Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и

Радиоэлектроники

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №3

по курсу “Логические основы интеллектуальных систем”

Вариант 4

Выполнил:

Студент гр. 321701 Самович В.М.

Проверил: Ивашенко В.П.

Минск 2025

**Тема:**

Программирование операций обработки и преобразования формул прикладных и неклассических логик.

**Цель:**

Приобрести навыки программирования алгоритмов обработки данных в неклассических логических моделях решения задач.

**Задание:**

Реализовать прямой нечёткий логический вывод, используя треугольную норму драстического произведения и нечёткую импликацию Вебера.

**Формат базы знаний:**

<база знаний>::= <список фактов>[<список правил>]

<список фактов>::= <факт>[{<факт>}]

<список правил>::= <правило>[{<правило>}]

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

<правило>::= <имя нечёткого множества>~><имя нечёткого множества>.

<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>}]

<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}]]

**Схемы функций программы:**

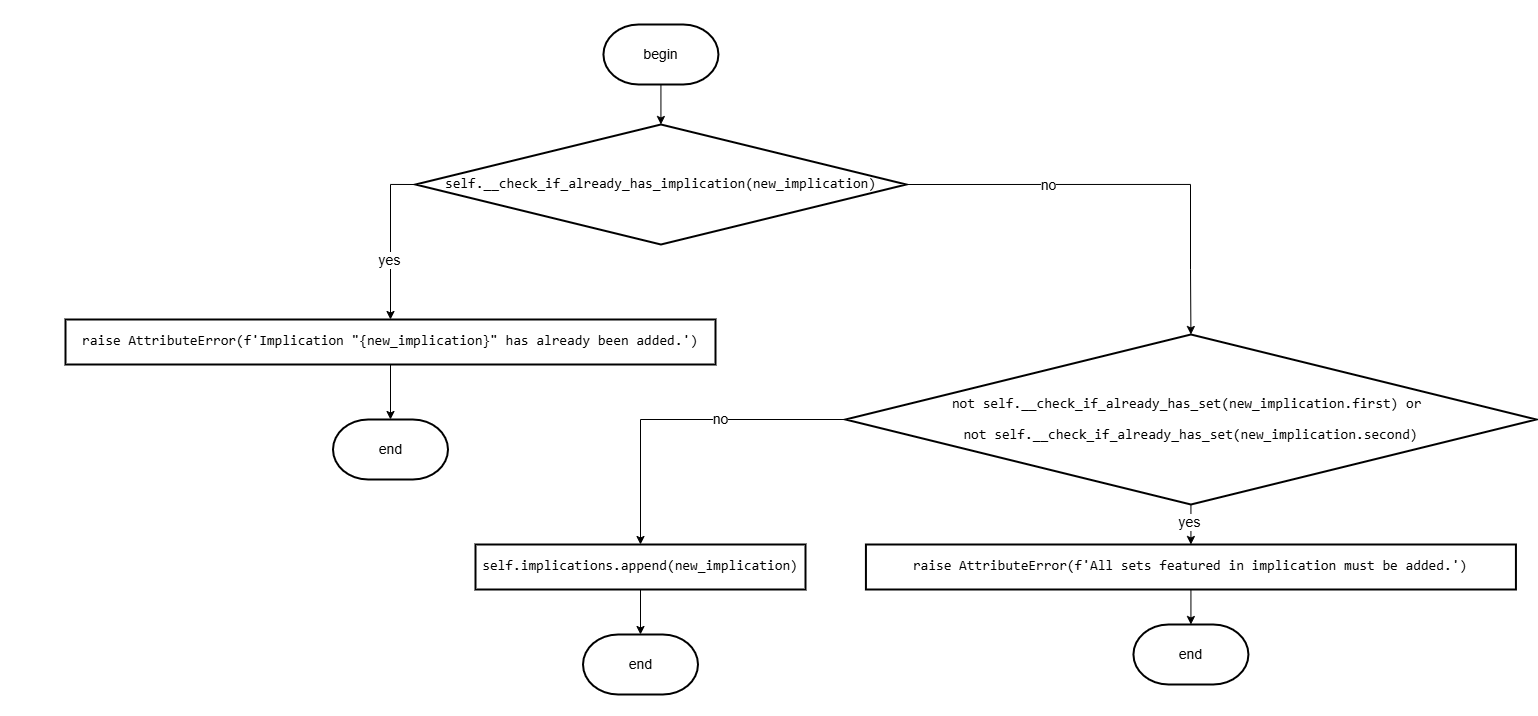


Рис 1. Метод add\_implication(self, new\_implication :ImplicationScheme) класса InferenceInput

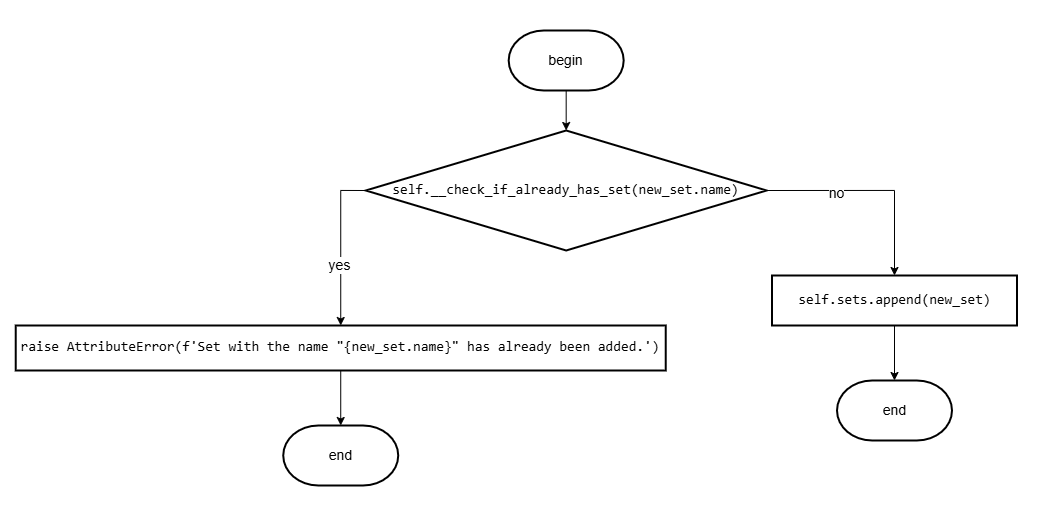


Рис 2. Метод add\_set(self, new\_set :FuzzySet) класса InferenceInput

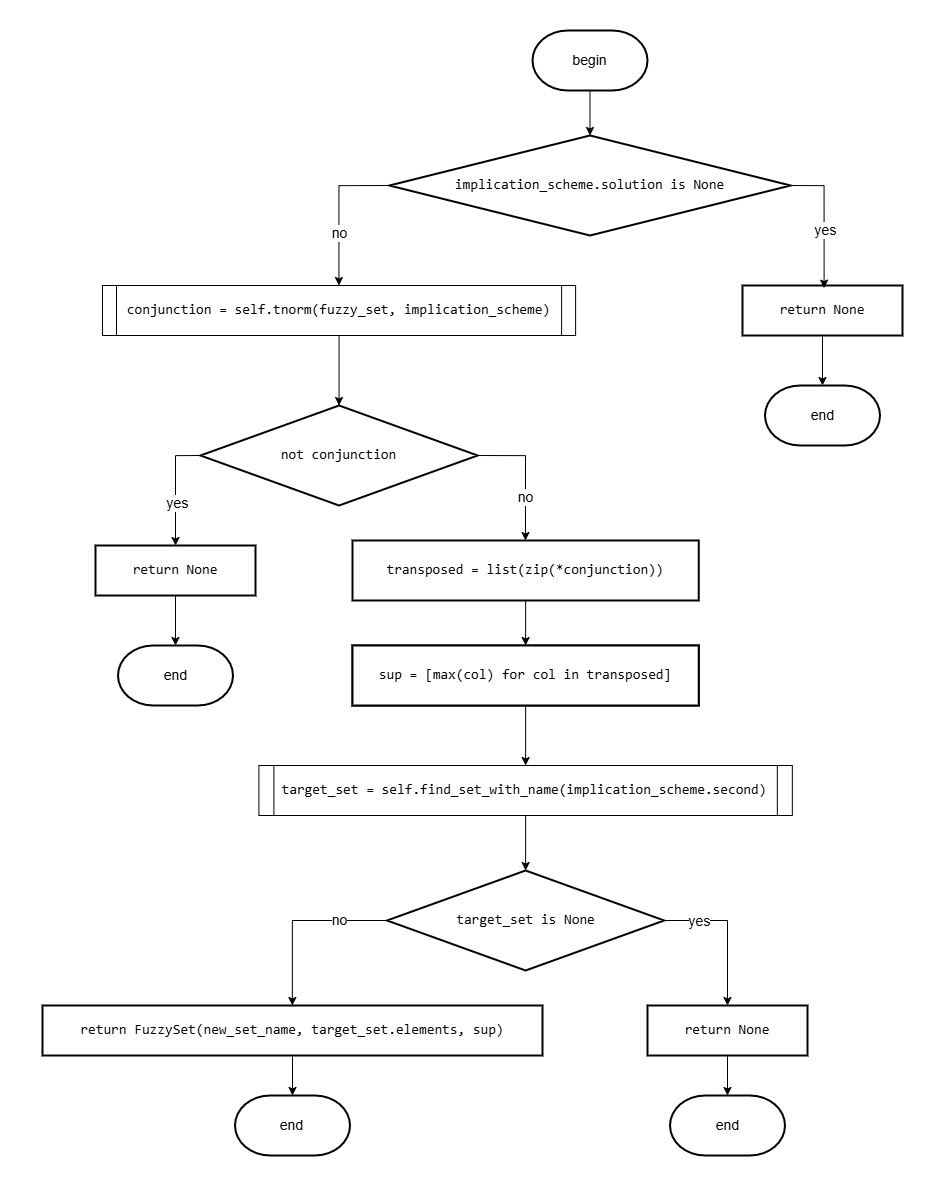


Рис 3. Метод apply\_modus\_ponens(self, fuzzy\_set :FuzzySet, implication\_scheme :ImplicationScheme, new\_set\_name :str) класса InferencePipeline

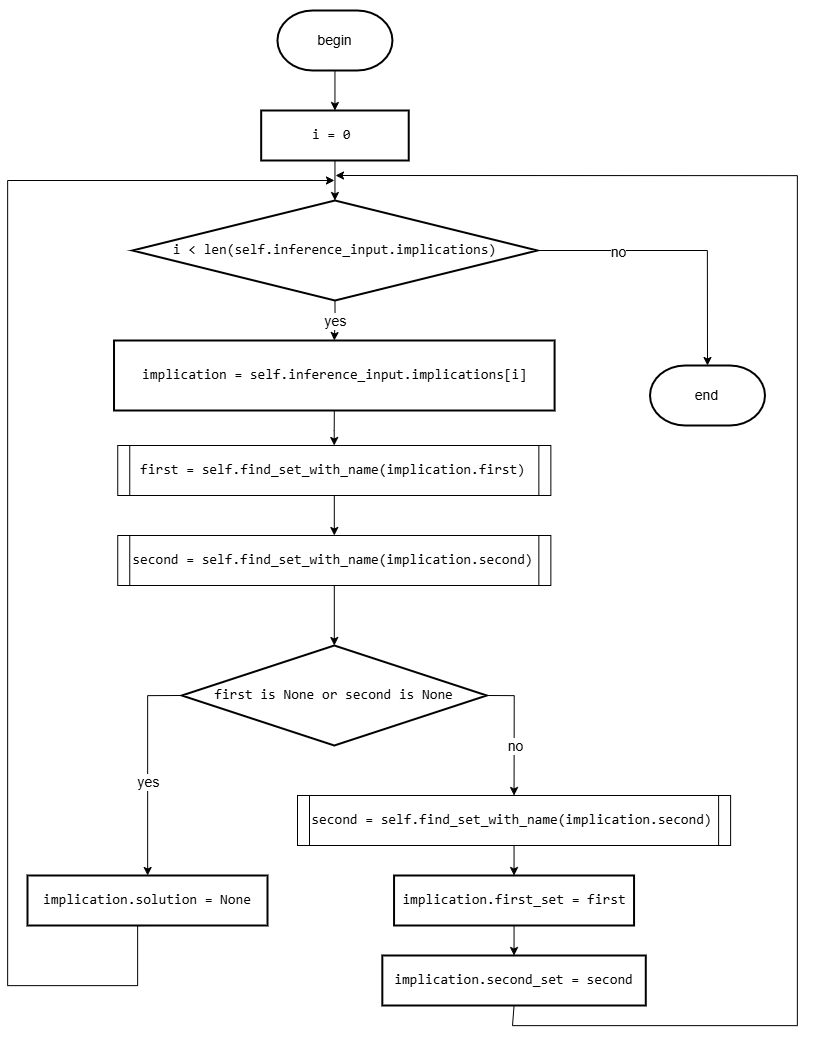


Рис 4. Метод calculate\_implications(self) класса InferencePipeline

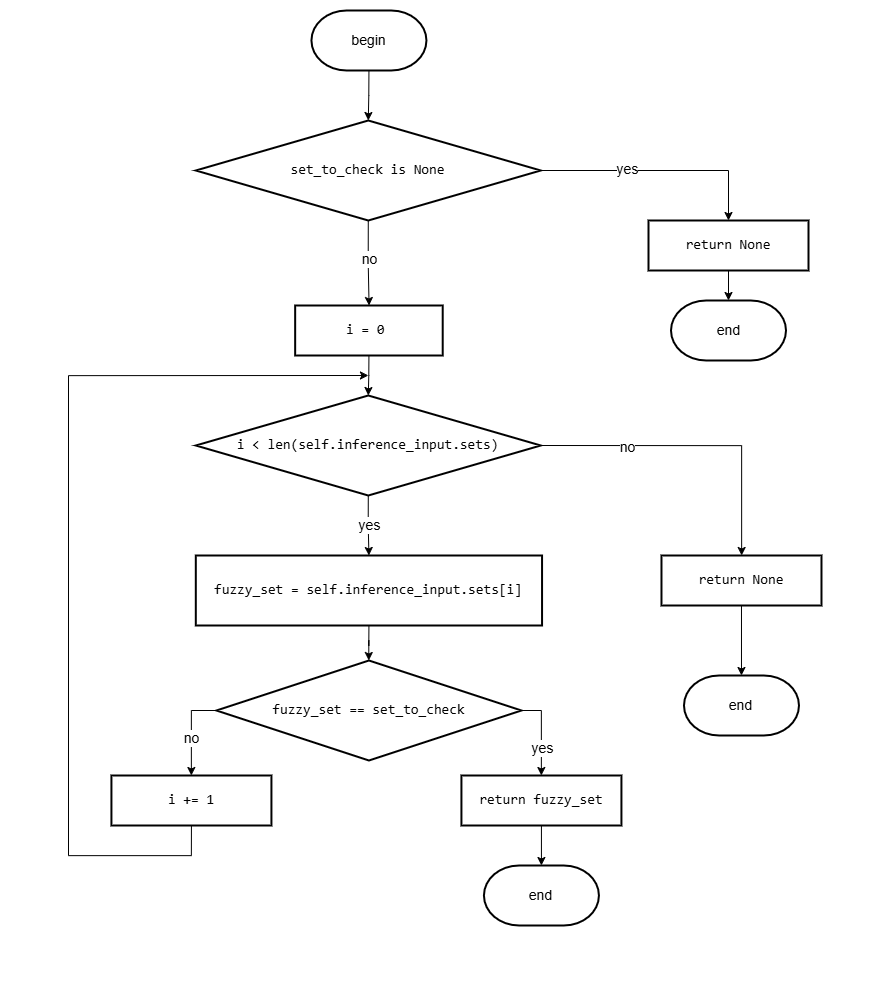


Рис 5. Метод check\_for\_duplicates(self, set\_to\_check :FuzzySet) класса InferencePipeline

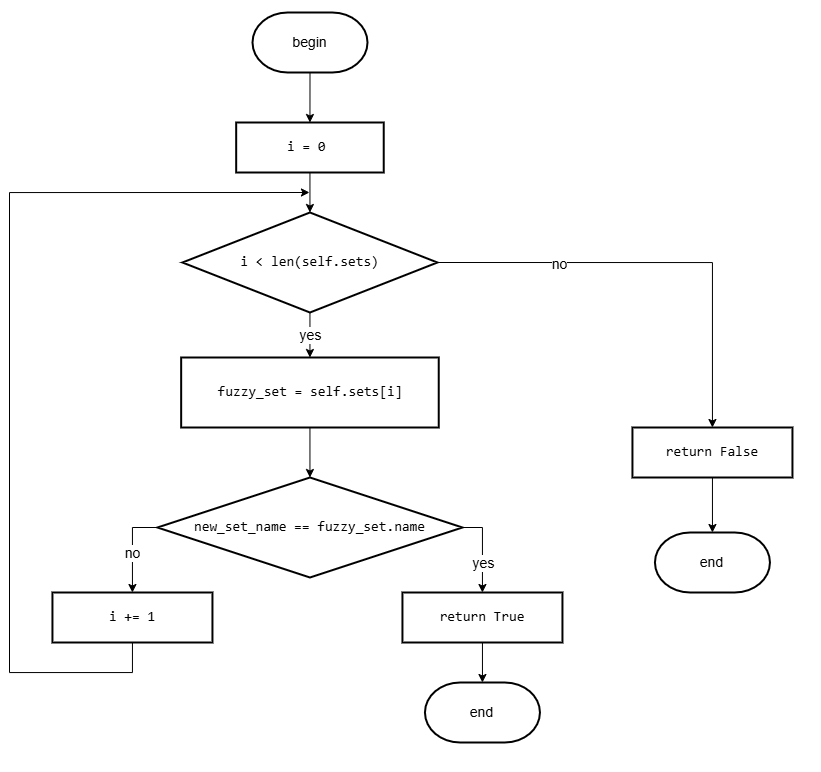


Рис 6. Метод \_\_check\_if\_already\_has\_set(self, new\_set\_name) класса InferenceInput

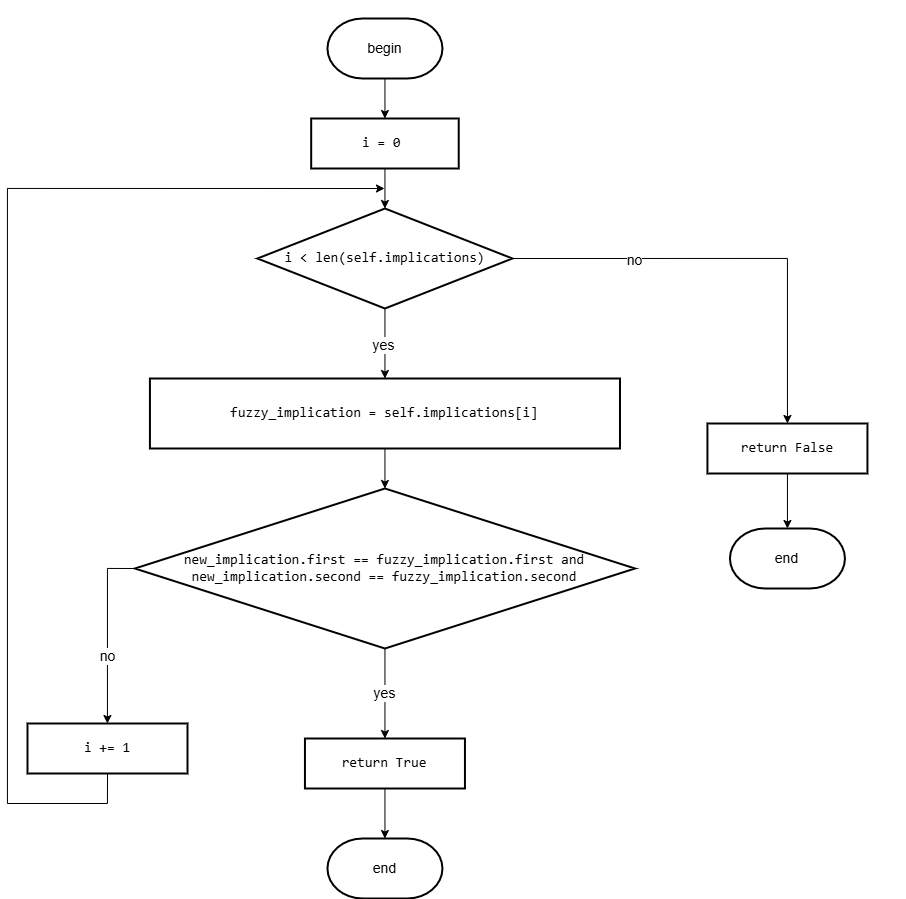


Рис 7. Метод \_\_check\_if\_already\_has\_implication(self, new\_implication :ImplicationScheme) класса InferenceInput

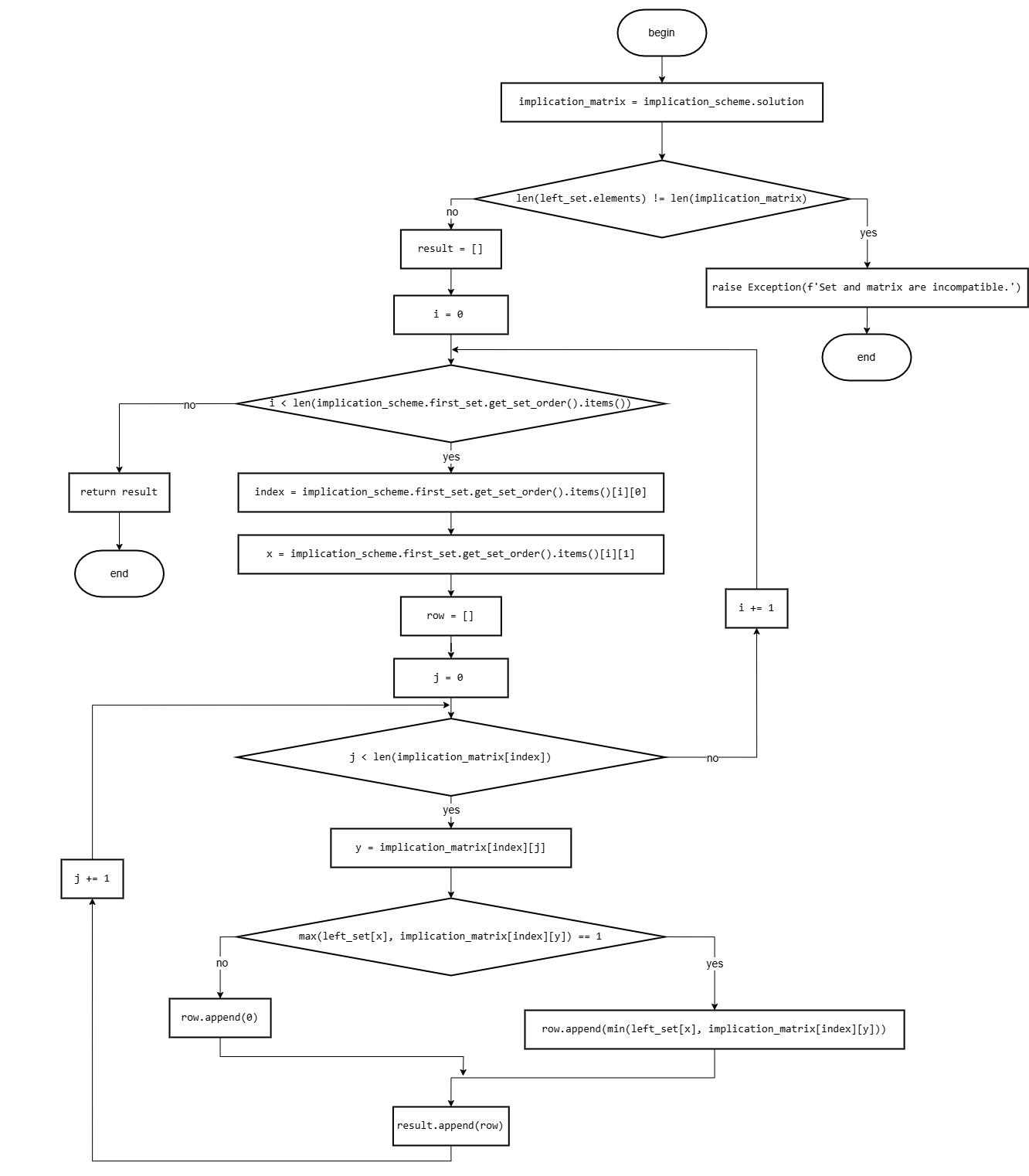


Рис 8. Метод drastic\_product(left\_set :FuzzySet, implication\_scheme :ImplicationScheme) класса InferenceConjunction

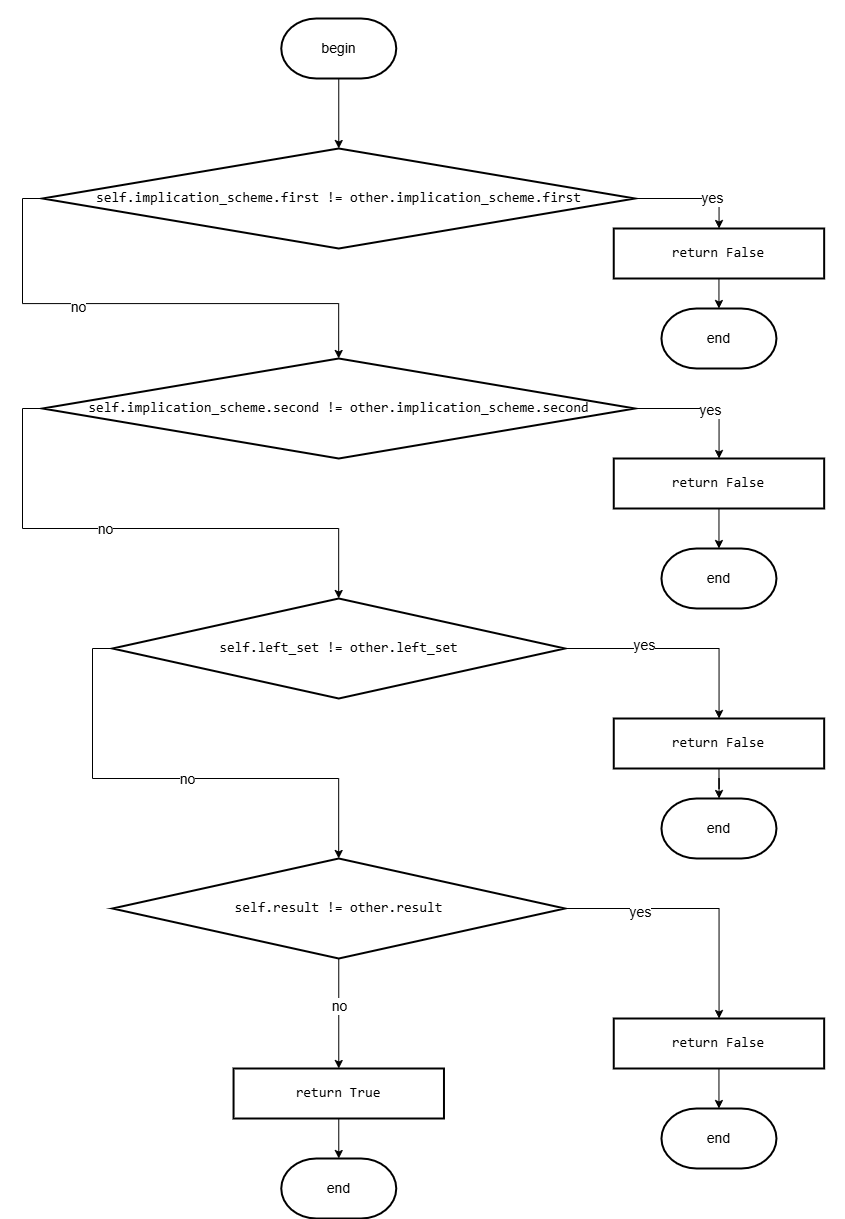


Рис 9. Метод \_\_eq\_\_(self, other) класса InferenceStep

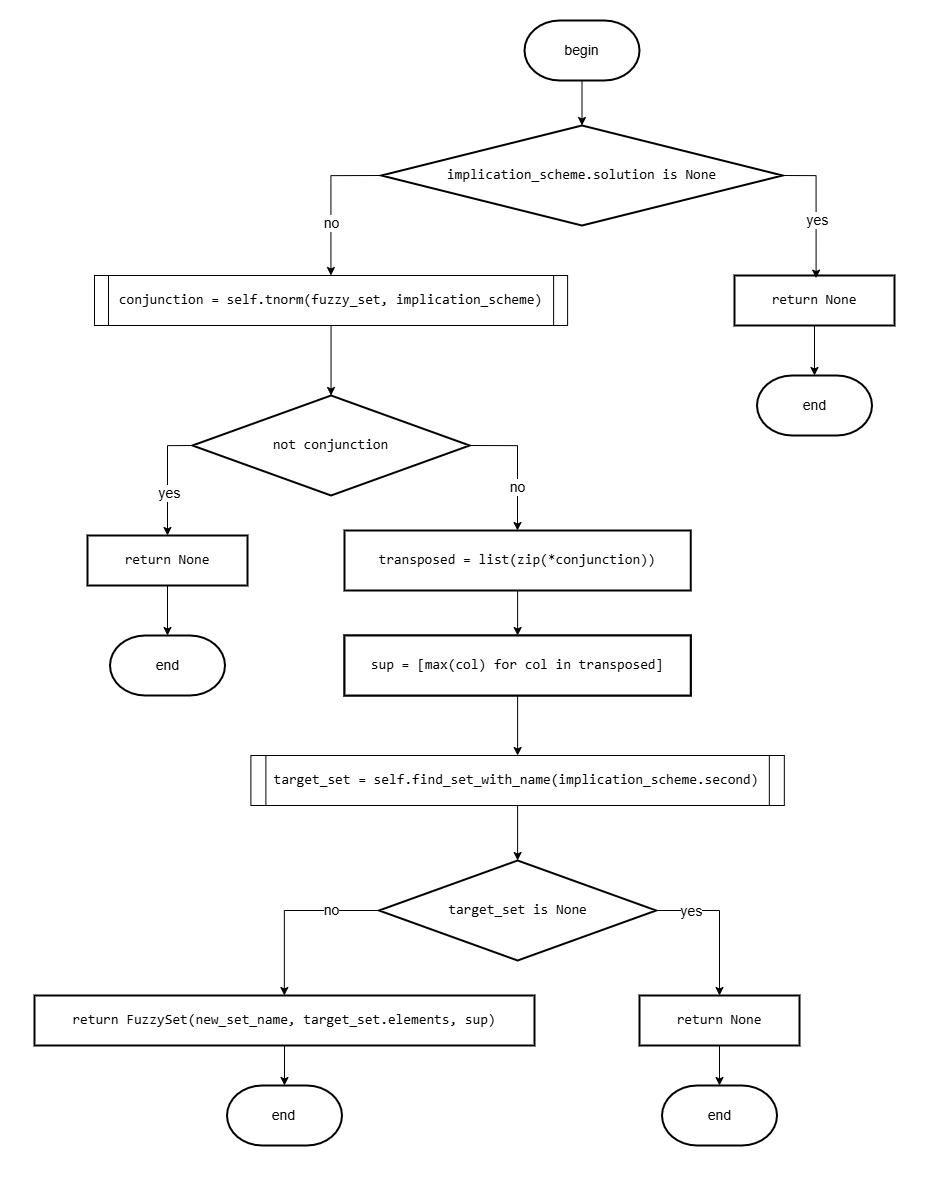


Рис 10. Метод find\_compatible\_sets(self, implication\_scheme :ImplicationScheme) класса InferencePipeline

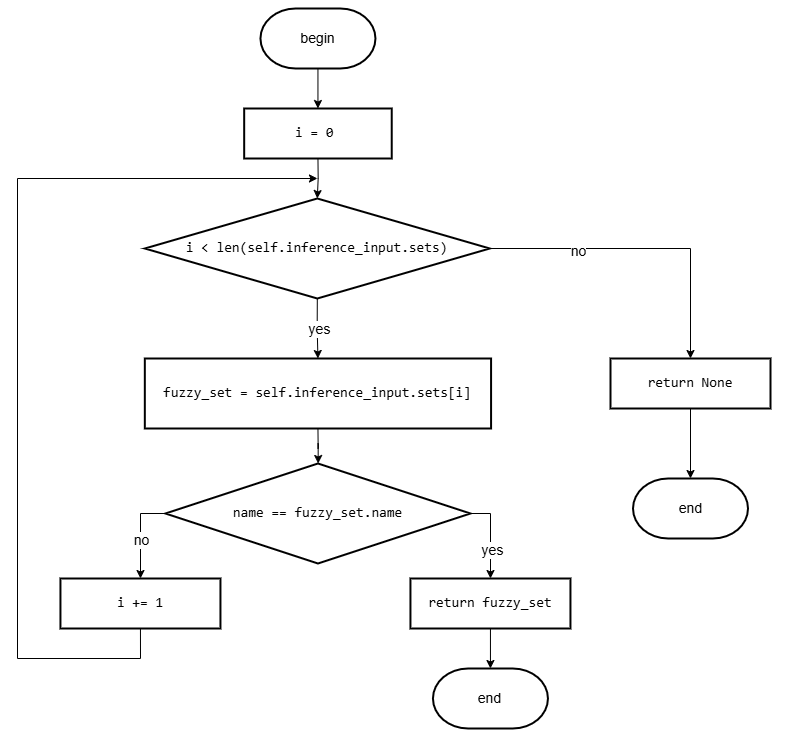


Рис 11. Метод find\_set\_with\_name(self, name :str) класса InferencePipeline

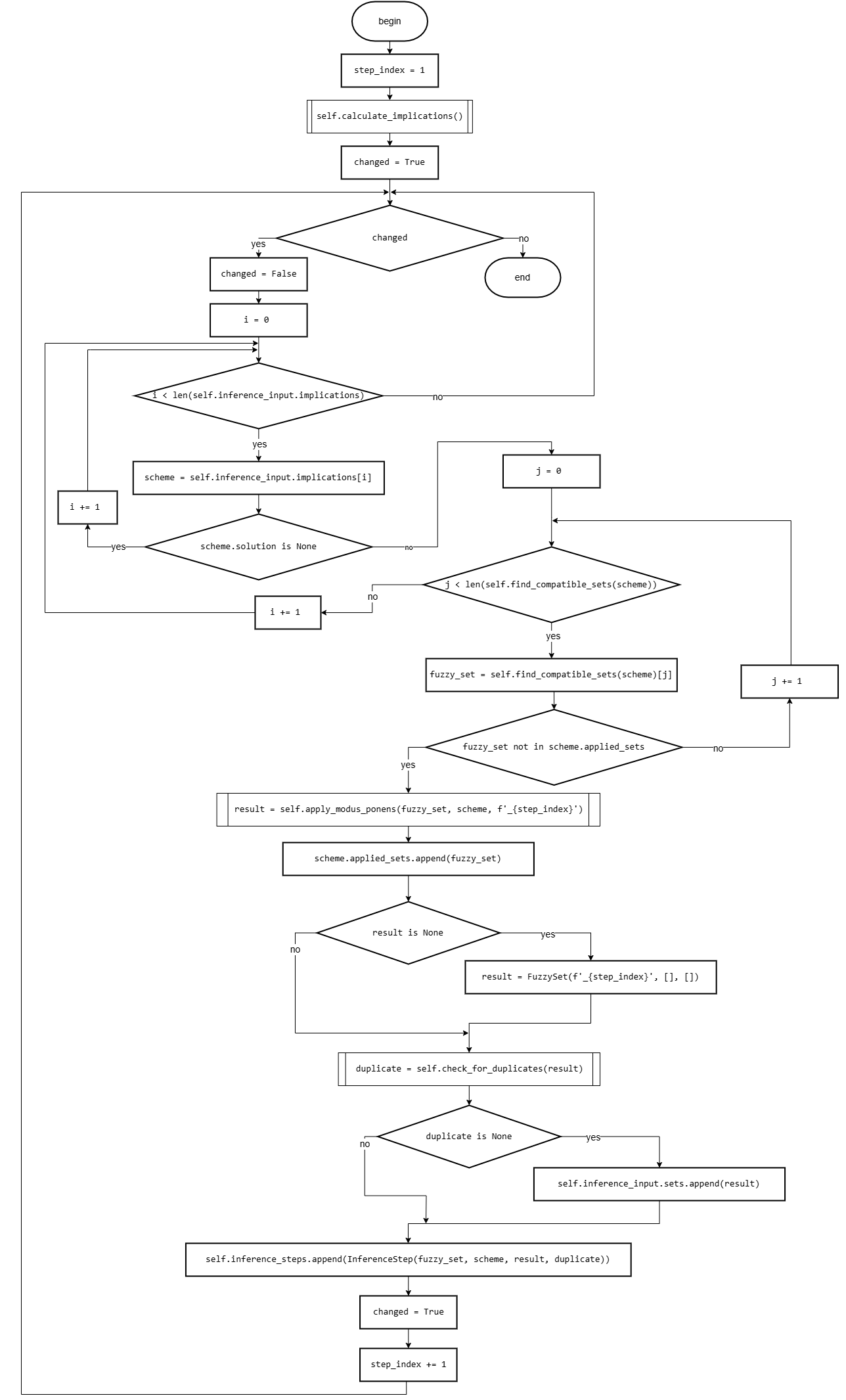


Рис 12. Метод inference(self) класса InferencePipeline

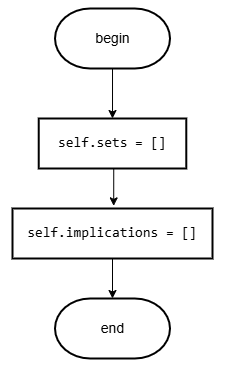


Рис 13. Конструктор \_\_init\_\_(self) класса InferenceInput

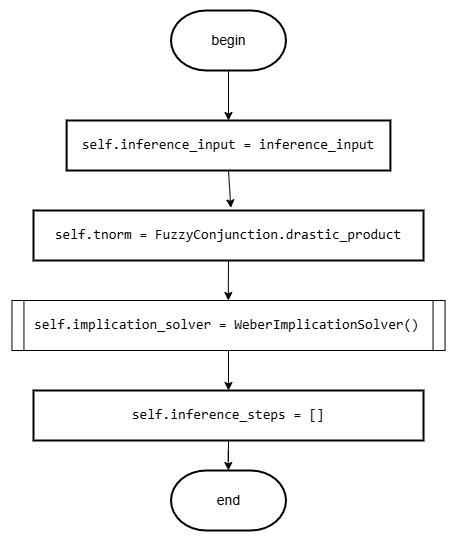


Рис 14. Конструктор \_\_init\_\_(self) класса InferenceInput

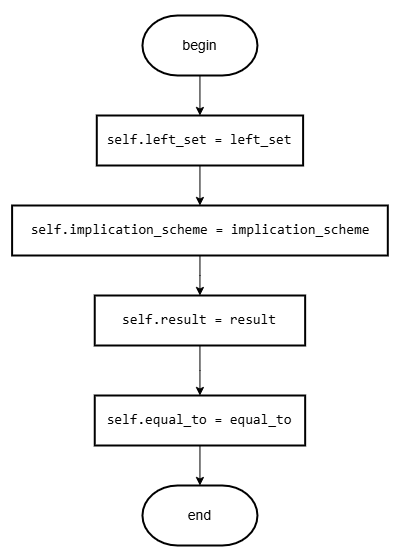


Рис 15. Конструктор \_\_init\_\_(self, left\_set, implication\_scheme, result, equal\_to = None) класса InferenceStep

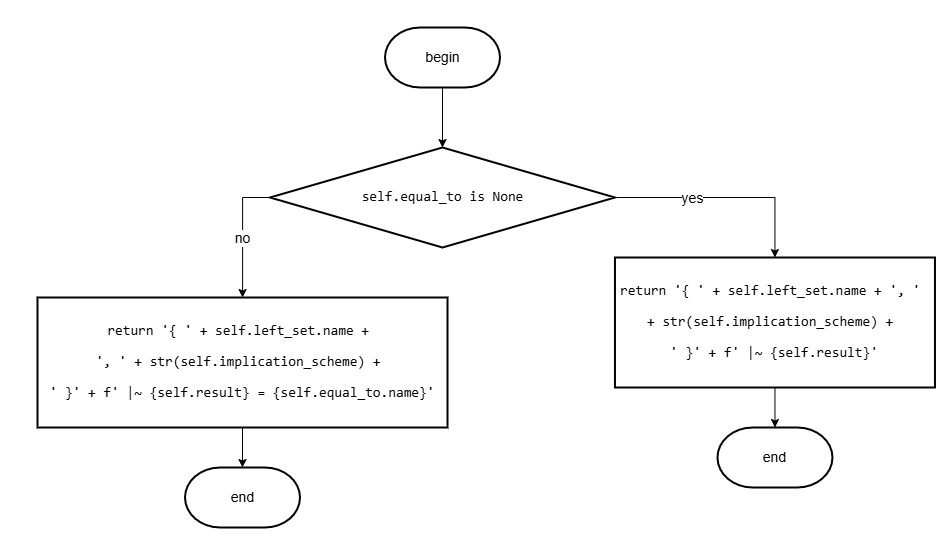


Рис 16. Конструктор \_\_str\_\_(self) класса InferenceStep

**Программная реализация:**

Для выполнения данной лабораторной работы был использован язык программирования Python. В программе используются такие структуры данных, как список, словарь и кортеж. Пользователь имеет возможность вводить факты и правила, как из консоли, так и из файлов.

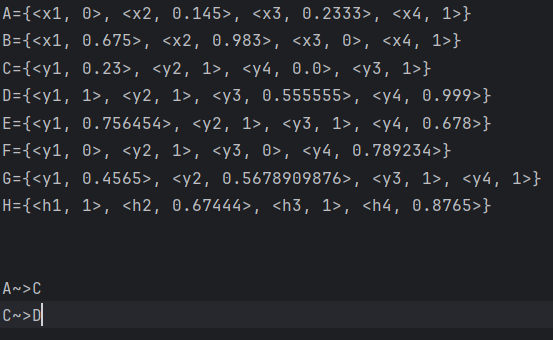


Рис. 17. Входные данные для примера 1

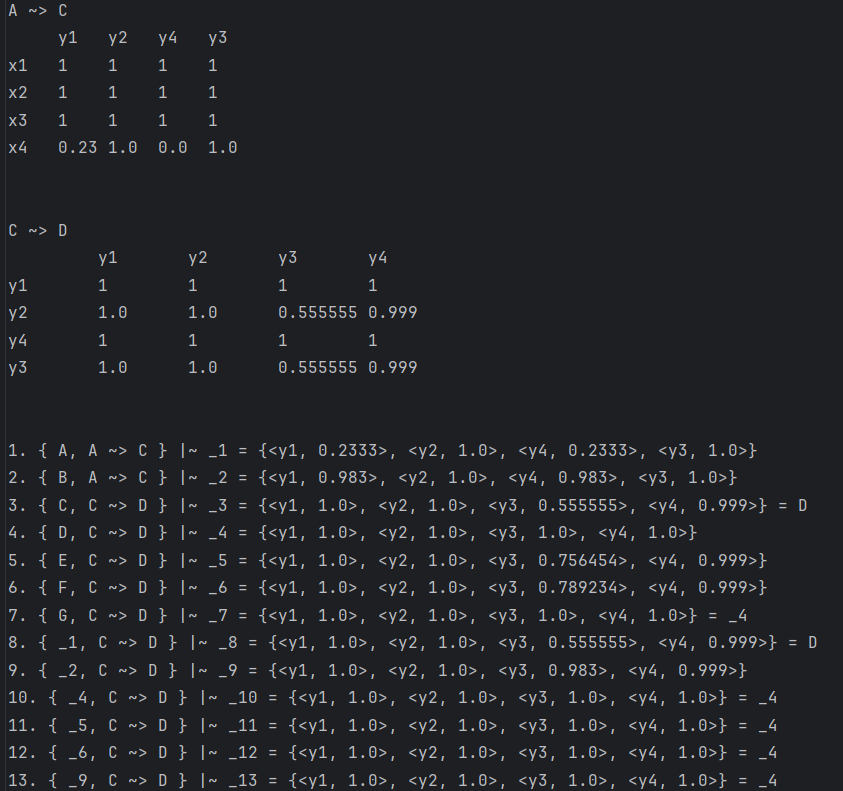


Рис. 18. Результат работы программы для примера 1

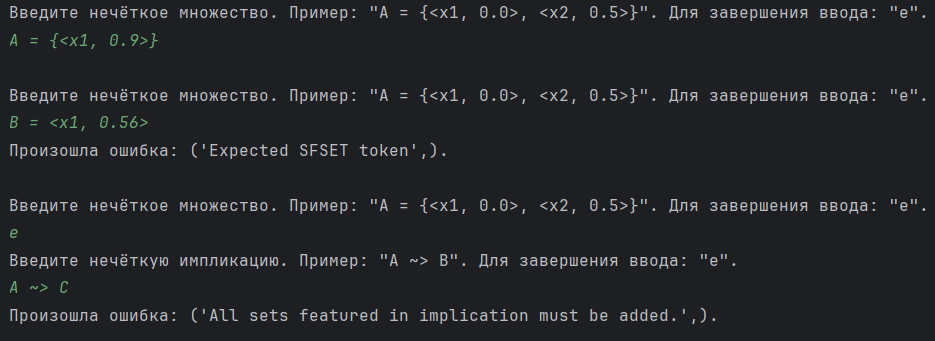


Рис. 19. Пример 2. Поведение программы при вводе некорректных данных

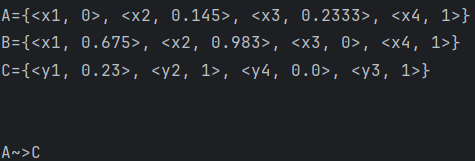


Рис. 20. Пример 3 входные данные

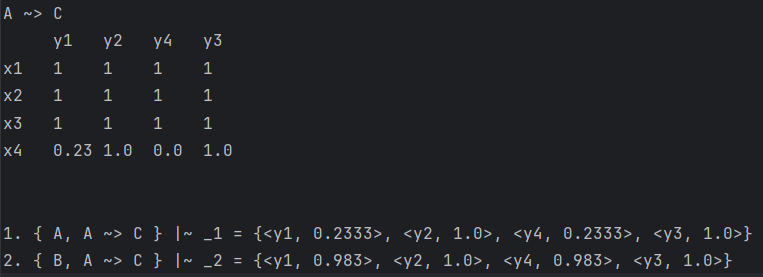


Рис. 21. Пример 3 результат

**Вывод:**

В ходе работы были приобретены навыки программирования алгоритмов обработки данных в неклассических логических моделях решения задач.

**Теоретические сведения были взяты из следующих источников:**

1. Логические основы интеллектуальных систем. Практикум: учебно-методическое пособие / В. В. Голенков, В. П. Ивашенко, Д. Г. Колб, К. А. Уваров. – Минск: БГУИР, 2011.