УДК 658.012.011.56

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ НЕЧЕТКОЙ ЛОГИКИ С ЛИНГВИСТИЧЕСКИМИ ПЕРЕМЕННЫМИ

Загибин Никита Олегович[1], Ульянов Сергей Викторович[2]

[1]Студент;
Государственный университет «Дубна»;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: nekitusb@mail.ru.

[2]Доктор физико-математических наук, профессор;
Государственный университет «Дубна»;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

В статье рассматриваются вопросы, связанные с реализацией интеллектуальных систем управления, в частности нечеткого регулятора, а также представлена разработанная программная библиотека, позволяющая строить нечеткие логические заключения, основанные на нечетких множествах. Все это расширяет классическое понятие логики вводя рассуждения, приближенные к нечеткому суждению человека. Программная реализация в виде подключаемой библиотеки позволит разработчикам создать свою базу знания нечеткой логики или нечеткие логические конструкции из нечетких множеств для проектов, направленных на моделирование человеческого умозаключения или разработки ИСУ. Данная работа составляет базис для разработки базы знаний с нечеткой логикой, построенной на лингвистических переменных. Таким образом, в качестве начального базиса в области разработки интеллектуальных систем управления были разработаны классы, поддерживающие нечеткие вычисления и нечеткую логику с помощью которых впоследствии можно проектировать базы знаний, гарантирующие достижения поставленных целей управления.

Ключевые слова: нечеткие множества, лингвистическая переменная, нечеткий вывод, нечеткие логические операции.

Для цитирования:
Загибин Н. О., Ульянов С. В. Программная реализация нечеткой логики с лингвистическими переменными // Системный анализ в науке и образовании: сетевое научное издание. 2021. № 1. С. 45–57. URL : http://sanse.ru/download/426.

SOFTWARE IMPLEMENTATION OF FUZZY LOGIC WITH LINGUISTIC VARIABLES

Zagibin Nikita O.[1], Ulyanov Sergey V.[2]

[1]Student;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: nekitusb@mail.ru.

[2]Doctor of Science in Physics and Mathematics, professor;
Dubna State University;
19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;
e-mail: ulyanovsv@mail.ru.

The article deals with issues related to the implementation of intelligent control systems, in particular the fuzzy controller, and also a developed software library is presented that allows you to build fuzzy logical conclusions based on fuzzy sets. All this expands the classical concept of logic by introducing reasoning that is close to the fuzzy judgment of a person. The software implementation in the form of a plug-in library will allow developers to create their own knowledge base of fuzzy logic or fuzzy logic constructions from fuzzy sets for projects aimed at modeling human inference or developing an ISU. This work forms the basis for the development of a knowledge base with fuzzy logic based on linguistic variables. Thus, as an initial basis for

the development of intelligent control systems, classes that support fuzzy calculations and fuzzy logic were developed and with their help you can later design knowledge bases that guarantee the achievement of the set management goals.

Keywords: fuzzy numbers, linguistic variable, fuzzy inference, fuzzy logical operations.

For citation:
Zagibin N., Ulyanov S. Software implementation of fuzzy logic with linguistic variables. System Analysis in Science and Education, 2021;(1):45–57(In Russ). Available from: http://sanse.ru/download/426.

Введение

Сегодня развитие технологии в области имитирования процессов интеллектуальной деятельности головного мозга человека достигло невиданных возможностей. Распознавание лиц, обработка больших данных, создание роботов с системой обучения и адаптации к реальности, интеллектуальное управление процессами производства и систем автоматического регулирования – это лишь малый список достижений результатов эволюции научной деятельности в области интеллектуальных технологий. Помимо всего прочего данные технологии позволяют убрать из систем управления человеческий фактор и медленную скорость адаптации в меняющихся, а также критических ситуациях. Это в большей мере играет роль в управлении такими объектами как атомные электростанции или системы управления космическими аппаратами. Тем не менее мы можем видеть, как в повседневную жизнь вместе с умной техникой происходит внедрение интеллектуальной технологии, которая позволяет оптимизировать повседневные процессы и достигать поставленные цели за меньшее время несмотря на непредвиденные ситуации.

«Интеллектуализацию» в представленной технологии представляют множество вычислений, состоящих в общем виде из: мягких вычисления; эволюционного программирования; генетических алгоритмов и алгоритмов оптимизации типа иммунных алгоритмов; алгоритмов оптимизации на основе поведенческих реакций и обмена информацией активных агентов (самоорганизация целенаправленного и оптимального поведения в условиях «толпы»); квантовых генетических алгоритмов для глобальной оптимизации; квантовых нейронных сетей обучения и мн. др. [1]. В большинстве случаев для реализации автоматического управления прибегают к использованию ПИД (пропорционально интегрально-дифференциального) регулятора. Данная система управления составляет нижний (исполнительный) уровень с обратной связью в иерархии системы интеллектуального управления. В обычной системе с ПИД регулированием коэффициенты самого регулятора динамически не изменяются и не корректируются под внешние возмущения, их задают статически при создании системы. Для гибкой работы данной системы управления применяют нечеткий регулятор (НР) базирующийся на работе нечетких систем с нечеткой логикой (рис. 1) [2]. Данная логика позволяет приблизиться к модели процесса размышления человека при помощи лингвистических переменных и сделать соответствующие выводы на основании функций принадлежности в рассматриваемом классе понятий. Это позволит внедрить даже субъективные знания об объекте управления, которые впоследствии обработки вывода нечеткой логики обратятся в четкие значения для регулирования объектом управления.

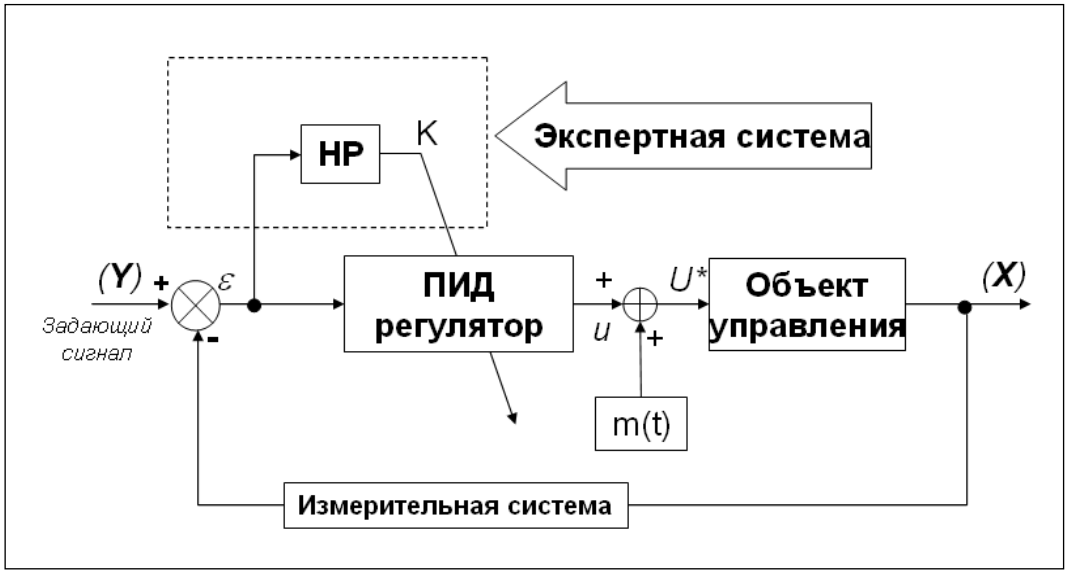

Рис. 1. Структура ИСУ первого поколения [1]

1. Актуальность

Потребность в сохранении ресурсов и извлечении большей экономической выгоды от использования и производства продукции сейчас как никогда раньше стоит перед человечеством в 21 веке. Еще в годы промышленной революции в большинстве случаев человеческий труд заменялся на машинный, сокращая время на производство. С приходом прогресса в электронной промышленности стало упрощаться и управление машин, где человек посредством заданных алгоритмов планировал их действие. Но сложность систем управления вместе со сложностью агрегатов возросла и как следствие в большинстве случаев стало невозможно описать динамическую модель объекта, откуда начали возникать непредвиденные ситуации управления и аварии. Для решения описанной проблемы достижения в непредвиденных ситуациях цели управления, адаптации в реальных условиях и самообучению применяется интеллектуальная система управления. Помимо управления данная система также позволяет снизить расход полезных ресурсов на исполнительном уровне. Как итог применения такой системы позволило добиться высокого уровня управления и экономической выгоды в отличии от их классических аналогов.

Для создания базы интеллектуальной систем управления (ИСУ), а также реализаций логических заключений имитируемые человеческое мышление (как пример нечеткое человеческое суждение «да нет, наверное») необходимы нечеткие системы. Нечеткие системы позволяют формировать базу нечетких логических заключений (базу знаний), используемую для обработки ввода и формирования управляющих (выходных) значений (как например управление ПИД-регулятором), как самой интеллектуальной системе, так и человеку эксперту, реализующему заданные принципы управления объектом [2]. В этом смысле нечеткий регулятор в интеллектуальном управлении является базовым элементом, в котором существует множество правил вывода объединённых в базу знаний, которая в последствии самоадаптируется за счет надстроечных элементов ИСУ для работы объекта управления (ОУ) в непредвиденных условиях. Но все это невозможно было бы представить без нечеткой логики, которая манипулирует нечеткими множествами сформированные в лингвистических переменных.

Программная реализация в виде подключаемой библиотеки позволит разработчикам создать свою базу знания нечеткой логики или нечеткие логические конструкции из нечетких множеств для проектов, направленных на моделирование человеческого умозаключения или разработки ИСУ.

2.Теоретическое представление лингвистической переменной и нечеткой логики

Лингвистическая переменная

Основой лингвистической переменной является нечеткое множество (понятие, введенное Л. Заде).

Определение [1]: нечеткое множество (*a fuzzy set*)

Пусть X есть некоторое универсальное множество (универсум). Тогда нечеткое множество A в X определяется как упорядоченное множество пар

$$A = \{(x, \mu(x)) | x \in X\},$$

где $\mu_A(x) \in [0,1]$ называется функцией принадлежности (ФП) элемента *x* к нечеткому множеству A.

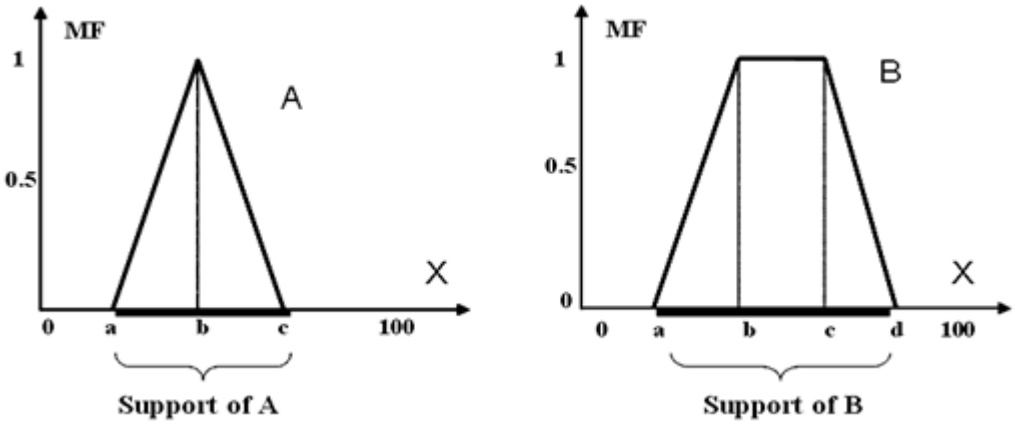

Рис. 2. Примеры нечетких множеств [1]

Определение [1]: лингвистическая переменная (ЛП) представляет собой следующую пятерку $\langle \chi, T(\chi), X, G, M \rangle$, где χ – имя переменной, $T(\chi)$ – терм-множество, задающее множество значений ЛП, являющихся языковыми выражениями (синтагмами), *X* – универсум, *G* – синтаксическое правило, используя которое мы можем формировать синтагмы $A, B, \ldots \in T(\chi)$, *M* – семантическое правило, используя которое каждой синтагме $A \in T(\chi)$ приписывается ее значение, являющееся нечетким множеством в универсуме *X*.

Приближенно можно сказать, что набор нечетких множеств, определенных на заданном универсуме X и принадлежащих логически к одному и тому же классу, является лингвистической переменной.

Нечеткие логические операции

Определение [1]: нечеткая конъюнкция (*Fuzzy conjunction* или *fuzzy AND*).

Значение истинности нечеткой конъюнкции $A \wedge B$ определяется следующим образом:

$$\mu_{A \wedge B}(x) = min(\mu_A(x), \mu_B(x)),$$

где $\mu_A(x), \mu_B(x)$ – значения истинности нечетких предикатов *A*, *B* соответственно.

Определение [1]: нечеткая дизъюнкция (*Fuzzy disjunction* или *fuzzy OR*). Значение истинности нечеткой конъюнкции $A \vee B$ определяется следующим образом:

$$\mu_{A\vee B}(x) = max\big(\mu_A(x), \mu_B(x)\big),$$

где $\mu_A(x), \mu_B(x)$ – значения истинности нечетких предикатов A, B соответственно.

Определение [1]: нечеткое отрицание (*Fuzzy negation*). Значение истинности нечеткого отрицания $\bar{A}$ определяется следующим образом:

$$\mu_{-_A}(x) = 1 - \mu_A(x),$$

где $\mu_A(x)$ – значение истинности нечеткого предиката A.

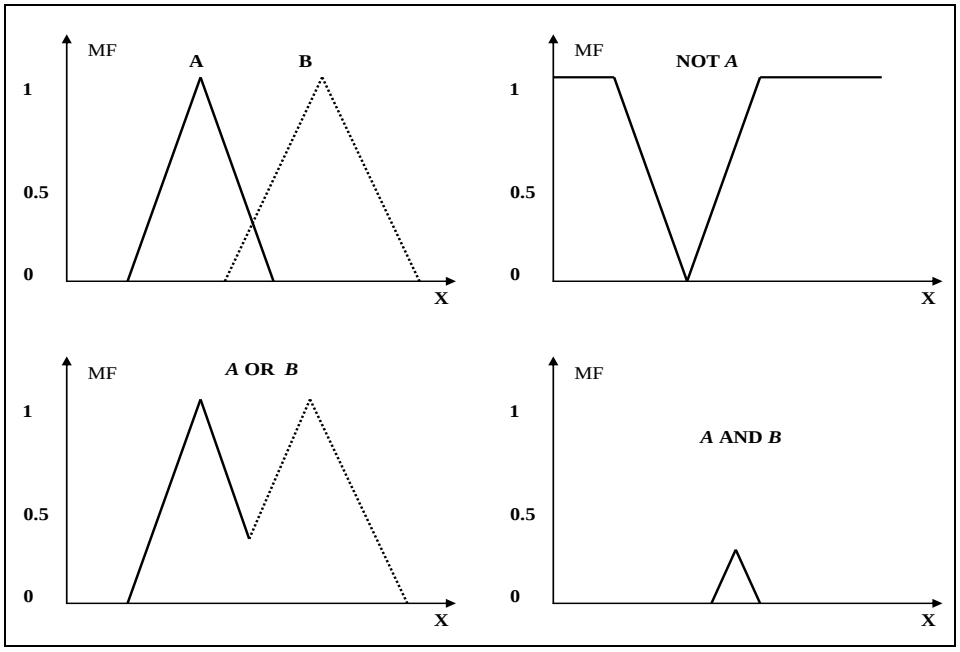

Рис. 3. Примеры нечетких операций над нечеткими множествами [1]

Нечеткая логика

Нечеткая логика базируется на выше приведённых логических операциях, построения правил и вывода из них.

Определение [1]: нечеткая импликация или нечеткое правило (*Fuzzy implication* или *a fuzzy rule*) представляет собой следующее выражение:

ЕСЛИ x есть A, ТО y есть B,

где A и B – лингвистические переменные, определенные нечеткими множествами на универсумах X и Y соответственно. В символьной форме запись имеет следующий вид: $R = A \to B$.

Часть «ЕСЛИ» (x есть A) называется антецедентом (*the antecedent*) или посылкой. Часть «ТО» (y есть B) называется следствием (*consequence*) или заключением.

Определение [1]: Нечеткая импликация (*Fuzzy implication*). Значение истинности нечеткой импликации $A \to B$ определяется следующим образом:

$$\mu_{A\to B}(x) = \mu_A(x) \wedge \mu_B(x)\ ,$$

где $\wedge$ есть операция нечеткой конъюнкции (*fuzzy AND operation*) и $\mu_A(x), \mu_B(x)$ – значения истинности нечетких предикатов A, B соответственно.

Нечеткий вывод, основанный на правиле *max-min* композиции

Такого типа вывод строится как «усечение» следствия от минимума из максимумов нечеткой конъюнкции посылок.

Рассмотрим следующий нечеткий вывод [1]:

Посылка 1 (нечеткий вход): x есть A'

Нечеткое правило: ЕСЛИ x есть A, ТО y есть B

Нечеткое заключение: y есть B'

Формула называется max-min композицией (max-min composition).

$$\mu_{B'}(y) = \max_x \min[\mu_{A'}(x), \mu_R(x,y)] = \bigvee_x [\mu_{A'}(x) \wedge \mu_R(x,y)].$$

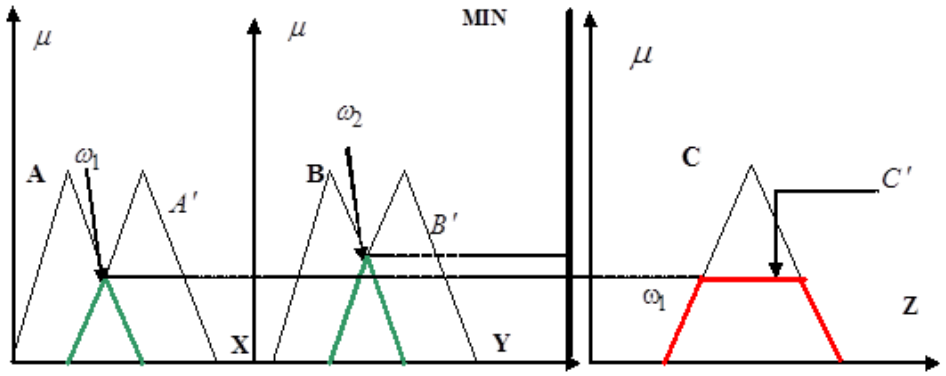

Рис. 4. Графическая интерпретация схемы нечеткого вывода на основе max-min композиции [1]

Нечеткая Модель Мамдани

В общем виде, четкое выходное значение в нечеткой модели Мамдани вычисляется по следующей формуле и вычисляется как дизъюнкция «усечений» следствий, построенных от минимального значения функции пригодности посылок правил:

$$F(x_1, \ldots, x_n) = \frac{\sum_{l=1}^{M} y^l \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}{\sum_{i=1}^{M} \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}.$$

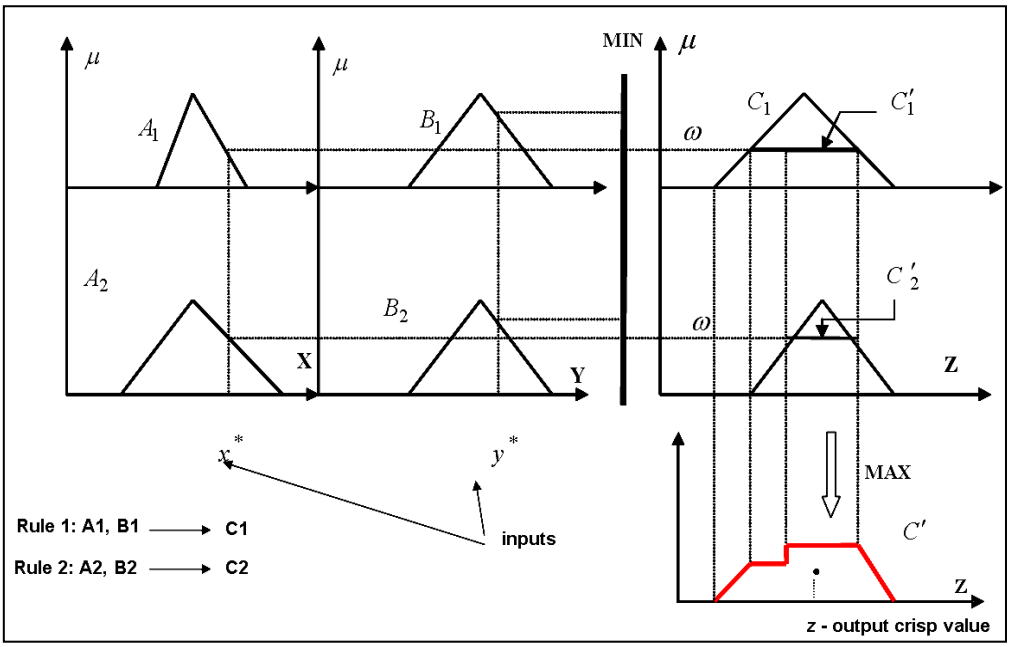

Рис. 5. Графическое представление нечеткого вывода в Мамдани модели [1]

Нечеткая Модель Сугено

В модели Сугено вывод определяется функцией от входных значений, где каждому правилу в соответствии ставится функция принимающая входные переменные и результатом модели считается сумма произведений данных функций, умноженных на минимальные значения функции пригодности посылок от выходных значений (активация правила) поделенная на сумму всех активаций правил.

$$F(x_1, x_2, \dots, x_n) = \frac{\sum_{l=1}^{M} f^l(x_1, x_2, \dots, x_n) \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}.$$

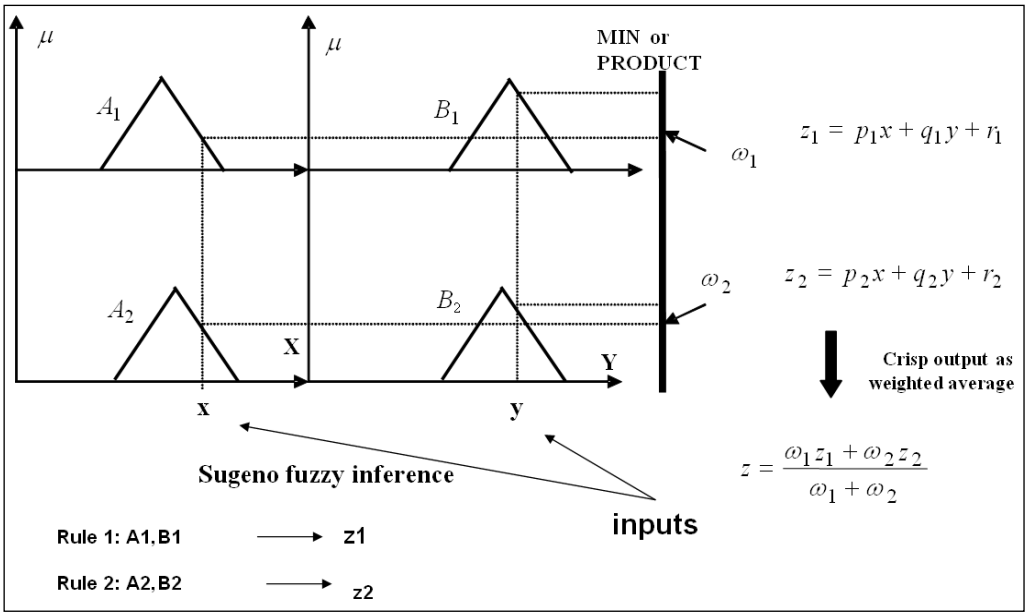

Рис. 6. Графическое представление нечеткого вывода в Сугено модели [1]

Нечеткая Модель Цукамото

В отличии от вышеперечисленных моделей нечеткого вывода в модели Цукамото используются следствии правил в виде монотонно убывающей или монотонно возрастающей функции пригодности нечеткого множества из которых находят значения аргумента данной функции по минимуму значения функции пригодности посылок (активация правил) и суммируя перемножения этих найденных двух значений делят на сумму активаций правил.

$$F(x_1, \ldots, x_n) = \frac{\sum_{l=1}^{M} z^l \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}{\sum_{i=1}^{M} \prod_{i=1}^{n} \mu_{j_i}^l(x_i)}.$$

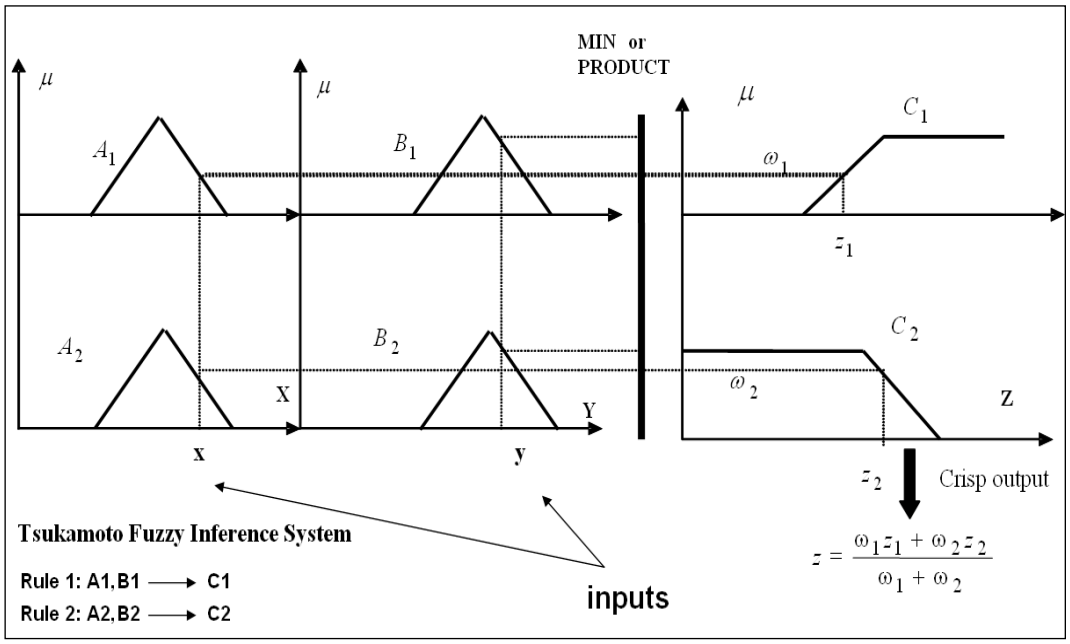

Рис. 7. Графическое представление нечеткого вывода в Цукамото модели [1]

3. Программная реализация лингвистических переменных и нечеткой логики

Программная реализация представляет собой два класса: *Linguistic_variable* и *Fuzzy_inference* (рис. 8), написанных на языке программирования *C++*.

Linguistic_variable – это класс создающий объекты лингвистических переменных, в основе которого лежит контейнер нечетких множеств *fuzzy_set* с их синтагмами (наименованиями). Его методы составляют основу нечетких логических операций (нечеткая конъюнкция, нечеткая дизъюнкция и нечеткое отрицание), добавление по точкам отрезков нечетких множеств и «отрисовку» данных множеств на элементе chart *WindowsForm*.

Fuzzy_inference – класс нечеткого вывода наследуемый от класса *Linguistic_variable*, реализующий в себе методы нечеткого вывода, основанный на правиле *max-min* композиции, нечеткие модели Мамдани, Сугено и Цукамото.

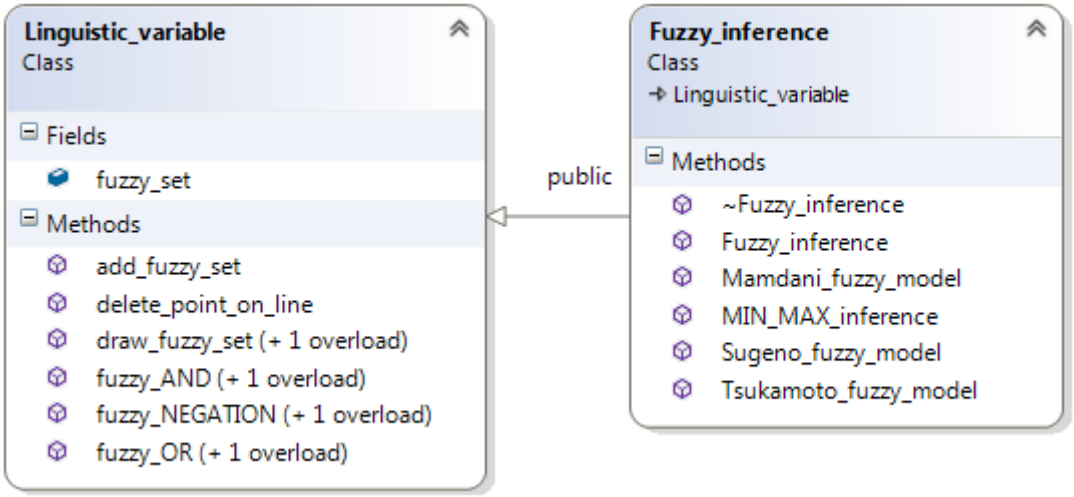

Рис. 8. Классы, составляющие нечеткую систему

Linguistic_variable

fuzzy_set в программой реализации представляет собой стандартный контейнер: *map<string, map<double,double>>*, где ключом представляется одно из терм множеств (наименование нечеткого множества) лингвистической переменной, а ключом внутреннего map является значение *x* из заданного набора нечеткого множества *A* в области универсального множества *X*. Значение ключа внутреннего map представляет функция пригодности $\mu_A(x) \in [0,1]$.

Чтобы не добавлять в нечеткое множество по одной точке в методе *add_fuzzy_set* можно указать наименование множества и указать необходимые точки:

Ling_var.add_fuzzy_set("*young*", { { 0, 1 }, { 40, 1 }, { 60, 0 } });

Для наглядного отображения на диаграмме потребуется всего лишь в методе *draw_fuzzy_set* перечислить интересующие наименования нечетких множеств:

Ling_var.draw_fuzzy_set(*chart1*, { "*young*", "*old*", "*old2*" });

fuzzy_AND принимает только два аргумента указывающих на множества и строит новое нечеткое множество на основе их нечеткой конъюнкции за счет вычисления пересечения прямых от двух точек и сравнения точки одного множества с её значением функции прямой от двух точек другого множества (рис. 9).

fuzzy_OR принимает также только два аргумента указывающих на множества и строит новое нечеткое множество на основе их нечеткой дизъюнкции за счет вычисления пересечения прямых от двух точек и сравнения точки одного множества с её значением функции прямой от двух точек другого множества (рис. 9).

fuzzy_OR принимает один аргумент указывающий на множество и строит новое нечеткое множество на основе нечеткого отрицания за счет вычитания из единицы функции принадлежности каждого аргумента нечеткого множества (рис. 9).

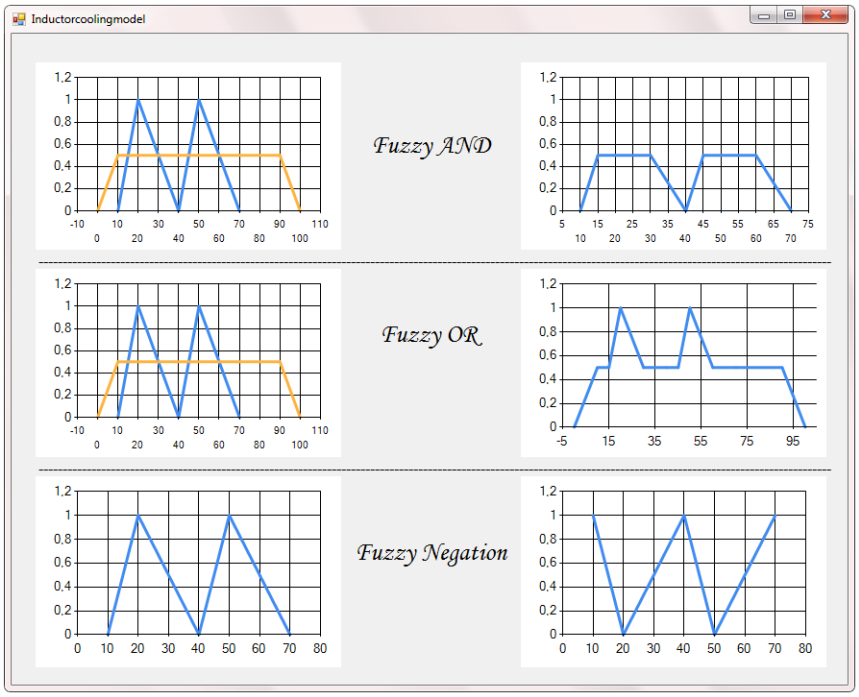

Рис. 9. Результаты нечетких логических операций

Fuzzy_inference

Чтобы использовать нечеткие логические операции и создавать собственные контейнеры для результатов было использовано наследование от класса Linguistic_variable. Здесь в аргументах функций используются двухмерные конструкции инициализаций правил где в обязательном порядке (кроме модели Сугено) необходимо чтобы количество «входных данных» было на 1 меньше количества перечисляемых посылок со следствием (т.к. они объединены). Как например вызов функции нечеткой модели Мамдани:

A_inference.Mamdani_fuzzy_model({28, 38}, {{&*Ling_var.fuzzy_set*["*young*"], &Ling_var.fuzzy_set["*old*"], &Ling_var.fuzzy_set["*old2*"]}, {&Ling_var.fuzzy_set["*old3*"], &Ling_var.fuzzy_set["*old4*"], &Ling_var.fuzzy_set["*old5*"] } });

MIN_MAX_inference – вывод *max-min* композиции реализует следующий алгоритм вывода, изображенный на рис. 10.

В модели Мамдани (*Mamdani_fuzzy_model*) входными данными являются значения x универсального множества лингвистической переменной. Алгоритм вывода приведен выше на рис. 10.

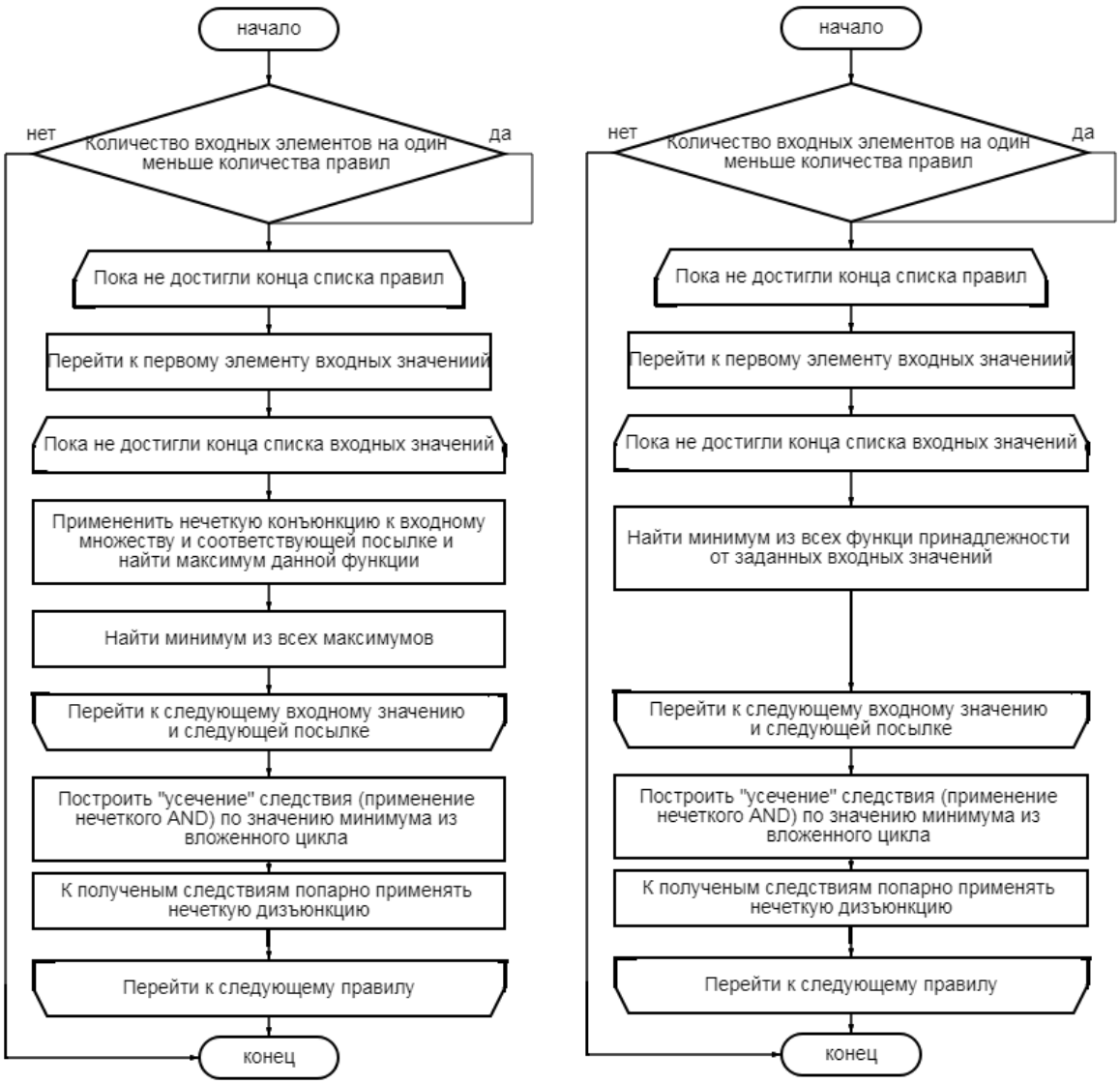

Рис. 10. Алгоритм вывода max-min композиции(слева) и алгоритм модели Мамдани (справа)

В модели Сугено (*Sugeno_fuzzy_model*) в качестве вывода используются сопоставленные в соответствие с правилами заданные функции, поэтому алгоритм теперь накапливает за циклические проходы сумму произведения минимальных значений на соответствующую функцию после чего в конце производится деление на сумму всех минимальных значений, где следующий алгоритм на рис. 11 отображает это.

В модели Цукамото (*Tsukamoto_fuzzy_model*) вместо двух первых моделей вывода используется монотонно возрастающая или монотонно убывающая функция пригодности в качестве следствия поэтому её алгоритм приобретает следующий вид (рис. 11).

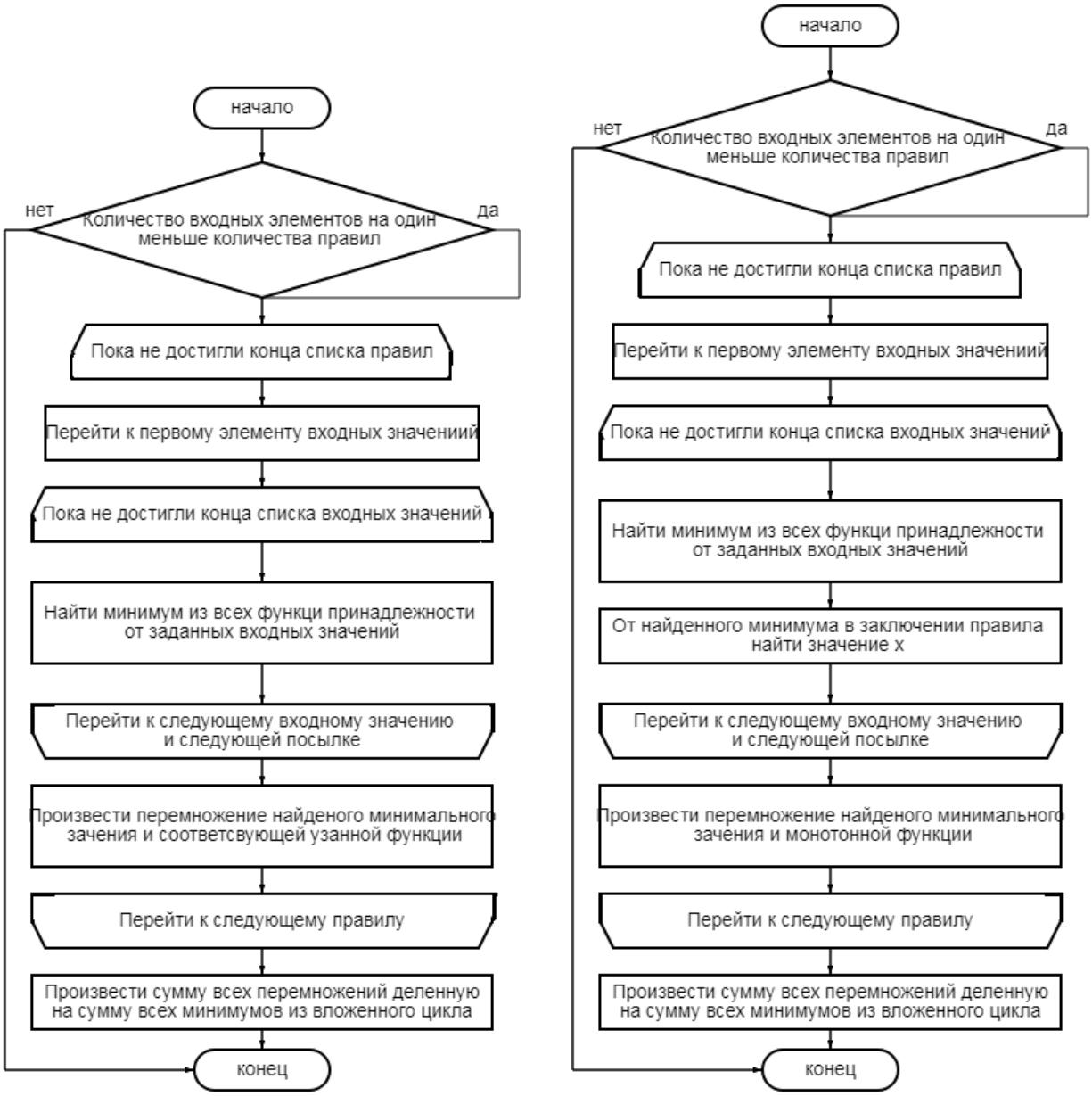

Рис. 11. Алгоритм модели Сугено (слева) и алгоритм модели Цукамото (справа)

Заключение

Без интеллектуальных технологий невозможно было бы представить развитие робототехники, систем автоматического управления и искусственного интеллекта применяемые в многочисленных отраслях общества, все это дает качественно новый подход к проектированию и созданию экономически привлекательных инструментов для управления процессами производства, техническими комплексами и машинами рационально перераспределяя и используя полезные ресурсы. Для реализации такого рода технологий необходимы средства, поддерживающие построение фундаментальных знаний об управляемом объекте. Таким образом в качестве начального базиса в области разработки интеллектуальных систем управления были разработаны классы поддерживающие нечеткие вычисления и нечеткую логику с помощью которых впоследствии можно проектировать базы знаний, гарантирующие достижения поставленных целей управления.

Список литературы

1. Ульянов С. В., Литвинцева Л. В., Добрынин В. Н., Мишин А. А. Интеллектуальное робастное управление: технологии мягких вычислений. М. : PronetLabs, 2011. 406 с.
2. Литвинцева Л. В., Ульянов С. В., Ульянов С. С. Проектирование робастных баз знаний нечетких регуляторов для интеллектуального управления существенно нелинейными динамическими системами. Ч. II // Изв. РАН, ТиСУ, 2006. № 5. С. 102–141.