накладывает ограничения на возможные значения переменных для уравнения.
7. Метод getSystemSolutions() применяет uniteVariableSolutions() для всех возможных найденных ранее уравнений.
8. Метод getVariableValues() находит все рассчитанные значения переменной.
9. Метод solveForVariable() рассчитывает ограниченные значения переменной исходя из найденных.
10. Метод tryIntersectIntervals() применяется для нахождения пересечения между двумя интервалами.
11. Метод filterSolutions() удаляет повторяющиеся и не влияющие на ответ решения.
12. Метод isSolutionOccluded() определяет, влияет ли решение на ответ.
13. Метод filterDuplicateSolutions() оставляет только уникальные уравнения.  
    Метод trySolveSystem() принимает на вход нечеткое множество и правило и пытается произвести нечёткий обратный логический вывод.

# 

# 

# **Тестирование программы:**

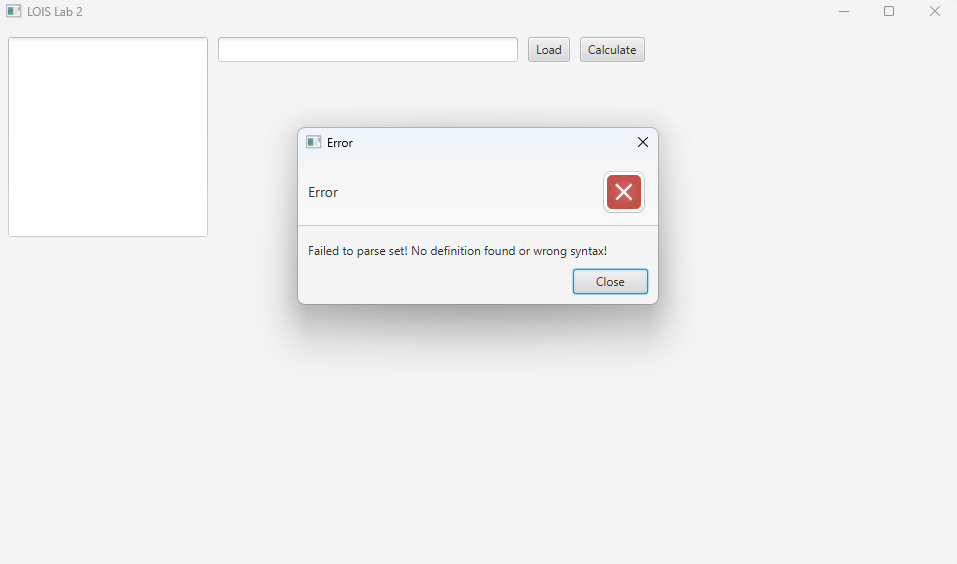


Рис. 2 – Тест 1. Пустой ввод

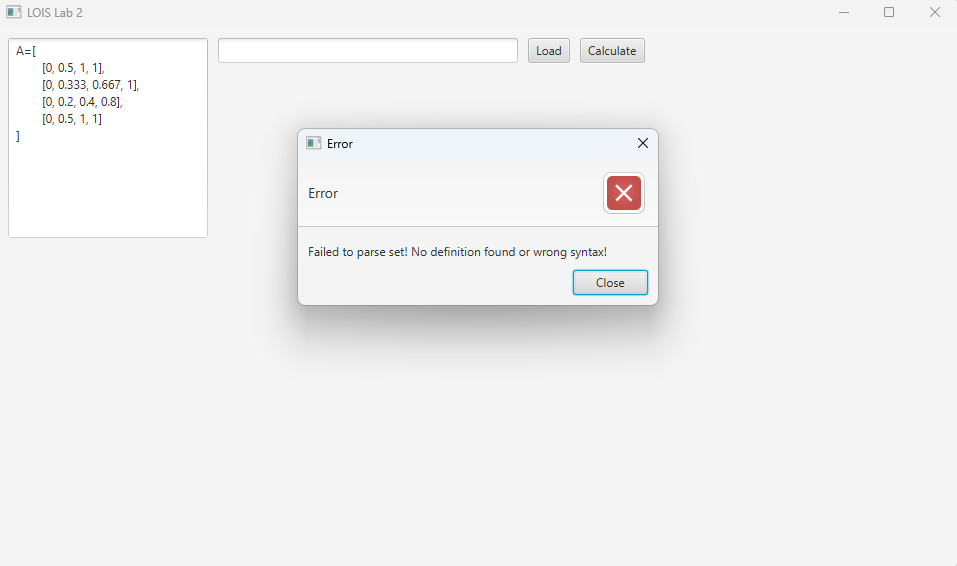


Рис. 3 – Тест 2. Частичный ввод



Рис. 4 – Тест 3. Успешное выполнение программы

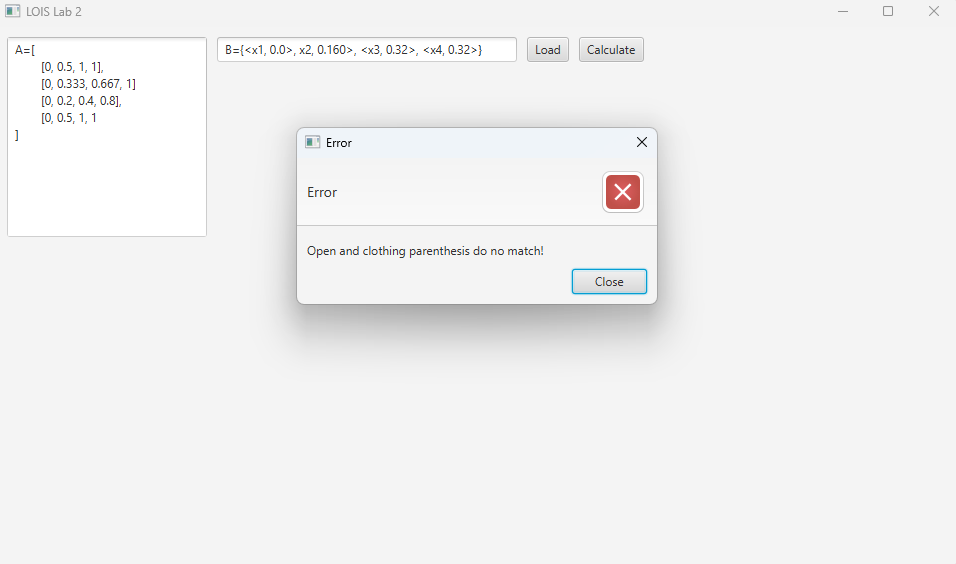


Рис. 5 – Тест 4. Неправильный синтаксис

**Вывод:**

# В процессе выполнения лабораторной работы были приобретены навыки разработки и программирования прикладных систем логического вывода, включая системы неклассического логического вывода. Это было достигнуто через создание и реализацию алгоритмов, описанных ранее. Была проведена отладка программы и осуществлено ручное тестирование окончательного результата.

**Персональный вклад членов команды:**

* Мойсевич А. В. – Реализация пользовательского интерфейса, составление отчета;
* Витковская С. И. – Реализация работы с файлами, составление отчета;
* Дьяков И. В. – Реализация логики обратного нечеткого логического вывода, тестирование системы.

**Список использованных источников:**

1. Труды ИСА РАН / А. А. Ахрем, М. Р. Ашинянц., С. А. Петров – Санкт-Петербург: ИСА РАН, 2007. – 265с.
2. Логические основы интеллектуальных систем. Практикум : учеб.-

метод. пособие / В. В. Голенков [и др.]. – Минск : БГУИР, 2011. – 70 с. : ил.

ISBN 978-985-488-487-5.