

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 №1
по дисциплине

**ЛОГИЧЕСКИЕ ОСНОВЫ ИНТЕЛЛЕКТУАЛЬНЫХ
СИСТЕМ**

Студент гр. 121701
Руководитель

Р. В. Липский
В. П. Ивашенко

Минск 2023

СОДЕРЖАНИЕ

1	Ход выполнения лабораторной работы	2
1.1	Описание	2
1.2	Теоретические сведения	3
1.3	Формат базы знаний	3
1.4	Реализация	4
1.5	Демонстрация результатов работы программы	5
1.5.1	Тест 1	5
1.5.2	Тест 2	6
1.5.3	Тест 3	7
1.6	Контрольные вопросы	7
1.7	Личный вклад	8
2	Вывод	9
	Список использованных источников	10

1 ХОД ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Вариант 21

Тема: Представление и обработка информации в условиях наличия не-факторов в рамках логических моделей

Цель: Приобрести навыки программирования алгоритмов обработки структур и формул в нечёткой логике

Задание: Реализовать систему прямого нечёткого логического вывода. Исходные данные и результаты должны быть сохранены в файле. Для расчёта меры возможности использовать импликацию Гёделя. Предусмотреть преодоление заикливания.

1.1 Описание

Задача заключается в написании алгоритма прямого нечеткого логического вывода, используя импликацию Гёделя. Входом программы является файл, содержащий множество нечётких правил и фактов.

Для реализации программы использовался язык программирования Python, а также генератор нисходящих анализаторов ANTLR.

Были использованы следующие структуры данных:

- Список
- Множество
- Словарь
- Кортеж

Структура приложения выглядит следующим образом:

- Pipfile и Pipfile.lock – файлы, содержащие информацию о третьесторонних зависимостях программы;
- Makefile – файл с инструкциями для утилиты make, которая нужна для автоматической сборки программы;
- KB.g4 – грамматика для ANTLR, описывающая синтаксис для описания нечётких правил и фактов;
- src – каталог, содержащий исходный код программы;
- src/fuzzy_logic.py – содержит алгоритм для прямого нечеткого логического вывода;
- src/main.py – содержит входную точку программы;
- src/parser – каталог, содержащий исходный код анализатора правил и фактов;
- src/parser/parser.py – содержит алгоритм для анализа строк, содержащих факты и правила.

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).

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

$\langle \text{база знаний} \rangle ::= \langle \text{список фактов} \rangle \ \langle \text{список правил} \rangle$

$\langle \text{список фактов} \rangle ::= \langle \text{факт} \rangle \ \langle \text{факт} \rangle$

$\langle \text{список правил} \rangle ::= \langle \text{правило} \rangle \ \langle \text{правило} \rangle$

$\langle \text{факт} \rangle ::= \langle \text{имя нечёткого множества} \rangle = \langle \text{нечёткое множество} \rangle$

$\langle \text{правило} \rangle ::= \langle \text{имя нечёткого множества} \rangle \sim \langle \text{имя нечёткого множества} \rangle .$

$\langle \text{нечёткое множество} \rangle ::= \{ \langle \text{список пар нечёткой принадлежности} \rangle \}$

$\langle \text{список пар нечёткой принадлежности} \rangle ::=$

$\langle \text{пара нечёткой принадлежности} \rangle, \langle \text{пара нечёткой принадлежности} \rangle$

$\langle \text{пара нечёткой принадлежности} \rangle ::= (\langle \text{элемент} \rangle, \langle \text{степень принадлежности} \rangle)$

$\langle \text{элемент} \rangle ::= \langle \text{имя} \rangle \mid \langle \text{множество} \rangle$

$\langle \text{множество} \rangle ::= \langle \text{ориентированное множество} \rangle \mid$

$\langle \text{неориентированное множество} \rangle$

$\langle \text{неориентированное множество} \rangle ::= \{ \langle \text{список элементов} \rangle \}$

$\langle \text{ориентированное множество} \rangle ::= (\langle \text{элемент} \rangle, \langle \text{список элементов} \rangle)$

$\langle \text{список элементов} \rangle ::= \langle \text{элемент} \rangle, \langle \text{элемент} \rangle, \langle \text{элемент} \rangle$

$\langle \text{имя нечёткого множества} \rangle ::= \langle \text{имя} \rangle$

$\langle \text{имя} \rangle ::= \langle \text{символ} \rangle \langle \text{символ} \rangle$

$\langle \text{символ} \rangle ::= \langle \text{буква} \rangle | \langle \text{цифра} \rangle$

$\langle \text{цифра} \rangle ::= 0 | \dots | 9$

$\langle \text{буква} \rangle ::= A | \dots | z$

$\langle \text{степень принадлежности} \rangle ::= \langle \text{действительное число с 0 по 1} \rangle$

$\langle \text{действительное число с 0 по 1} \rangle ::= \langle \text{единица} \rangle |$

$\langle \text{действительное число с 0 до 1} \rangle$

$\langle \text{действительное число с 0 до 1} \rangle ::= 0 . \langle \text{цифра} \rangle$

$\langle \text{единица} \rangle ::= 1 . 0$

1.4 Реализация

Программа включает в себя класс `FuzzyLogic`, который отвечает за прямой нечеткий логический вывод. Он включает в себя следующие методы:

- `__impl(set1, set2)` – реализация функции импликации Гёделя;
- `__and(set1, matrix)` – реализация Т-нормы;
- `__infer(self, set1, set2, set3)` – метод, находящий выводимый предикат;
- `run_inference(self, sets, rules)` – метод, находящий все выводимые уникальные предикаты.

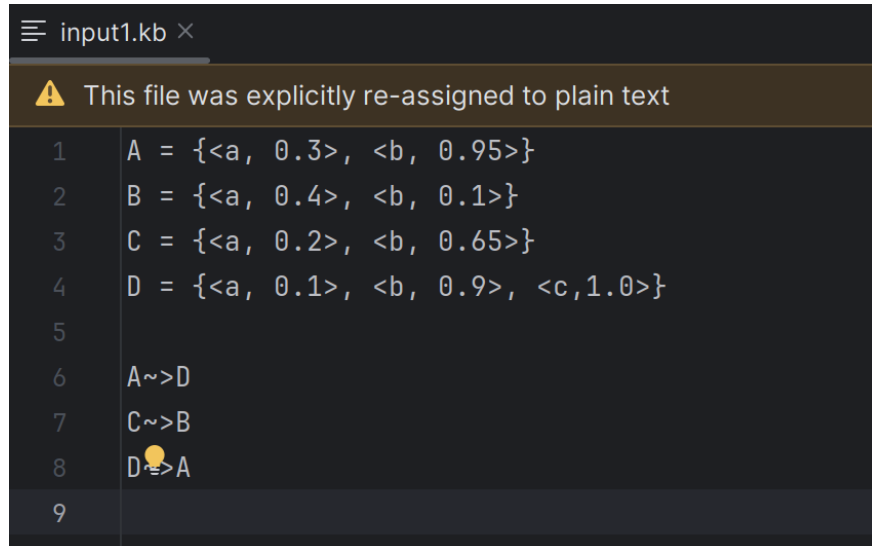
Для чтения правил и фактов из файла программа включает в себя класс `ANTLRFacade`, который содержит следующие методы:

- `__get_assignments(self)` – получает множество фактов из текста;
- `__get_impls(self)` – получает множество правил из текста;
- `parse(self)` – получить пару из множества фактов и множества правил.

1.5 Демонстрация результатов работы программы

1.5.1 Тест 1

Входной файл input1.kb:



```
input1.kb ×
⚠ This file was explicitly re-assigned to plain text
1  A = {<a, 0.3>, <b, 0.95>}
2  B = {<a, 0.4>, <b, 0.1>}
3  C = {<a, 0.2>, <b, 0.65>}
4  D = {<a, 0.1>, <b, 0.9>, <c, 1.0>}
5
6  A~>D
7  C~>B
8  D~>A
9
```

Рисунок 1.1 – Входной файл input1.kb для теста 1

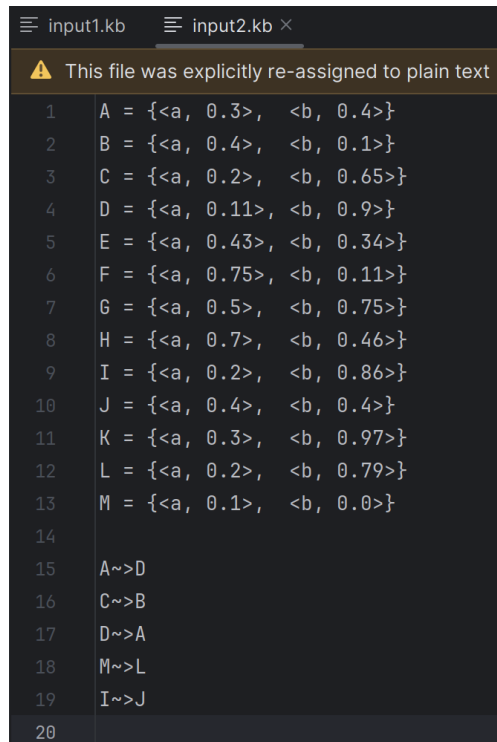
Результат работы программы для файла input1.kb:

```
(LOIS) rlipiski@rlipiski-pc:~/opt/LOIS$ python src/main.py --kb example/input1.kb
I4 = A/~\ (A~>D) = {<b, 0.9>, <a, 0.1>, <c, 0.95>}
I5 = B/~\ (A~>D) = {<c, 0.4>, <b, 0.4>, <a, 0.1>}
I6 = C/~\ (A~>D) = {<c, 0.65>, <b, 0.65>, <a, 0.1>}
I7 = I5/~\ (D~>A) = {<b, 0.4>, <a, 0.3>}
I8 = I6/~\ (D~>A) = {<b, 0.65>, <a, 0.3>}
```

Рисунок 1.2 – Результат работы программы для файла input1.kb

1.5.2 Тест 2

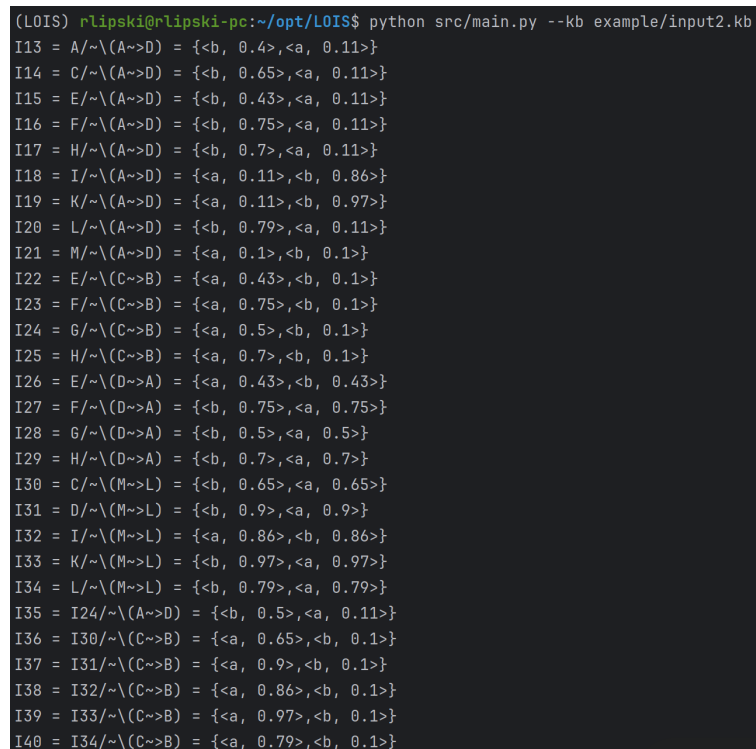
Входной файл input2.kb:



```
input1.kb  input2.kb ×
⚠ This file was explicitly re-assigned to plain text
1  A = {<a, 0.3>, <b, 0.4>}
2  B = {<a, 0.4>, <b, 0.1>}
3  C = {<a, 0.2>, <b, 0.65>}
4  D = {<a, 0.11>, <b, 0.9>}
5  E = {<a, 0.43>, <b, 0.34>}
6  F = {<a, 0.75>, <b, 0.11>}
7  G = {<a, 0.5>, <b, 0.75>}
8  H = {<a, 0.7>, <b, 0.46>}
9  I = {<a, 0.2>, <b, 0.86>}
10 J = {<a, 0.4>, <b, 0.4>}
11 K = {<a, 0.3>, <b, 0.97>}
12 L = {<a, 0.2>, <b, 0.79>}
13 M = {<a, 0.1>, <b, 0.0>}
14
15 A~>D
16 C~>B
17 D~>A
18 M~>L
19 I~>J
20
```

Рисунок 1.3 – Входной файл input2.kb для теста 2

Результат работы программы для файла input2.kb:



```
(LOIS) rlipiski@rlipiski-pc:~/opt/LOIS$ python src/main.py --kb example/input2.kb
I13 = A/~\ (A~>D) = {<b, 0.4>, <a, 0.11>}
I14 = C/~\ (A~>D) = {<b, 0.65>, <a, 0.11>}
I15 = E/~\ (A~>D) = {<b, 0.43>, <a, 0.11>}
I16 = F/~\ (A~>D) = {<b, 0.75>, <a, 0.11>}
I17 = H/~\ (A~>D) = {<b, 0.7>, <a, 0.11>}
I18 = I/~\ (A~>D) = {<a, 0.11>, <b, 0.86>}
I19 = K/~\ (A~>D) = {<a, 0.11>, <b, 0.97>}
I20 = L/~\ (A~>D) = {<b, 0.79>, <a, 0.11>}
I21 = M/~\ (A~>D) = {<a, 0.1>, <b, 0.1>}
I22 = E/~\ (C~>B) = {<a, 0.43>, <b, 0.1>}
I23 = F/~\ (C~>B) = {<a, 0.75>, <b, 0.1>}
I24 = G/~\ (C~>B) = {<a, 0.5>, <b, 0.1>}
I25 = H/~\ (C~>B) = {<a, 0.7>, <b, 0.1>}
I26 = E/~\ (D~>A) = {<a, 0.43>, <b, 0.43>}
I27 = F/~\ (D~>A) = {<b, 0.75>, <a, 0.75>}
I28 = G/~\ (D~>A) = {<b, 0.5>, <a, 0.5>}
I29 = H/~\ (D~>A) = {<b, 0.7>, <a, 0.7>}
I30 = C/~\ (M~>L) = {<b, 0.65>, <a, 0.65>}
I31 = D/~\ (M~>L) = {<b, 0.9>, <a, 0.9>}
I32 = I/~\ (M~>L) = {<a, 0.86>, <b, 0.86>}
I33 = K/~\ (M~>L) = {<b, 0.97>, <a, 0.97>}
I34 = L/~\ (M~>L) = {<b, 0.79>, <a, 0.79>}
I35 = I24/~\ (A~>D) = {<b, 0.5>, <a, 0.11>}
I36 = I30/~\ (C~>B) = {<a, 0.65>, <b, 0.1>}
I37 = I31/~\ (C~>B) = {<a, 0.9>, <b, 0.1>}
I38 = I32/~\ (C~>B) = {<a, 0.86>, <b, 0.1>}
I39 = I33/~\ (C~>B) = {<a, 0.97>, <b, 0.1>}
I40 = I34/~\ (C~>B) = {<a, 0.79>, <b, 0.1>}
```

Рисунок 1.4 – Результат работы программы для файла input2.kb

1.5.3 Тест 3

Входной файл input3.kb:

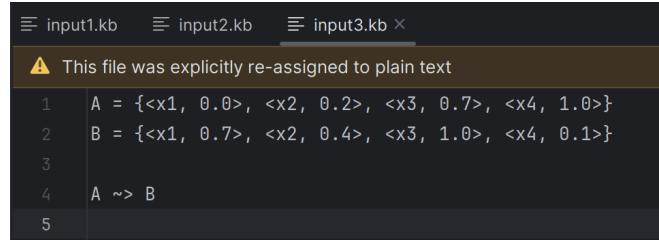


Рисунок 1.5 – Входной файл input3.kb для теста 3

Результат работы программы для файла input3.kb:

```
(LOIS) rlipiski@rlipiski-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 Контрольные вопросы

1. Если множества α и β являются нормальными, то возможен ли случай при каких-либо значениях α' , когда результат не будет являться нормальным множеством?

Приведем пример, когда результат не будет нормальным множеством:

Вход программы:

$$\begin{aligned} A &= \{ \langle x, 1.0 \rangle, \langle y, 0.1 \rangle, \langle z, 0.4 \rangle \} \\ B &= \{ \langle x, 1.0 \rangle, \langle y, 0.6 \rangle, \langle z, 0.3 \rangle \} \\ C &= \{ \langle x, 0.3 \rangle, \langle y, 0.3 \rangle, \langle z, 0.4 \rangle \} \\ A &\leadsto B \end{aligned}$$

Выход:

$$\begin{aligned} I_3 &= B \tilde{\wedge} (A \leadsto B) = \{ \langle y, 0.6 \rangle, \langle z, 0.6 \rangle, \langle x, 1 \rangle \} \\ I_4 &= C \tilde{\wedge} (A \leadsto B) = \{ \langle y, 0.4 \rangle, \langle z, 0.3 \rangle, \langle x, 0.4 \rangle \} \\ I_5 &= I_4 \tilde{\wedge} (A \leadsto B) = \{ \langle y, 0.4 \rangle, \langle z, 0.4 \rangle, \langle x, 0.4 \rangle \} \end{aligned}$$

Множества I_4 , I_5 не являются нормальными, следовательно, такие случаи возможны.

Если предикат C не является нормальным, то результат также не будет нормальным, потому что значения степеней принадлежности будут меньше 1, что приведет к образованию не нормального множества в результате нечеткого вывода.

2. Если множества α и β не являются нормальными, то возможен ли случай при каких-либо значениях α' , когда результат будет являться нормальным множеством?

Приведем пример, когда результат будем нормальным множеством:
Вход программы:

$$A = \{ \langle x, 0.4 \rangle, \langle y, 0.1 \rangle, \langle z, 0.4 \rangle \}$$

$$B = \{ \langle x, 0.3 \rangle, \langle y, 0.6 \rangle, \langle z, 0.3 \rangle \}$$

$$C = \{ \langle x, 0.3 \rangle, \langle y, 0.3 \rangle, \langle z, 1.0 \rangle \}$$

$$A \rightsquigarrow B$$

Выход:

$$I_3 = A \tilde{\wedge} (A \rightsquigarrow B) = \{ \langle x, 0.3 \rangle, \langle y, 0.4 \rangle, \langle z, 0.3 \rangle \}$$

$$I_4 = B \tilde{\wedge} (A \rightsquigarrow B) = \{ \langle z, 0.6 \rangle, \langle y, 0.6 \rangle, \langle x, 0.6 \rangle \}$$

$$I_5 = C \tilde{\wedge} (A \rightsquigarrow B) = \{ \langle y, 1 \rangle, \langle x, 0.3 \rangle, \langle z, 0.3 \rangle \}$$

$$I_6 = I_3 \tilde{\wedge} (A \rightsquigarrow B) = \{ \langle z, 0.4 \rangle, \langle y, 0.4 \rangle, \langle x, 0.4 \rangle \}$$

$$I_7 = I_5 \tilde{\wedge} (A \rightsquigarrow B) = \{ \langle y, 1 \rangle, \langle x, 1 \rangle, \langle z, 1 \rangle \}$$

Множества I_5 , I_7 являются нормальными, следовательно, такие случаи возможны.

3. Какими значениями α' , α и β можно гарантировать, что результат будет являться нормальным множеством?

– Факт, используемый при выводе, является нормальным множеством.

1.7 Личный вклад

В рамках данной лабораторной работы Липским Р. В. был реализован алгоритм нечеткого логического вывода на языке Python.

Стронгиным А. В. была разработана грамматика и парсер программы.
Жолнерчиком И. А. был подготовлен отчёт о выполненной работе.

2 ВЫВОД

В процессе выполнения лабораторной работы, мы получили навыки реализации нечёткой логики, а именно прямого нечёткого логического вывода при помощи программирования. В рамках данной работы были разработаны модули, отвечающие за анализ исходного текста базы знаний, а также непосредственно алгоритм прямого нечёткого логического вывода.

При помощи разработанного программного продукта нам удалось построить корректные выводы для нескольких случаев, а также дать ответы на контрольные вопросы, прилагающиеся к лабораторной работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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