МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних технологій

НЕЧІТКА ЛОГІКА ТА М'ЯКІ ОБЧИСЛЕННЯ

Навчальний посібник

Одеса: НУОП, 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних технологій

НЕЧІТКА ЛОГІКА ТА М'ЯКІ ОБЧИСЛЕННЯ

Для здобувачів 3 курсу бакалаврату денної форми навчання
Спеціальність: 126 - Інформаційні системи та технології
Освітня програма: Інформаційні системи та технології

Навчальний посібник

Затверджено
на засіданні кафедри
інформаційних технологій
Протокол №__ від __.__.2023 р.

Одеса: НУОП, 2024

Навчальний посібник з дисципліни " Нечітка логіка та м'які обчислення " для здобувачів бакалаврату спеціальності - 126 Інформаційні системи та технології освітньої програми «Інформаційні системи та технології» / Укл .: М.Д. Рудніченко, О.С. Потієнко. - Одеса: НУОП, 2024. - 61 с.

Укладачі: Рудніченко М. Д., к.т.н., Потієнко О.С., к.т.н.

ЗМІСТ

Лекція 1 «Поняття нечіткої логіки»

Математична теорія нечітких множин (нечітких множин) і нечеткая логіка (нечітка логіка) є узагальненнями класичної теорії множин і класичної формальної логіки. Дані поняття були вперше запропоновані американськими вченими Лотфі Заде (Lotfi Zadeh) у 1965 р. Основною причиною прояву нової теорії стало наявність нечетких і наближених розсудів при описі людини процесів, систем, об'єктів.

До того як нечеткий підхід до моделювання складних систем отримав знання у всьому світі, пройшло не одне десятиліття з моменту створення теорії нечетких множин. І на цьому шляху розвитку нечетких систем прийнято виділяти три періоди.

Перший період (кінець 60-х–початок 70 років) характеризується розвитком теоретичного апарату нечетких множин (Л. Заде, Е. Мамдані, Беллман). У другому періоді (70–80-е роки) з'являються перші практичні результати в області нечеткого управління складними технічними системами (парогенератор з нечетким управлінням). Одночасно стало розглядатися питання побудови експертних систем, заснованих на нечеткой логіці, розроблених нечетких контролерів. Нечеткие экспертные системы для поддержки принятия решений находят широкое применение в медицине и экономике.

Наконец, в третьому періоді, який длиться з кінця 80-х років і продовжується в даний час, з'являються пакети програм для побудови нечетких експертних систем, а області застосування нечеткой логіки помітно розширюються. Вона застосовується в автомобільній, аерокосмічній і транспортній промисловості, в області виробів побутової техніки, у сфері фінансів, аналізу та прийняття управлінських рішень і багатьох інших.

Триумфальне шествие нечеткой логіки по світу почалося після доказів у кінці 80-х Бартоломеем Коско знаменитої теореми FAT (Fuzzy Approximation Theorem). У бізнесі та фінансах нечетка логіка отримала знання після того, як у 1988 році експертна система на основі нечетких правил для прогнозування фінансових індикаторів єдина передбачила біржовий крах. І кількість успішних фазі-застосованих в даний час нараховується тисячами.

Порівняння звичайної та нечіткої логік наведено на рис.1.1.

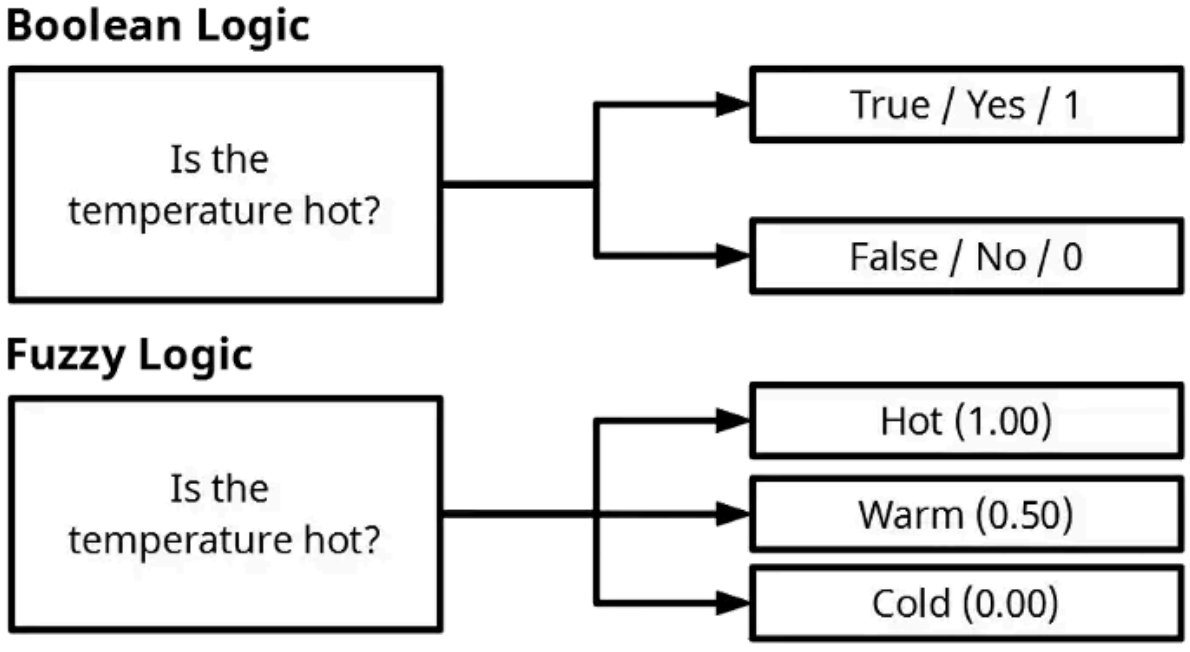


Рисунок 1.1 – Порівняння звичайної та нечіткої логік

Нечітка логіка (або теорія нечітких множин) є розділом математики, який дозволяє працювати з невизначеністю і нечіткістю. Ось деякі основні поняття нечіткої логіки:

1. Нечітка множина (Fuzzy Set).

Нечітка множина — це узагальнення класичного (чіткого) множини, в якому кожен елемент має ступінь атрибутів, що приймають значення в інтервалі від 0 до 1. Ступінь атрибутів описує нечіткість чи невизначеність.

2. Функція приналежності (функція членства).

Функція атрибутів визначає, наскільки елемент належить нечіткому множині. Ця функція може бути задана графічно і характеризує ступінь неозначеності.

3. Невизначеність (Невизначеність).

Нечітка логіка допомагає моделювати і працювати з невизначеністю в даних. Це особливо корисно в тих випадках, коли точні значення важко визначити чи піддається суб'єктивному сприйняттю.

4. Оператори нечіткої логіки.

Нечітка логіка надає свої власні оператори, такі як оператори нечіткого об'єднання (або), нечіткого перетину (і), нечіткого заперечення і т.д. Ці оператори застосовуються до функцій функцій.

5. Нечіткий висновок (Fuzzy Inference).

Нечіткий висновок використовується для прийняття рішень на основі правил, у яких використовуються нечіткі умови. Це часто застосовується у нерівній логіці для побудови нечітких систем управління.

6. Лінгвістичні зміни та терміни.

У нечіткій логіці використовуються лінгвістичні змінні і терміни, які являють собою якісні описи, такі як «маленьке», «середнє», «велике». Вони служать для формалізації розумів і виведення у нечіткій системі.

7. Нечіткі системи управління (Fuzzy Control Systems).

Нечіткі системи управління використовують нечітку логіку для моделювання та управління системами, особливо в умовах невизначеності та за наявності нечітких вхідних даних.

8. Невизначеність та розпливчастість (Невизначеність).

Важливі поняття, пов'язані з нечіткою логікою, включають невизначеність (коли невідомі точні значення) та розпливчастість (коли межі між категоріями нечіткі).

Ці основні поняття невизначеної логіки широко використовують у різних аспектах, таких як штучний інтелект, управління системами, обробка сигналів та прийняття рішень.

Ось деякі важливі характеристики нечіткої логіки:

- Гнучкість та простота реалізації навчання за допомогою машини техніка
- Допомагає імітувати логіку людського мислення.
- Логіка може мати два значення, які представляють два можливі рішення.
- Дуже підходящий метод для невизначених чи приблизних міркувань.
- Нечітка логіка розглядає висновок процес поширення пружних обмежень.
- Нечітка логіка дозволяє будувати нелінійні функції довільної ком.
- Нечітка логіка має створюватися під повним керівництвом експертів.

Проте нечітка логіка ніколи не є панацеєю від усіх. Тому не менш важливо розуміти, де не слід використовувати нечітку логіку.

Ось певні ситуації, коли краще не використовувати Fuzzy Logic:

- Якщо вам не зручно зіставляти простір введення з простором виводу.
- Нечітку логіку не слід використовувати, якщо ви можете керуватися здоровим глуздом.
- Багато контролерів можуть чудово справлятися зі своїм завданням без використання нечіткої логіки.

Архітектурна схема системи на базі нечіткої логіки наведена на рис.1.2. Архітектура Fuzzy Logic складається з чотирьох основних частин:

1. База правил. Вона містить усі правила та умови «якщось», які пропонують експерти для управління системою прийняття рішень. Нещодавнє оновлення нечіткої теорії пропонує різні методи проектування та налаштування нечітких контролерів. Ці оновлення значно скорочують кількість нечіткого набору правил.

2. Фазифікація. Крок фазифікації допомагає перетворити вхідні дані. Він дозволяє перетворювати чіткі числа на нечіткі множини. Точні вхідні дані вимірюються датчиками і передаються в систему керування для подальшої обробки. Наприклад, кімнатна температура, тиск тощо.

3. Механізм логічного висновку. Це допоможе визначити ступінь відповідності нечітких вхідних даних до правил. На основі збігу в % він визначає, які правила необхідно реалізувати відповідно до даного поля введення. Після цього застосовувані правила об'єднуються розробки керуючих впливів.

4. Дефаззифікація. Нарешті виконується процес дефазифікації для перетворення нечітких множин у чітке значення. Існує безліч типів методів, тому вам необхідно вибрати той, який найкраще підходить для використання із експертною системою.

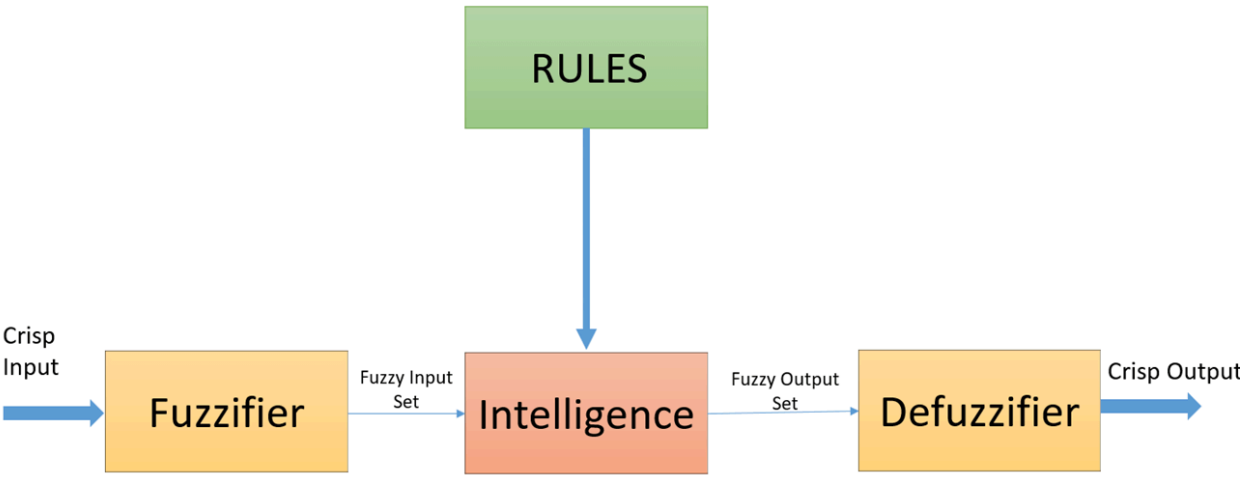


Рисунок 1.2 – Архітектурна схема системи на базі нечіткої логіки

Переваги нечіткої логіки:

– Ця система може обробляти вхідні дані будь-якого типу, включаючи безладні, невпорядковані або неправильні дані.

– Системи нечіткої логіки прості для створення та розуміння.

– Нечітка логіка базується на математичних принципах теорії множин, і міркування досить прості для розуміння.

– Він є дуже ефективним рішенням складних проблем у всіх аспектах нашого життя, оскільки імітує людське мислення та прийняття рішень.

– Для опису алгоритмів необхідна обмежена кількість інформації, яка потребує обмеженого простору для зберігання.

– Користувачі можуть використовувати його для регулювання побутової техніки та інших продуктів.

– Ви можете використовувати недорогі датчики, що зменшує вартість і складність всієї системи.

Недоліки нечіткої логіки:

– Системи з нечіткою логікою мають повільний час виконання та повільно генерують результати.

– Нечітка логіка в штучному інтелекті може бути непридатною для ситуацій, що вимагають високої точності.

– Нечітка система, заснована на знаннях, вимагає значного тестування обладнання для підтвердження та валідації.

– Оскільки нечітка логіка використовує точні та неточні дані, точність часто може бути знижена.

У загальному випадку механізм нечіткої логіки містить наступні 4 складові: фазифікація, нечіткий висновок, композиція, та дефазифікація (рис.1.3).

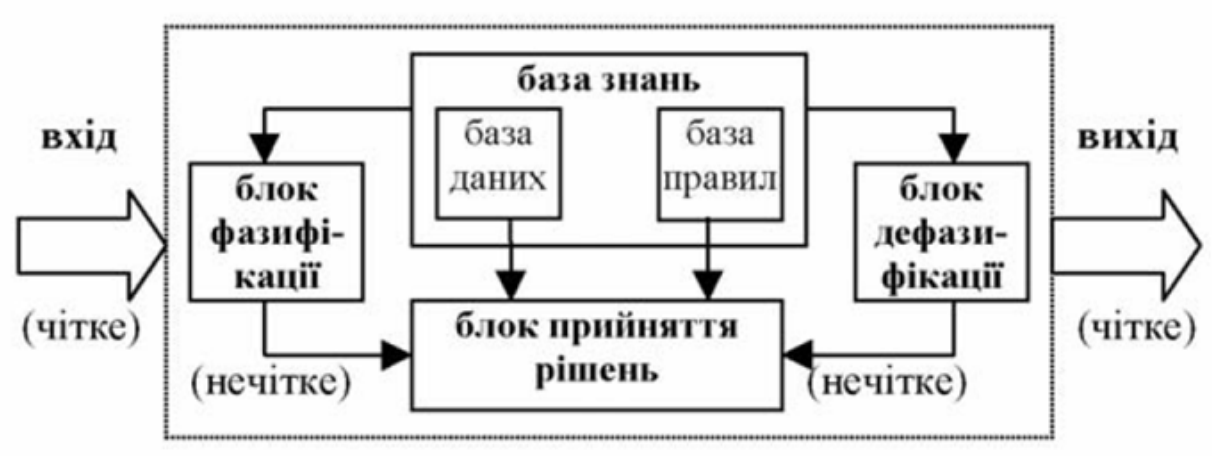


Рисунок 1.3 – Схема компонентів системи НЛВ

Лекція 2 «Нечіткі множини»

Нечіткою безліччю називається деяка, цілком певна сукупність об'єктів довільної природи, званих елементами множини, щодо яких стверджувати, належить той чи інший елемент аналізованої сукупності чи ні можна лише з певною часткою визначеності.

Позначення. НМ позначатимемо великими літерами латинського алфавіту A, B, C і т.д., елементи НМ - малими: a, b, c.

Для певності точні множини будемо позначати великими літерами латинського алфавіту: [A, B, C, ...), а елементи множин - рядковими: (a, b, c, ...).

Нечітка множина є розширенням класичної (чіткої) множини, і її характеристики включають наступні аспекти:

1. Елементи нечіткої множини. Як і в точних множинах, нечітка множина також складається з елементів. Однак кожен елемент нечіткої множини має ступінь приналежності, яка є числом в інтервалі від 0 до 1, що вказує на ступінь приналежності елемента множині.

2. Функція приналежності (Membership Function). Ключовою характеристикою нечіткої множини є функція приналежності, яка визначає ступінь належності кожного елемента множини до нього. Функція приналежності зазвичай є графік, де по осі X відображаються елементи множини, а по осі Y - їх ступеня приналежності.

3. Розмитість. Вона дозволяє врахувати невизначеність даних, а також враховувати різні рівні чіткості в описі елементів множини.

4. Операції на нечітких множинах. Нечіткі множини підтримують операції, аналогічні операціям над точними множинами, такі як об'єднання, перетин та доповнення. Однак у контексті нечітких множин ці операції виконуються лише на рівні функцій власності.

5. Нечіткі відносини. У нечіткій логіці також використовуються нечіткі відносини між елементами множин. Ці відносини можуть бути використані для вираження невизначеності у відносинах між елементами.

6. Лінгвістичні терми. Нечіткі множини часто описуються лінгвістичними термами, такими як "маленький", "середній", "великий". Ці терми допомагають абстрагувати значення і роблять опис нечіткої множини зрозумілішим для людини.

Таким чином нечітка множина - це безліч пар <m(x)/x>, де x приймає деяке інформативне значення, а m(x) відображає x в одиничний відрізок, приймаючи значення від 0 до 1. При цьому m(x) є ступенем належності x до чогось (0 - не належить, 1 - належить на всі 100%).

Так, наприклад, можна задати для числа 7 безліч: <0/1>,<0.4/3>,<1/7>

Це безліч говорить про те, що 7 – це на 0% одиниця, на 40% трійка та на 100% сімка.

Нечітка змінна визначається як <A, X, Ca>.

A - найменування змінної,

X={x} - область визначення змінної, набір можливих значень x,

Ca={<Ma(x)/x>} - нечітка множина, що описує обмеження на можливі значення змінної A (семантику).

Приклад: <"Сім",{1,3,7},{<0/1>,<0.4/3>,<1/7>}>.

Цим записом ми визначили відповідність між словом та деякими цифрами. Причому як у назві змінної, так і в значеннях x можна було використовувати будь-які записи, що несуть будь-яку інформацію.

Лінгвістична змінна визначається як <B,T,X,G,M>.

B – найменування змінної.

T - безліч її значень (базове терм-множина), складається з найменувань нечітких змінних, областю визначення кожної з яких є безліч X.

G - синтаксична процедура (граматика), що дозволяє оперувати елементами терм-множини T, зокрема - генерувати нові осмислені терми. T`=TUG(T) задає розширену терм-множину (U - знак об'єднання).

M - семантична процедура, що дозволяє приписати кожному новому значенню лінгвістичної змінної нечітку семантику, шляхом формування нової нечіткої множини.

Нечітка безліч (або нечітке число), описує деякі поняття в функціональному вигляді, тобто такі поняття як "приблизно рівно 5", "швидкість трохи більше 300 км/год" і т. д., як видно ці поняття неможливо уявити одним числом, хоча в реальності люди дуже часто користуються ними.

Нечітка змінна це теж саме, що і нечітке число, тільки з додаванням імені, яким формується поняття описуване цим числом.

Лінгвістична змінна - це безліч нечітких змінних, вона використовується для того щоб дати словесний опис деякому нечіткому числу, отриманому в результаті деяких операцій. Шляхом деяких операцій підбирається найближче за значенням з лінгвістичної перемінної.

Нечіткі числа краще зберігати як відсортований безліч пар (згортається по носіях), за рахунок цього можна прискорити виконання всіх логічних і математичних операцій. Коли реалізуємо арифметичні операції, то потрібно враховувати похибку обчислень, тобто 2/4 <> 1/2 для комп'ютера. Носії в нечітких числах повинні бути короткими будь-якому числу, інакше результати ариф. операцій буде "некрасивими", результат буде неточним, особливо це видно при множенні.

За рахунок зберігання нечітких чисел у відсортованому вигляді арифметичні операції у виконуються за майже лінійною залежністю (у часі), тобто при збільшенні кількості пари, швидкість обчислень падала лінійно.

Розглянемо простий приклад нечіткої множини, пов'язаної з температурою. Допустимо, у нас є безліч "Температура повітря", і ми хочемо описати поняття "тепло". Замість чіткого визначення, ми будемо використовувати нечітку множину та функцію приналежності.

Безліч "Температура повітря":

Це безліч включає всі можливі значення температури повітря, наприклад, від -10 до 40 градусів Цельсія.

Функція приладдя "Тепло":

Припустимо, що хочемо визначити, що для нас "тепло". Ми можемо використовувати функцію приладдя, яка відображатиме кожне значення температури у міру приналежності до поняття "тепло". Наприклад:

Якщо температура дорівнює 20°C, рівень приналежності до поняття "тепло" може бути 0.8.

Якщо температура дорівнює 10°C, рівень належності може бути 0.5.

Якщо температура дорівнює 30°C, рівень належності може бути 0.9.

Графічно функція приналежності може виглядати, наприклад, як трикутник з піком у точці, де температура вважається ідеально "теплою", а бічні сторони визначають, наскільки температура відхиляється від цієї ідеальної точки. Графічно це може виглядати наступним чином (рис.2.1).

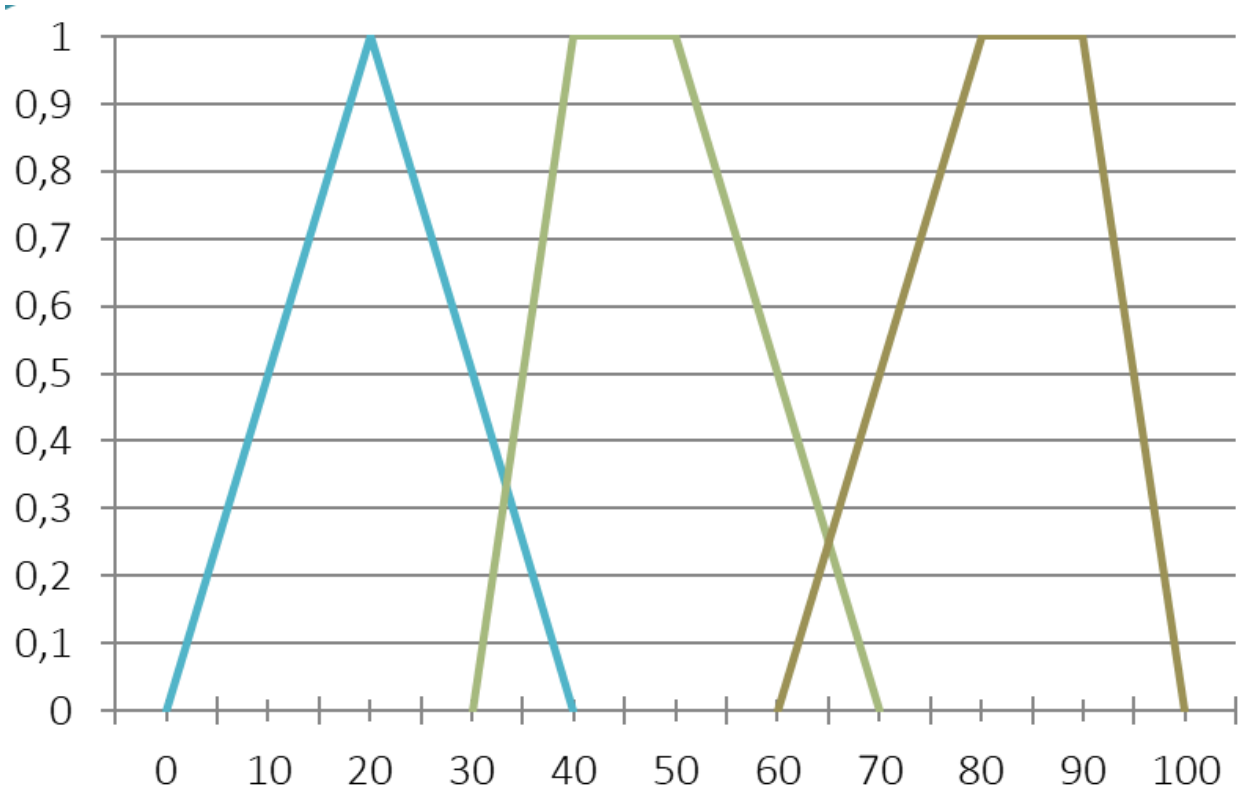


Рисунок 2.1 – Приклад візуалізації функцій приналежності

Таким чином, безліч "Температура повітря" з поняттям "тепло" є прикладом нечіткої множини, де чітко визначена межа "теплоти" відсутня, і кожне значення температури має свій ступінь приналежності до поняття "тепло".

Потужністю НМ, що позначається A, називається кількість елементів НМ.

Кінцевими називаються НМ, що мають кінцеву потужність носія.

Нескінченними називаються НМ, що мають нескінченний носій.

Дві множини називаються рівносильними, якщо між елементами цих множин встановлено взаємно однозначну відповідність.

Рахунковим НМ називається НМ, носій якого рівномірний N, тобто елементи носія можна пронумерувати.

Континуальним називається НМ, носій якого рівномірний R.

Розглянемо ряд популярних функцій приналежності, що часто характеризують різні нечіткі множини.

Шматково-лінійні функції власності. Як перший тип функцій приналежності розглянемо функції, які, як випливає з їхньої назви, складаються з відрізків прямих ліній, утворюючи безперервну або шматково-безперервну функцію. Найбільш характерним прикладом таких функцій є "трикутна" та "трапецієподібна" (рис. 3.1) функції приналежності. У нашому випадку кожна з цих функцій задана на універсумі X = [0, 10], як обраний замкнутий інтервал дійсних чисел. У випадку вибір універсуму може бути довільним, і обмежений жодними правилами.

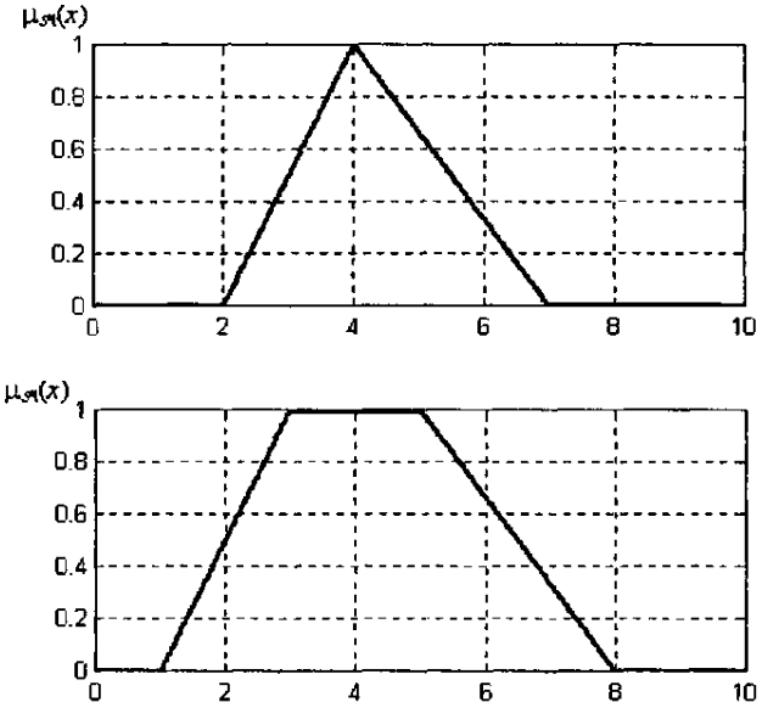


Рисунок 3.1 – Графіки трикутної та трапеціїдальної функцій приналежності

Перша з цих функцій приналежності в загальному випадку може бути задана наступним аналітично виразом:

$$f_{\Delta}(x;,a,b,c) = \begin{Bmatrix} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le c \le x \\ 0, & c \le x \end{Bmatrix}, \tag{3.1}$$

де a, b, c — деякі числові параметри, що набувають довільних дійсних значень і впорядковані ставленням $a \le b \le c$. Як можна помітити, ця функція приналежності породжує нормальну опуклу унімодальну нечітку множину з носієм — інтервалом (a, c), межами (a, c) \{b}, ядром {b} та модою b.

Трапецієподібна функція приналежності у загальному випадку може бути задана аналітично таким виразом

$$f_T(x;, a, b, c, d) = \begin{Bmatrix} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \frac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{Bmatrix}, \quad (3.2)$$

де a, b, c, d - деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням $a \le b \le c \le d$:

Як неважко помітити, параметри a і d характеризують нижню основу трапеції, а параметри b і c - верхня основа трапеції. При цьому дана функція приналежності породжує нормальну опуклу нечітку множину з носієм - інтервалом (a, d), межами (a, b) (c, d) та ядром [b, c]. Ці функції використовуються для завдання таких властивостей множин, які характеризують невизначеність типу: "приблизно одно", "середнє значення", "розташований в інтервалі", "подібний до об'єкта", "схожий на предмет" та ін. Вони також служать, для подання нечітких чисел та інтервалів.

Z-подібні та S-подібні функції приналежності отримали свою назву на вигляд кривих, які представляють їх графіки. Перша з функцій цієї групи називається Z-подібною кривою або сплайн-функцією і може бути задана аналітично наступним виразом

$$f_{z_1}(x;, a, b) = \begin{Bmatrix} 1, & x \le a \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), & a \le x \le b \\ 0, & x > b \end{Bmatrix}, \quad (3.3)$$

де a, b — деякі числові параметри, що набувають довільних дійсних значень і впорядковані ставленням a<b.

Графік цієї функції для деякої нечіткої множини та універсуму X=[0, 10] зображено на рис. 3.2 а, при цьому значення параметрів відповідно дорівнюють a=3, b=6.

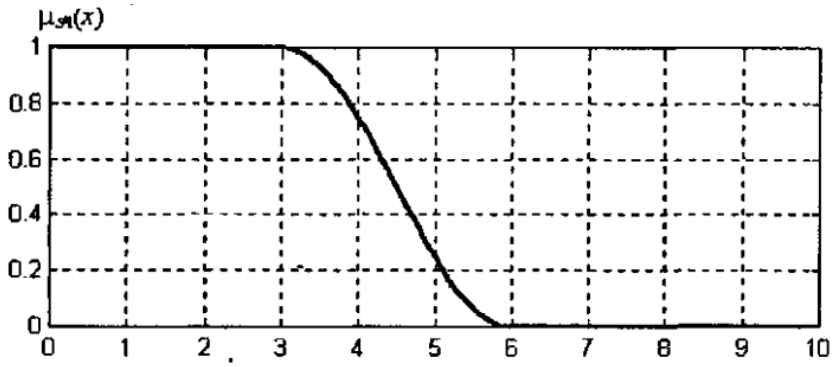


Рисунок 3.2 – Графік Z-функції

Z-функції використовуються для представлення таких властивостей нечітких множин, які характеризуються невизначеністю типу: "мала кількість", "невелике

значення", "незначна величина", "низька собівартість продукції", "низький рівень цін або доходів", "низька процентна ставка" та багато інших. Спільним для всіх таких ситуацій є слабкий ступінь прояву тієї чи іншої якісної чи кількісної ознаки. Особливість нечіткого моделювання при цьому полягає у поданні відповідних нечітких множин за допомогою незростаючих (монотонно спадних) функцій приналежності.

Друга з функцій групи, що розглядається, називається S-подібною кривою або сплайн-функцією і в загальному випадку може бути задана аналітично наступним виразом

$$\boldsymbol{f_{S1}(x;,a,b)} = \begin{Bmatrix} 0, & x < a \\ \frac{1}{2} + \frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right), & a \le x \le b \\ 1, & x > b \end{Bmatrix}, \tag{3.4}$$

де a, b — деякі числові параметри, що набувають довільних дійсних значень і впорядковані ставленням: a<b. Графік цієї функції для деякої нечіткої множини та універсуму X=[0, 10] зображений на рис. 3.3.

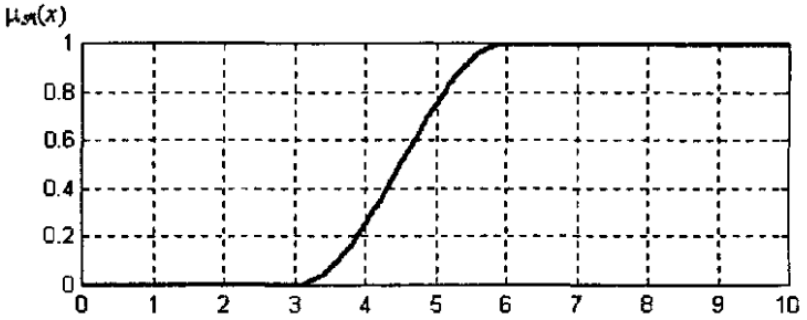


Рисунок 3.2 – Графік S-функції

До типу S-подібних та одночасно Z-подібних функцій приналежності може бути віднесена так звана сигмоїдальна функція (сигмоїд), яка в загальному випадку задається аналітично наступним виразом

$$f_{S3}(x;a,b) = \frac{1}{1+e^{-a(x-b)}}, \tag{3.5}$$

де a, b - деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням a <b: a e - основа натуральних логарифмів, яке ініціює завдання відповідної експоненційної функції. При цьому у випадку a>0 може бути отримана S-подібна функція приналежності, а у випадку a<0 Z-подібна функція приналежності.

Розглянуті S-подібні функції використовуються для представлення таких нечітких множин, які характеризуються невизначеністю типу: "велика кількість", "велике значення", "значна величина", "високий рівень доходів та цін", "висока норма прибутку", "висока якість послуг" ", "високий сервіс обслуговування" та багатьох інших. Спільним для всіх таких ситуацій є високий ступінь прояву тієї чи іншої якісної чи кількісної ознаки. Особливість нечіткого моделювання при цьому полягає у поданні відповідних нечітких множин за допомогою неубутніх (монотонно зростаючих) функцій приналежності.

Базові операції над нечіткими множинами.

1. Об'єднання: створюється нове безліч елементів вихідних множин, причому для однакових елементів належність береться максимальною.

A U B = {<Maub(x)/x>}

Maub(x) = max {Ma(x), Mb(x)}

2. Перетин: створюється нова множина з однакових елементів вихідних множин, належність яких береться мінімальною.

A П B = {<Maпb(x)/x>}

Maпb(x) = min {Ma(x), Mb(x)}

3. Додаток: інвертується належність кожного елемента.

C = ~A = {<Mc(x)/x>}

Mc(x) = 1-Ma(x)

4. Ступень: належність кожного елемента зводиться у ступінь.

CON - концентрація, ступінь = 2 (зменшує ступінь нечіткості)

DIN - розтягування, ступінь = 1/2 (збільшує ступінь нечіткості)

5. Різниця: нова множина складається з однакових елементів вихідних множин.

A - B = {<Ma-b(x)/x>}

Ma-b(x) = Ma(x)-Mb(a), якщо Ma(x)>Mb(x) інакше 0

6. Носій: складається з елементів вихідної множини, приналежності яких більше нуля.

Supp(A) = {x|x?X /\ Ma(x)>0}

7. Примноження на число: приналежності елементів примножуються на число.

q*A = {<q*Ma(x)/x>}

8. Супремум: Sup - точна верхня грань (максимальне значення приналежності, що є у множині).

9. Нормалізація: нечітка множина нормально якщо супремум множини дорівнює одиниці. Для нормалізації перечитують приналежність елементів:

M'a(x) = Ma(x)/(Sup Ma(x))

10. Альфа-зрез: безліч альфа рівня - ті елементи вихідної множини, приналежність яких вище або дорівнює заданому порогу. Поріг, що дорівнює 1/2, називають точкою переходу.

Aq = {x|x?X /\ Ma(x)>q}

11. Нечітке включення: ступінь включення нечіткої множини

V(A1,A2) = (Ma1(x0)->Ma2(x0))&(Ma1(x1)->Ma2(x1))&..

За Лукасевичем:

Ma1(x)->Ma2(x) = 1&(1-Ma1(x)+Ma2(x))

По Заде:

Ma1(x)->Ma2(x) = (1-Ma1(x)) \/ Ma2(x)

12. Нечітка рівність: ступінь нечіткої рівності

R(A1,A2) = V(A1,A2) & V(A2,A1)

У класичної чіткої (двозначної) логіці існують логічні операції «І», «АБО», «НЕ», які визначаються єдиним чином і вони становлять повні системи, тобто. будь-яке рівняння чіткої логіки шляхом логічних перетворень може бути виражене у вигляді логічних комбінацій:

"не-і-або";

"не-і";

"не-або".

Одне із завдань теорії нечітких множин полягає у узагальненні чітких логічних операцій на їх нечіткі аналоги. Нечітким розширенням операції "і" у загальній формі є T

або тріангулярна норма, яка в теорії нечітких множин позначається символом (Т). Іншою назвою Т-норми є S-конорма.

Ця операція визначається як відображення:

$$\mu_{A_1}(x)(T)\mu_{A_2}(x) \to \mu_{A_3}(x);$$
$$\mu_{A_1}(x) \in [0;1],\ \mu_{A_2}(x) \in [0;1],\ \mu_{A_3}(x) \in [0;1],$$

для якого виконуються аксіоми:

$$\begin{aligned}
&1.\mu_{A_1}(x)(T)\{\mu_{A_2}(x)=1\}=\{\mu_{A_3}(x)=\mu_{A_1}(x)\},\quad \forall\mu_{A_1}(x)\in[0;1]\\
&2.\mu_{A_1}(x)(T)\{\mu_{A_2}(x)=0\}=\{\mu_{A_3}(x)=0\},\quad \forall\mu_{A_1}(x)\in[0;1]\\
&3.\mu_{A_1}(x)(T)\mu_{A_2}(x)=\mu_{A_2}(x)(T)\mu_{A_1}(x)\ ;\\
&4.\mu_{A_1}(x)(T)\{\mu_{A_2}(x)(T)\mu_{A_3}(x)\}=\{\mu_{A_1}(x)(T)\mu_{A_2}(x)\}(T)\mu_{A_3}(x);\\
&5.\mu_{A_1}(x)\le\mu_{A_2}(x)\to\mu_{A_1}(x)(T)\mu_{A_3}(x)\le\mu_{A_2}(x)(T)\mu_{A_3}(x).
\end{aligned} \qquad (4.1)$$

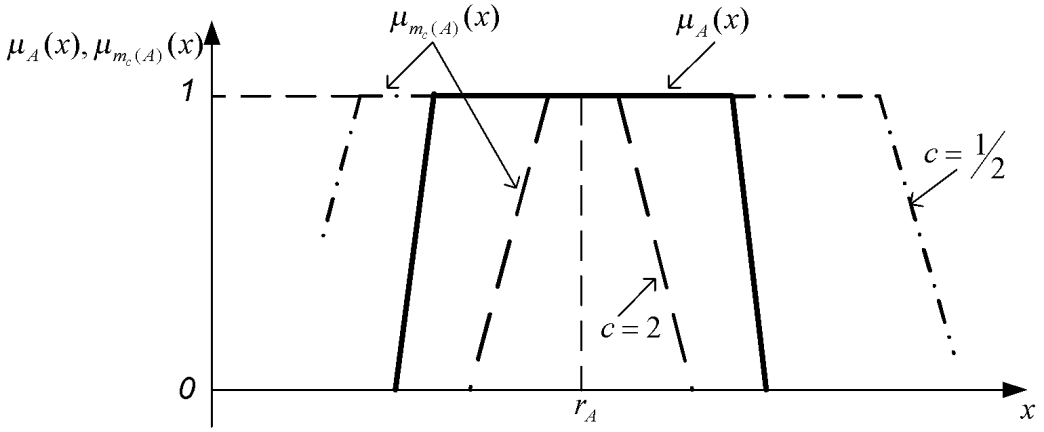


Рисунок 4.1 - Нечітка множина (тіло), модифіковані нечіткі множини «дуже( )»(---,) і «більше або менше( )» (-·-·-·-,), отримані шляхом зміни масштабу щодо точки

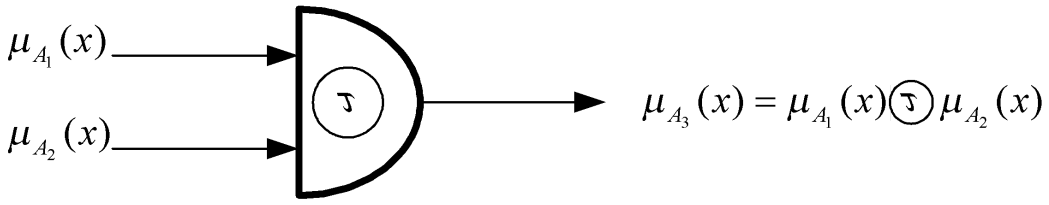


Рисунок 4.2 – Схемотехнічне подання Т-норми

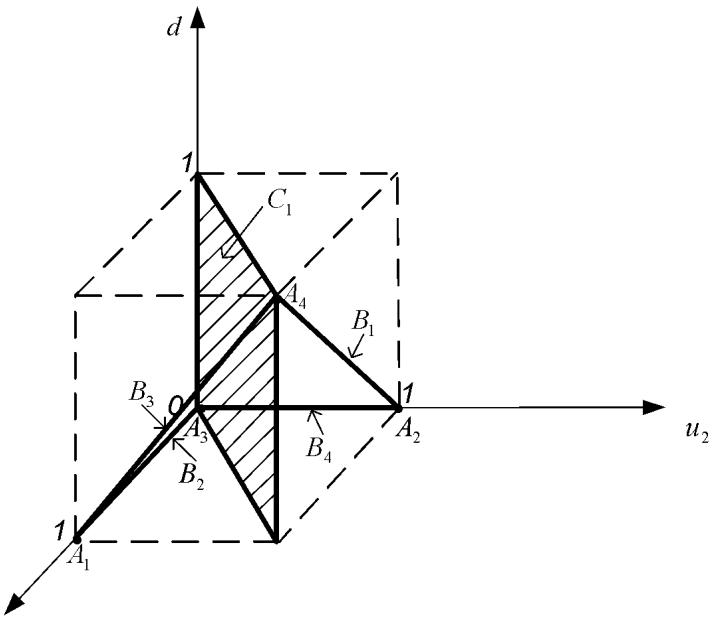


Рисунок 4.3 – Граничні умови Т-норми

Перші 2 аксіоми у (4.1) називаються аксіомами граничних умов Т-норми; друга пара – об'єднання чи перетину, остання - упорядкованості.

На схемотехнічному рівні операція (Т) реалізується у вигляді схеми з двома входами та одним виходом. Аксіоми означають, що входи є рівнозначними і немає необхідності їх розрізняти. З позицій традиційної математики операція (Т) реалізує функцію двох змінних:

y=T(u_1,u_2), де $u_1 = \mu_{A_1}(x), u_2 = \mu_{A_2}(x), y = \mu_{A_3}(x).$

Ця функція «Т» у тривимірному просторі (u1,u2,y) зображує деяку поверхню. Геометрична фігура, побудована відповідно до аксіом граничних умов, дає можливість визначити максимальні та мінімальні значення Т-норми, як функції двох змінних. При її побудові величин $\mu_{A_1}(x), \mu_{A_2}(x).$ надають різні значення, що належать відрізку [0;1], і відповідно визначають значення що в системі координат (u1,u2,y) дає відповідну сукупність точок.

$$u_1 = 1,\ u_2 = 0 \ \rightarrow\ y\Big/_{\substack{u_1=1\\u_2=0}} = 1(T)0 = 0.$$

Це зображує точку A_1 з координатами (u_1=1; u_2=0; y=0). Після аналогічних обчислень виходять координати точок A_2-A_4:

$$y\Big/_{\substack{u_1=0\\u_2=1}} = 0(T)1 = 0 \ \rightarrow\ (0;1;0) = A_2;$$

$$y\Big/_{\substack{u_1=0\\u_2=0}} = 0(T)0 = 0 \ \rightarrow\ (0;0;0) = A_3;$$

$$y\Big/_{\substack{u_1=1\\u_2=1}} = 1(T)1 = 1 \;\rightarrow\; (1;1;1) = A_4.$$

Сукупність точок A_2-A_4 визначають вершини одиничного куба. При зміні u_1 від 0 до 1 і фіксованому u_2=1 отримаємо рівняння прямої B_1:

$$y\Big/_{\substack{u_1\in[0;1]\\u_2=1}} = u_1(T)1 = u_1 .$$

Далі аналогічно для прямих B_2-B_4:

$$y\Big/_{\substack{u_1\in[0;1]\\u_2=0}} = u_1(T)0 = 0.$$

- рівняння прямої B_2

$$y\Big/_{\substack{u_1=1\\u_2\in[0;1]}} = 1(T)u_2 = u_2 .$$

- рівняння прямої B_3

$$y\Big/_{\substack{u_1=0\\u_2\in[0;1]}} = 0(T)u_2 = 0$$

- рівняння прямої B_4.

$$\mu_{A_1}(x)(T)\mu_{A_2}(x) = \mu_{A_2}(x)(T)\mu_{A_1}(x) \rightarrow u_1 = u_2$$

- рівняння площини C_1.

Таким чином, максимальні та мінімальні значення Т-норми зображуються геометричною фігурою симетричною щодо плоскості u_1-u_2=0.

Теоретично нечітких множин залежно від способів завдання операції (Т), які задовольняють аксіомам (4.1), існує нескінченна кількість нечітких операцій «і». Теоретично нечіткого управління знаходять застосування такі їх типи.

Логічний твір (Заде, 1973 р.):

$$\mu_{A_3}(x) = \mu_{A_1\wedge A_2}(x) = \mu_{A_1}(x)(T)\mu_{A_2}(x) = \mu_{A_1}(x)\wedge\mu_{A_2}(x) =$$
$$= \min\left(\mu_{A_1}(x), \mu_{A_2}(x)\right), \quad \forall x \in R_1.$$

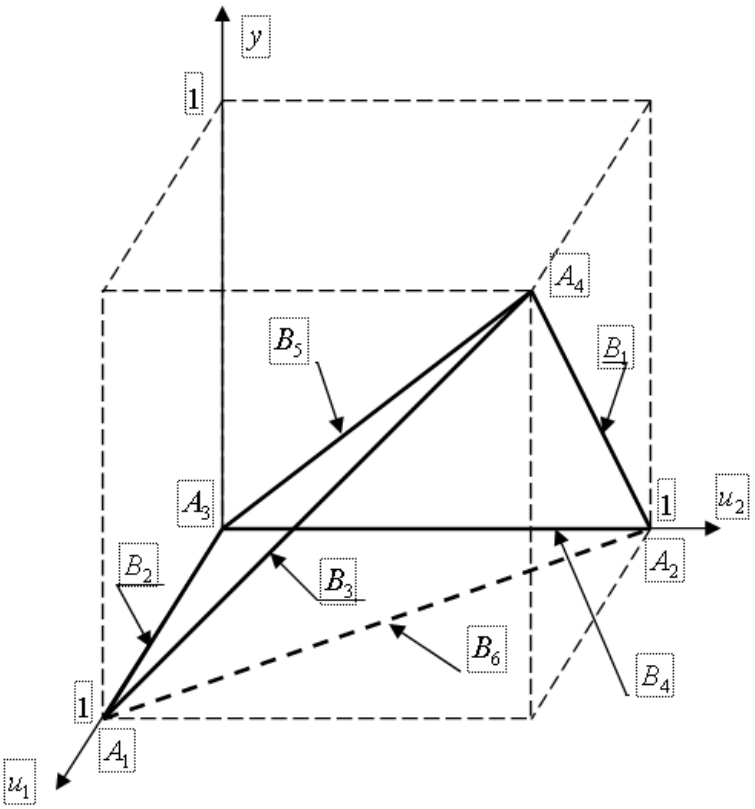


Рисунок 4.4 – Кордонні умови логічного твору

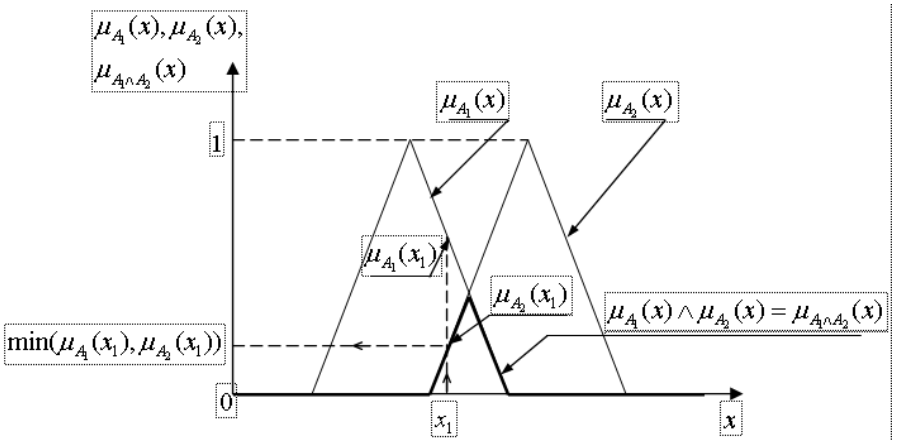


Рисунок 4.5 – Логічний твір нечітких множин

Алгебраїчний твір (Бандлер і Кохоут, 1980):

$$\mu_{A_3}(x) = \mu_{A_1}(x)(T)\mu_{A_2}(x) = \mu_{A_1}(x) \cdot \mu_{A_2}(x), \quad \forall x \in R_1,$$

де символ «◠» – твір, прийнятий у класичній алгебрі. Графік граничних умов Т-норми для т.т. А1-А4, прямих В1-В4 виходить аналогічно попередньому. Знайдемо рівняння кривої В5. Геометрична інтерпретація твору алгебри зображена на рис.4.6.

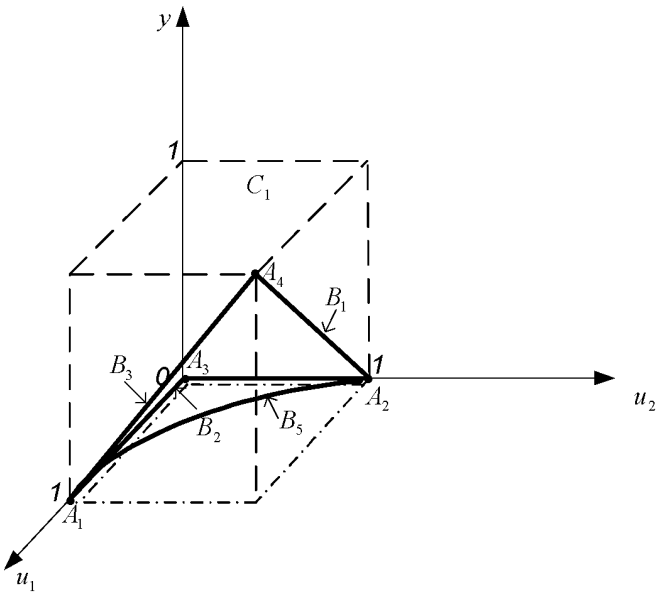


Рисунок 4.6 – Граничні умови твору алгебри

Кордонний твір (Лукашевич, Гілес, 1976):

$$\mu_{A_3}(x) = \mu_{A_1}(x)(T)\mu_{A_2}(x) = \mu_{A_1}(x) \otimes \mu_{A_2}(x) =$$
$$= \max\left(\mu_{A_1}(x) + \mu_{A_2}(x) - 1; 0\right) \equiv \left(\mu_{A_1}(x) + \mu_{A_2}(x) - 1\right) \vee 0, \quad (2.8)$$

де символ $\otimes$ - граничний твір). Графік граничних умов аналогічний попередньому (сукупність т.т. A_1-A_4 та прямих B_1-B_4). Крива B_6 має вигляд:

$y=u_1 \otimes u_2 = u_2 \otimes u_1 \rightarrow (u_1+u_2-1) \vee 0 = 0 \vee (u_1+u_2-1) \rightarrow u_1+u_2-1=0 \rightarrow$

$\rightarrow u_2=1-u_1$.

Геометрична інтерпретація наведена на рис. 4.7.

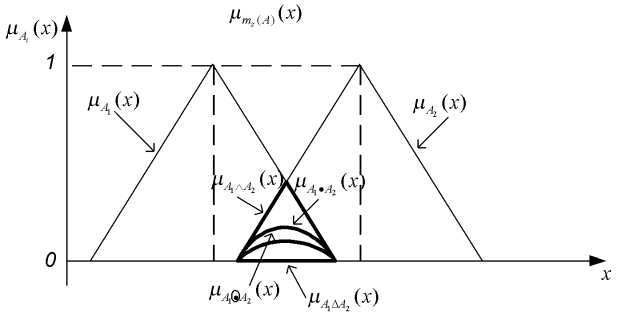


Рисунок 4.7 – Логічний, алгебраїчний, граничний та сильний твіри нечітких множин

Лекція 5 «Методи побудови функцій приналежності»

На практиці зручно використовувати функції приналежності, які допускають аналітичне уявлення у вигляді деякої простої математичної функції. Це спрощує як відповідні чисельні розрахунки, а й скорочує обчислювальні ресурси, необхідних зберігання окремих значень цих функцій власності.

При побудові функцій приналежності нечітких множин слід дотримуватися деяких правил, які визначаються характером невизначеності, що має місце при побудові конкретних нечітких моделей. З практичної точки зору з кожною нечіткою безліччю зручно асоціювати деяку властивість, ознаку або атрибут, які характеризують сукупність об'єктів універсуму, що розглядається. При цьому за аналогією з класичними множинами властивість, що розглядається, може породжувати деякий предикат, який цілком природно назвати нечітким предикатом.

Цей нечіткий предикат може набувати не одне з двох значень істинності ("істина" або "брехня"), а цілий континуум значень істинності, які для зручності вибираються з інтервалу [0,1]. При цьому значенню "істина", як і раніше, відповідає число 1, а значенню "брехня" - число 0. , змістовно це означає наступне. Чим більшою мірою елемент має розглянуту властивість, тим близько до 1 має бути значення істинності відповідного нечіткого предикату.

І навпаки, чим меншою мірою елемент має розглянуту властивість, тим більше близько до 0 має бути значення істинності цього нечіткого предикату. Якщо елемент безумовно не має властивість, що розглядається, то відповідний нечіткий предикат приймає значення "брехня" (або число 0).

Якщо ж елемент безперечно має властивість, що розглядається, то відповідний нечіткий предикат приймає значення "істина" (або число 1). Тоді в загальному випадку завдання нечіткої множини з використанням спеціальної властивості еквівалентне завданням такої функції приналежності, яка змістовно представляє ступінь істинності відповідного одномісного нечіткого предикату.

Найбільшого поширення при побудові функцій приналежності нечітких множин набули прямі та опосередковані методи.

У прямих методах експерт чи група експертів легко задають кожному за значення функції приналежності. Як правило, прямі методи побудови функцій приналежності використовуються для таких властивостей, які можуть бути виміряні деякою кількісною шкалою. Наприклад, такі фізичні величини, як швидкість, час, відстань, тиск, температура та інші мають відповідні одиниці та еталони для свого виміру. При цьому доцільно обмежити розгляд лише тими значеннями величин, які мають фізичний сенс у контексті завдання, що розв'язується. При прямому побудові функцій власності слід враховувати те, що теорія нечітких множин вимагає абсолютно точного завдання функцій власності. Найчастіше буває достатньо зафіксувати лише найбільш характерні значення та вид (тип) функції власності.

Процес побудови чи завдання нечіткої множини з урахуванням деякого відомого заздалегідь кількісного значення вимірної ознаки отримав навіть спеціальну назву — фаззификация чи приведення до нечеткости. Йдеться тому, що хоч іноді нам буває відомо деяке значення вимірної величини, ми визнаємо той факт, що це значення відомо неточно, можливо з похибкою чи випадковою помилкою. При цьому чим менше ми впевнені в точності вимірювання ознаки, тим більшим буде інтервал носія відповідної нечіткої множини. Слід пам'ятати, що у більшості практичних випадків абсолютна точність виміру є лише зручною абстракцією для побудови математичних моделей. Саме з цієї причини фазифікація дозволяє більш адекватно уявити об'єктивно присутню неточність результатів фізичних вимірів.

Як правило, непрямі методи визначення значень функції належності використовуються у тих випадках, коли відсутні очевидні вимірювані властивості, які

можуть бути використані для побудови нечітких моделей предметної області, що розглядається. Серед непрямих методів найвідоміший так званий метод попарних порівнянь. Цей метод використовується для кінцевих нечітких множин і ґрунтується на наступному припущенні. Якби значення потрібні функції приналежності були відомі і рівні значенням $\mu_{\tilde{A}}(x)$ для $i \in \{1,2,...,n\}$, то попарні порівняння відповідних елементів носія нечіткої множини $\tilde{A}$ можна було б подати у вигляді матриці А з елементами a_{ij}, при цьому елементи цієї матриці рівні: $a_{ij} = \mu_{\tilde{A}}(x_i) / \mu_{\tilde{A}}(x_j)$, де символ "/" означає операцію поділу. На практиці буває простіше спочатку побудувати матрицю А у припущенні, що її діагональні елементи повинні дорівнювати 1, а симетричні щодо головної діагоналі елементи повинні бути взаємно зворотними. Остання умова означає, що якщо ступінь приналежності одного з елементів оцінюється в а раз сильніше ступеня приналежності іншого, то ступінь приналежності другого елемента повинна бути в 1/а раз сильніше за ступінь приналежності першого елемента. У цьому випадку завдання побудови функції приладдя зводиться до знаходження такого вектора w, який є рішенням наступного рівняння

$$A \cdot w = \lambda_{\max} w, \tag{5.1}$$

де $\lambda_{\max}$ найбільше власне значення матриці А. Оскільки всі значення елементів матриці А є позитивними за побудовою, рішення даного рівняння існує і є позитивним. Власне, процес попарного порівняння елементів може бути заснований на суб'єктивній інтуїції або на виконанні деякої послідовності алгоритмічних або логічних дій. При цьому окремі елементи універсуму можуть використовуватися як еталони або всі елементи можуть бути розділені на групи з подальшим порівнянням цих груп між собою. З алгоритмічних процедур найбільшої популярності набули методи ітеративного уточнення значень функцій належності, засновані на нейронних мережах та генетичних алгоритмах. Логічні процедури використовують методи індуктивного навчання та побудови нечітких метаправил. Іноді застосовуються методи обробки статистичних даних, факторного та дискримінантного аналізу з метою виділення значущих ознак для подальшого порівняння елементів універсуму, що розглядається. На закінчення слід зазначити, що у разі нестачі інформації про особливості функцій приналежності нечітких змінних рекомендується починати побудову нечіткої моделі з використання найпростіших форм функції приналежності, а саме кусково-лінійних функцій. Надалі їх характер може бути уточнений і врахований на етапі корекції нечіткої моделі.

Існують наступні алгоритми нечіткого висновку Мамдані, Сугено та деякі інші, зокрема Ларсена та Цукомото. Проаналізуємо їх особливості докладніше.

Першим і найпопулярнішим з них є Mamdani, що дозволяють гнучко та оперативно виробляти логічний нечіткий висновок щодо змінних завдяки використанню мінімаксної функції в якості основи.

Відповідно до сучасних теоретичних положень в області НМ формований логічний висновок за даним алгоритмом здійснюється строго відповідно до створеної нечіткої БЗ, тобто:

$$\bigvee_{p=1}^{k_j}\left(\bigwedge_{i=1}^{n} x_i = a_{i,jp} \quad c \quad весом \quad w_{jp}\right) \to y = d_j, j = \overline{1,m} \tag{6.1}$$

У самій БЗ значення всіх лінгвістичних змінних (вхідних та вихідних) є окремими НМ.

Інтегруємо такі позначення для формалізації логіки роботи даного алгоритму:

– $\mu_{jp}(x_i)$ - ФП вхідного значення змінної до відповідного нечіткого лінгвістичного терму $a_{i,jp}$, тобто:

$$a_{i,jp} = \int_{\underline{x_i}}^{\overline{x_i}} \mu_{jp}(x_i)/x_i, x_i \in \left|\underline{x_i}, \overline{x_i}\right| \tag{6.2}$$

.

– $\mu_{d_j}(y)$ - ФП вихідного значення змінної до відповідного нечіткого термо d_j, тобто:

$$d_j = \int_{\underline{y}}^{\overline{y}} \mu_{d_j}(y)/y, y \in \left|\underline{y}, \overline{y}\right| \tag{6.3}$$

Слід зазначити, що підсумкові ступені приналежності сформованого вхідного вектора $x^* = (x_1^*, x_2^*, ..., x_n^*)$ з усіх нечітких лінгвістичних термів d_j, отриманим безпосередньо із створеної БЗ, може бути визначено як:

$$\mu_{d_j}(X^*) = \bigvee_{p=1,k_j} w_{jp} \cdot \bigwedge_{i=1,n}\left[\mu_{jp}\left(x_i^*\right)\right], j = \overline{1,m} \tag{6.4}$$

,

де $\vee(\wedge)$ є операцією по s-нормі (для зворотного випадку t-норми), що виражається у вигляді реалізацій вихідної логічної операцій OR (AND).
З функціональних можливостей цього підходу найчастіше застосовується практично: у разі функції OR – визначення максимуму; у разі функції AND – визначення мінімуму.

В результаті формується підсумкова НМ $\tilde{y}$, яка відповідає вхідному вектору X^*:

$$\tilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + ... + \frac{\mu_{d_m}(X^*)}{d_m} \tag{6.5}$$

Специфікою подібного НМ є фактор того, що для нього узагальненою та універсальною множиною є вихідна терм-множина по змінній y. Такі НМ називають НМ другого порядку.

У процесі міграції від НМ зазначеного на універсальному НМ у ряді нечітких лінгвістичних термів $(d_1, d_2, ..., d_m)$ до НМ на проміжку $[\underline{y}, \overline{y}]$ потрібно:

– обмежити функію приналежності $\mu_{d_j}(y)$ по значенню $\mu_{d_j}(X^*)$;

– агрегувати підсумкові НМ у вигляді проведення процедури об'єднання такого виду:

$$\tilde{y} = \underset{j=\overline{1,m}}{agg}\left(\int_{\underline{y}}^{\overline{y}} \min(\mu_{d_j}(X^*), \mu_{d_j}(y)) / y \right), \tag{6.6}$$

де agg є функцією агрегацію НМ для процедури знаходження максимального значення.
Вихідне чітке значення за цільовою змінною y, яке відповідає підсумковому вхідному вектору X^* оцінюючи в результаті дефазифікації НМ $\tilde{y}$.

В рамках цього підходу доцільно застосування методу дефаззифікації на основі обчислення центру тяжіння:

$$y = \frac{\int_{\underline{y}}^{\overline{y}} y \cdot \mu_{\tilde{y}}(y) dy}{\int_{\underline{y}}^{\overline{y}} \mu_{\tilde{y}}(y) dy}. \tag{6.7}$$

Алгоритм Mamdani реалізується за допомогою низки етапів:

1. Процедура фазифікації реалізується з метою оцінки підсумкового ступеня істинності терму, значення всіх підключених до процесу функції належності тотожним стосовно всім передумова логічних правил.

2. Процедура нечіткого логічного висновку, з урахуванням якої здійснюється обчислення значень обмеження кожного з створених логічних правил з його лівої частини, з урахуванням чого формуються набори створених функцій власності.

3. Етап агрегації створених функцій після процедури усічення, що реалізується шляхом композиції сформованих НМ у контексті одного обчислювального процесу.

4. Процедура дефазифікація реалізується на формування чіткого логічного висновку значень змінних з урахуванням використання середнього чи загального центра.

Геометричний зміст подібного процесу полягає в отриманні центру тяжіння для деякої кривої $MF(y)$.

Рис. 6. 1 візуально відображає підсумковий процес здійснення нечіткого логічного висновку для поданого на вхід набору з двох змінних та нечітких правил $R1$ та $R2$, одержаних на базі продукцій.

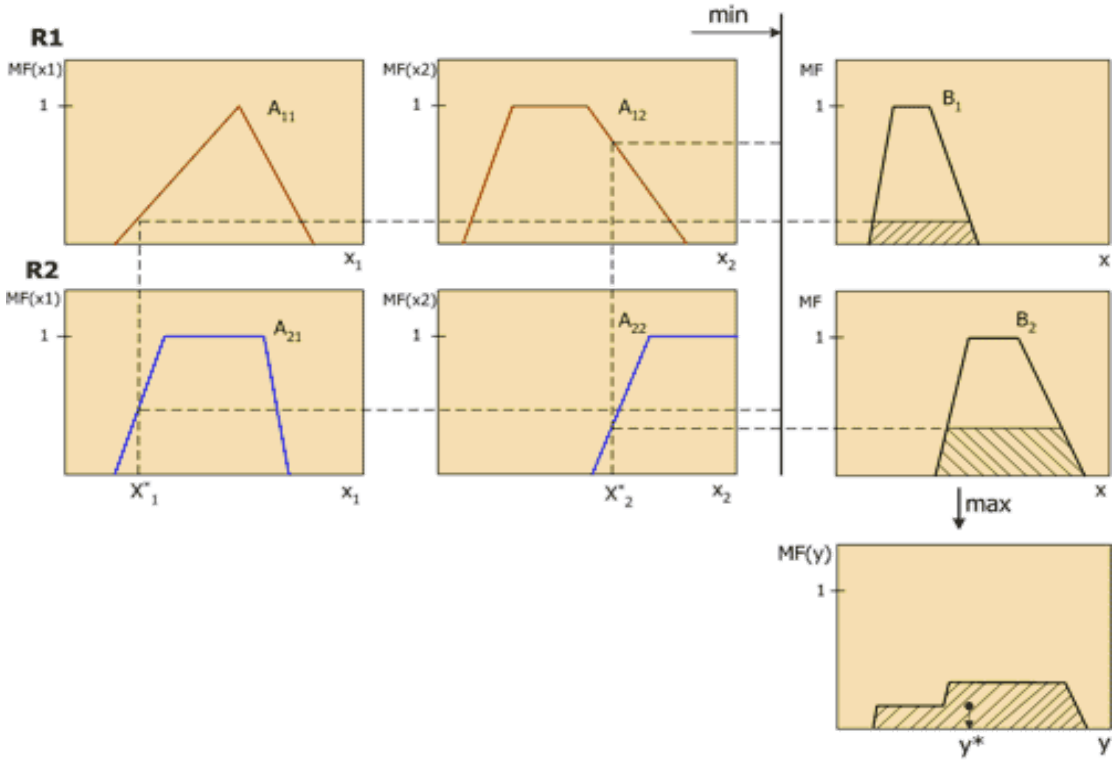


Рисунок 6.1– Узагальнена схема нечіткого логічного висновку за алгоритмом Mamdani

Порівняємо цей алгоритм з логікою роботи алгоритму Сугено. Відповідно до розроблених положень алгоритм Сугено для процедури проведення нечіткого логічного висновку реалізується за БЗ наступного виду:

$$\bigwedge_{p=1}^{k_j}\left(\bigwedge_{i=1}^{n} x_i = a_{i,jp} \quad c \quad весом \quad w_{jp}\right) \rightarrow y = b_{j,0} + b_{j,1}\cdot x_1 + b_{j,2}\cdot x_2 + ... + b_{j,n}\cdot x_n, j = \overline{1,m} \qquad (6.8)$$

де $b_{j,i}$ являють собою набори чисел різного діапазону.

БЗ, складена для алгоритму Сугено практично тотожна БЗ по Мамдані - алгоритму, крім відмінностей у висновках наборів логічних правил d_j, що задаються не на основі використання нечітких лінгвістичних термів, а загальною функцією по вхідним змінним, тобто

$$d_j = b_{j,0} + \sum_{i=\overline{1,n}} b_{j,i} \cdot x_i \quad . \tag{6.9}$$

Правила, що формуються в контексті БЗ за алгоритмом Сугено, являють собою транслятори сигналів по одному лінійному закону в інший. При цьому загальні межі одержуваних підобластей фактично є розмитими, що дозволяє одночасно здійснювати відразу кілька побудов лінійних залежностей з різними ступенями.

Підсумкові ступені належності складеного вхідного вектора до нечітких значень може бути визначено у вигляді:

$$\mu_{d_j}(X^*) = \bigvee_{p=\overline{1,k_j}} w_{jp} \cdot \bigwedge_{i=\overline{1,n}} \left[\mu_{jp}\left(x_i^*\right) \right], j = \overline{1,m} \quad . \tag{6.10}$$

При використанні алгоритму Сугено часто застосовуються реалізації трикутних норм виду: ймовірнісного OR як основа s-норми та простий твір для t-норми.

В результаті проведення всіх обчислювальних операцій формується вихідний НМ y`, що знаходиться у повному відповідність до вхідного вектора X^*:

$$\tilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + \ldots + \frac{\mu_{d_m}(X^*)}{d_m} \quad . \tag{6.11}$$

Слід зазначити, що створене за алгоритмом Сугено НМ є звичайним НМ першого порядку, заданим на багатьох чітких значень чисел, на відміну від розглянутого раніше результату алгоритму Мамдані. Вихідне значення цільової змінної у може бути розраховане у вигляді суперпозиції набору лінійних залежностей, що існують в заданій точці X* n-мірного простору факторів. Специфіка алгоритму Сугено у візуальному вигляді наведена на рис.6.2 за 2 правилами.

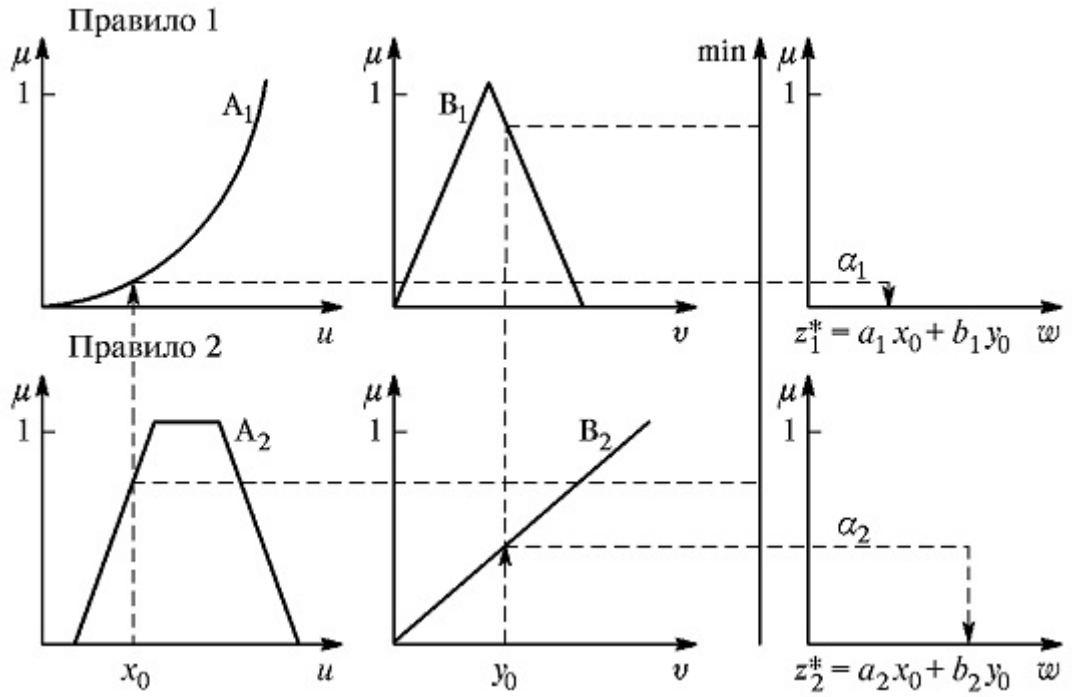


Рисунок 6.2 – Геометрична візуалізація специфіки виконання алгоритму Сугено

Алгоритм Ларсена загалом схожий раніше розглянуті алгоритми з низкою нюансів, тобто. здійснюється у кілька етапів:

- Формується набір БЗ на основі низки правил системи нечіткого логічного висновку за аналогією з Мамдані-підходом.
- Фаззифікація створених вхідних змінних моделей проводиться також ідентично до алгоритму Мамдані.
- Агрегування сформованих умов за всіма правилами НЛ виконується на базі нечіткого «AND» за такими елементарними логічними висловлюваннями:

$$A, B : T(A \cap B) = \min\{T(A); T(B)\} \tag{6.12}$$

- Активізація отриманих на попередньому етапі підскладання за створеними правилами нечіткої логічної продукції виконується на основі застосування методу prod-активізації, необхідного для утворення набору логічних консеквентів ядер продукційних правил НМ.
- Акумуляція отриманих підзаключень за всіма правилами БЗ реалізується ідентично Мамдані-підходу на основі виконання max-об'єднання тобто

$$T(A \cap B) = \min\{T(A); T(B)\} \tag{6.13}$$

- Дефаззифікація може бути проведена будь-яким із заданих раніше підходів.

Геометрична візуалізація принципів реалізації алгоритму Ларсена наведена на рис.6.3.

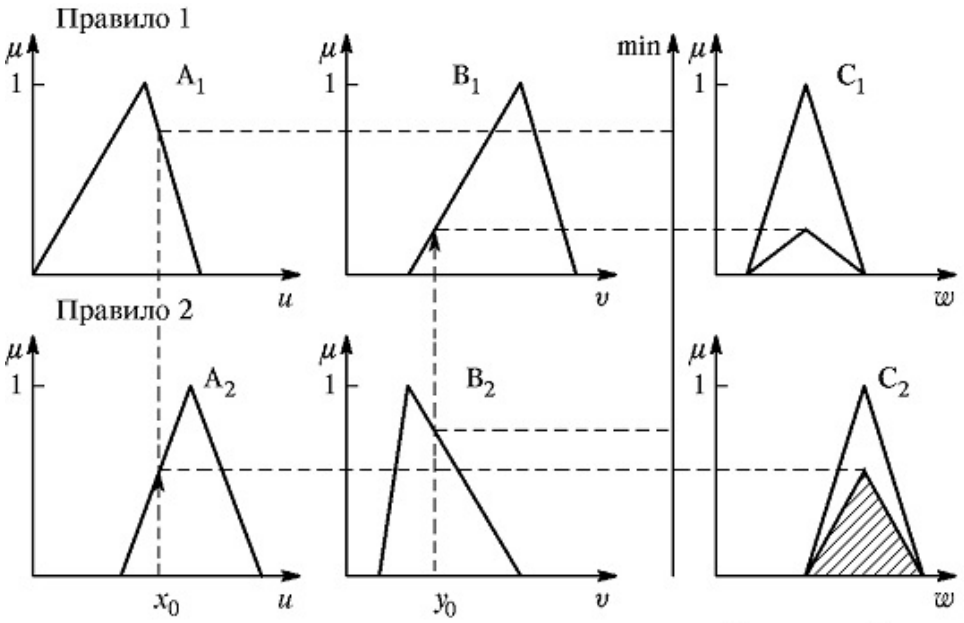


Рисунок 6.3 – Геометрична візуалізація алгоритму Ларсена

Відповідно до Цукамото-підходом передбачається, що функції C1(z) та C2(Z) для вихідних лінгвістичних змінних є строго монотонними.

Даний алгоритм базується на етапах формування БЗ, фазифікації та агрегації умов проваїл ідентичний алгоритму Мамдані. Проте етап активізації сформованих підукладень всіх правил БЗ реалізується дещо інакше у межах 2 процедур.

Підсумкові значення консеквентів нечітких правил БЗ визначаються у вигляді твору алгебри значення ступеня істинності наявного антецедента правила і відповідного вагового коефіцієнта. Після цього, на відміну від Мамдані алгоритму замість формування ФПР за всіма правилами використовується вираз μ(x)=c для визначення вихідного чіткого значення цільової лінгвістичної змінної. μ (x) у цьому випадку є ФПР термів лінгвістичної змінної і є оцінкою ступеня істинності логічних посилок відповідно до НЛ для консеквентів ядер нечітких правил за продукційним типом. Етап акумуляції висновків складених правил БЗ не здійснюється, т.к. дискретні множини всіх чітких значень за цільовими лінгвістичними змінними вже раніше отримані. для кожної із вихідних лінгвістичних змінних. Приклад візуалізації процесу операцій із Цукамото підходу наведено на рис. 6.4.

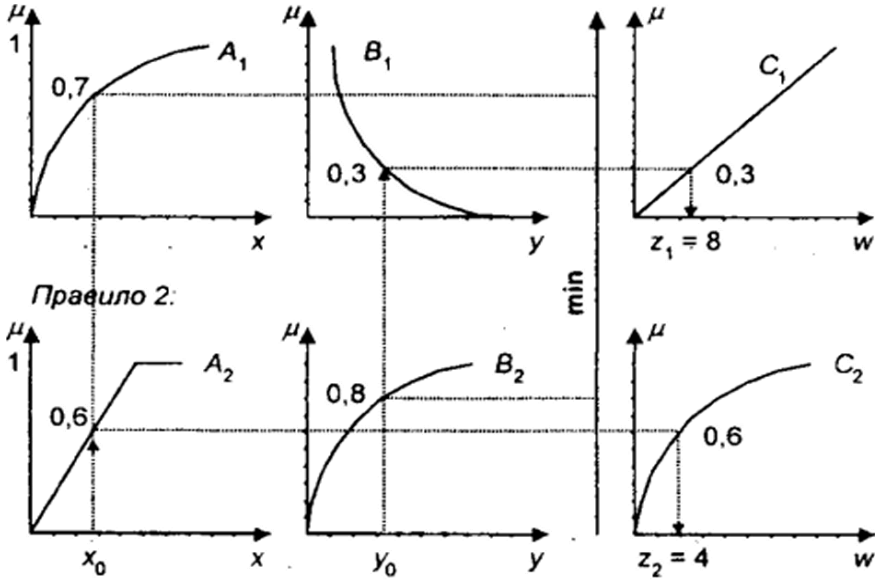


Рисунок 6.4 – Візуалізація алгоритму Цукамото

В результаті проведеного аналізу розглянутих методів можна агрегувати загальну інформацію у вигляді таблиці 6.1, наведеної нижче.

Таблиця 6.1 – Результати проведення порівняльного аналізу алгоритмів нечіткого логічного висновку

Назва алгоритму	Опис
Мамданы	Даний алгоритм є найбільш швидким і універсальним з розглянутих, за допомогою чого має перевагу зі швидкості проведення комплексу обчислювальних процесів при формуванні нечіткого логічного висновку. Його зручність у простоті та наочності, завдяки зручності інтерпретації кількісних значень як вхідних, так і вихідних змінних.
Цукамото	Алгоритм ефективний в обмеженому ряді випадків, зокрема при необхідності аналізу та оцінки моделей для опису лише монотонних функцій за цільовими змінними. Наочність і точність алгоритму трохи менше, ніж в інших.
Ларсена	Перевагою даного алгоритму порівняно з Мамдані-підходом є більший рівень точності при аналізі та оцінці немонотонних вхідних змінних множини, проте недолік полягає у необхідності проведення більшої кількості операцій мультиплікації.
Сугено	Сугено підхід може бути застосований у ситуаціях, коли дані менш формалізовані, немає даних за формою ФПР, відомі значення лише вагових коефіцієнтів. На відміну від Мамдані алгоритму у разі неможливо створити логічні правила БЗ, до складу яких входять диз'юнкції, які у лівої частини імплікації.

Лекція 7 «Можливості сучасних систем побудови нечітких моделей»

На сьогоднішній час існує декілька систем, здатних моделювати та створювати нечіткі моделі, розглянемо їх можливості та переваги.

1. Matlab являє собою уніфіковану та універсальну програмну платформу проведення математичних обчислень та моделювання складних систем та процесів, що імплементує в собі середовище розробки програмного коду, незалежні плагіни, модулі та утиліти візуалізації. За допомогою застосування системи Matlab розробники та дослідники здатні проводити комплексний аналіз та вивчення даних у контексті вирішення прикладних завдань шляхом розробки та імплементації обчислювальних алгоритмів, створюючи імітаційні та когнітивні моделі, а також окремі програмні скрипти та додатки [20].

Мова розробки програмного коду в системі Matlab є с-подібною, імплементуючи різні парадигми, як функціональну, так і об'єктно-орієнтовану, підтримується різноманітний модульний інструментарій, виклик вбудованих в систему бібліотек реалізації різних математичних функцій з виконання операцій з багатовимірними векторами. Засоби розробки, реалізовані в системі уможливлюють різнобічний аналіз застосовуваних підходів та методів для побудови та моделювання систем у режимі реального часу з оперативним формуванням наборів вихідних даних з мінімальним використанням безпосереднього програмного коду та його розробкою.

Систему Matlab на практиці часто використовують інженери, аналітики та програмісти для вирішення таких завдань, як:

- обробці та аналізі наборів аналогових та цифрових сигналів з метою побудови моделей засобів передачі та обробки даних у процесі використання технологій та процесів зв'язку;
- аналізі та обробці мультимедійних даних різних форматів;
- моделювання комплексних систем автоматичного управління об'єктами та системами, їх окремими блоками та модулями;
- автоматизації процесів тестування функціоналу створених програм, програмних скриптів, імітаційних моделей;
- аналіз фінансових рядів.

Matlab імплементує у своєму складі набори засобів автоматизації процесів розробки коду для вирішення типових обчислювальних, оптимізаційних та аналітичних завдань математичної тематики, що у ряді випадків є більш ефективними та швидкими в порівнянні з іншими високорівневими мовами програмування.

Ядро цієї системи оптимізовано для оперативної обробки наборів зведених та багатовимірних матриць різних типів даних, реалізуючи модульні засоби інтеграції сторонніх структур даних табличних форматів.

Спеціалізовані можливості та переваги використання системи полягають у підтримці низки низькорівневих науково-прикладних математичних функцій лінійної алгебри, проведення інтегральних обчислень та реалізації перетворення Фур'є, побудови поліноміальних моделей різного ступеня, аналізу та оцінки статистичних гіпотез, ймовірнісних законів, вирішення завдань різними чисельними методами, числа та диференціальних рівнянь.

Інтерфейс системи Matlab наведено на рис.7.1.

Загальними перевагами використання системи Matlab є:

- велика кількість функціональних модулів, структурованих за різними категоріями та доповнюючих один одного;
- підтримка складних операцій обробки та проведення обчислень над великими обсягами даних у вигляді векторів та матриць;

– реалізація внутрішніх модулів і фреймворків для створення інтерактивного інтерфейсу створюваних програмних додатків;

– високий рівень продуктивності та загальної швидкості проведення обчислювальних процесів завдяки підтримці використання вставок коду та інтеграції мов C та C++.

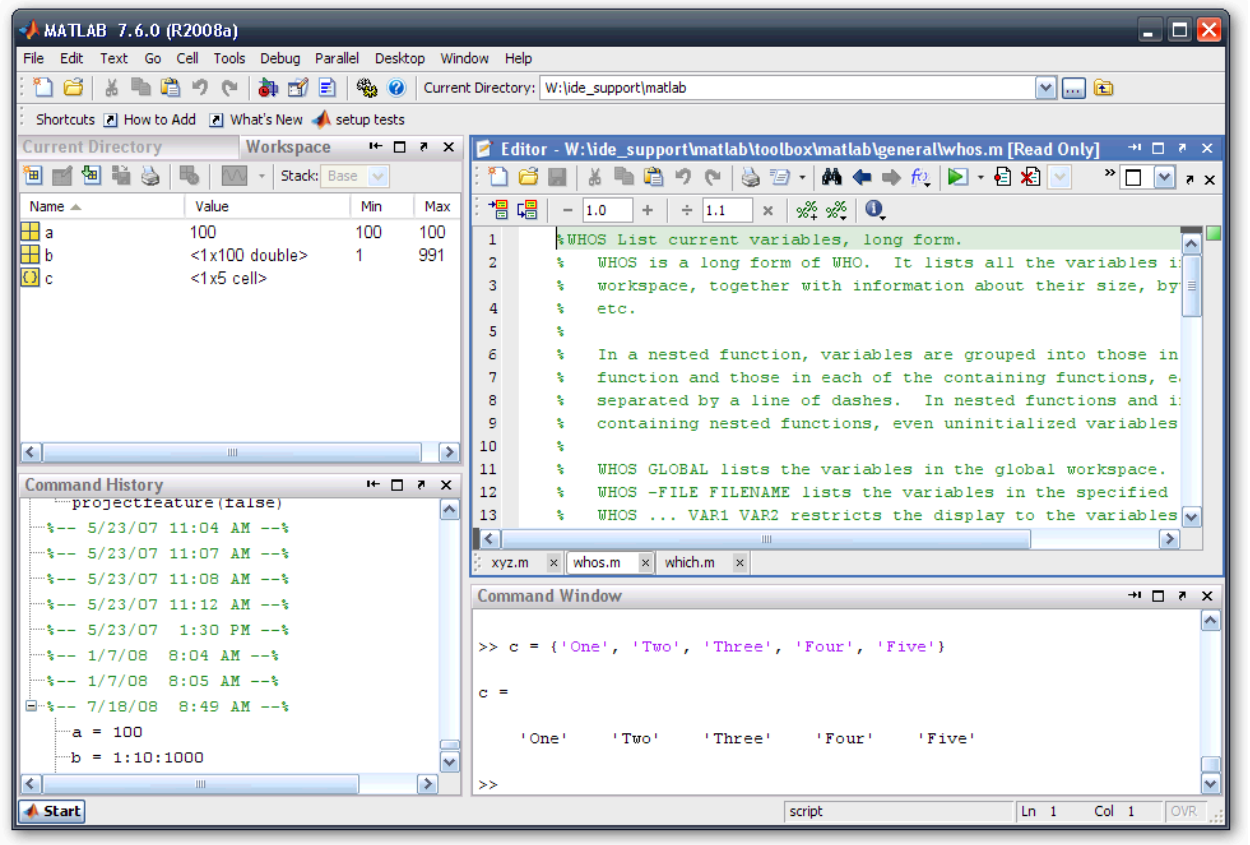


Рисунок 7.1 – Інтерфейс системи Matlab

Поряд із перевагами системи існує і ряд деяких недоліків:

– значний рівень ціни за використання самої системи та її окремих модулів у повноцінному режимі;

– фрагментарна підтримка складних функцій та засобів складання проектів під різні платформи;

– складності інтеграції та конвертації з мовою java та деякими пропрієтарними платформами інтернету речей.

У контексті використання системи Matlab для вирішення задач НЛ доцільним є застосування модуля Fuzzy Logic Toolbox (FLT).

FLT є самостійним програмним модулем, що з'явився в системі з її перших версій і надає набір функцій зі створення нечітких моделей в інтерактивному режимі для створення систем, що використовують принципи НЛ для вирішення завдань класифікації, кластеризації, аналізу та оцінки даних. Завдяки можливостям інтеграції модулів у Matlab реалізовано підтримку використання системи Simulink.

У складі FLT розробники оперують поняття нечіткої структури моделі даних, що імплементує перелік вхідних та вихідних змінних для формування загальної концепції системи.

Для реалізації можливостей створення моделей НЛ FLT використовується ряд окремих модулів із графічним інтерфейсом, зокрема:

– форми завдання різних функцій перетворення;

- довідково-пошуковий модуль;
- набір компонентів інтеграції нечітких моделей у Simulink;
- набір демонстраційних прикладів використання НМ.

Модуль FLT реалізує низку алгоритмів проведення нечіткого логічного висновку, зокрема розглянуті раніше Мамдані та Сугено.

Ключові відмінності у реалізації даних алгоритмів у контексті FLT полягають у різних підходах до створення вихідних змінних з урахуванням формування БЗ. Зокрема, у разі використання першого алгоритму значення вихідної цільової змінної формується на основі створення НЛТ, у другому алгоритмі використовується лінійна комбінація всіх поданих на вхід моделі змінних.

FLT реалізований у контексті системі MatLab у вигляді структури даних, представленої на рис.7.2.

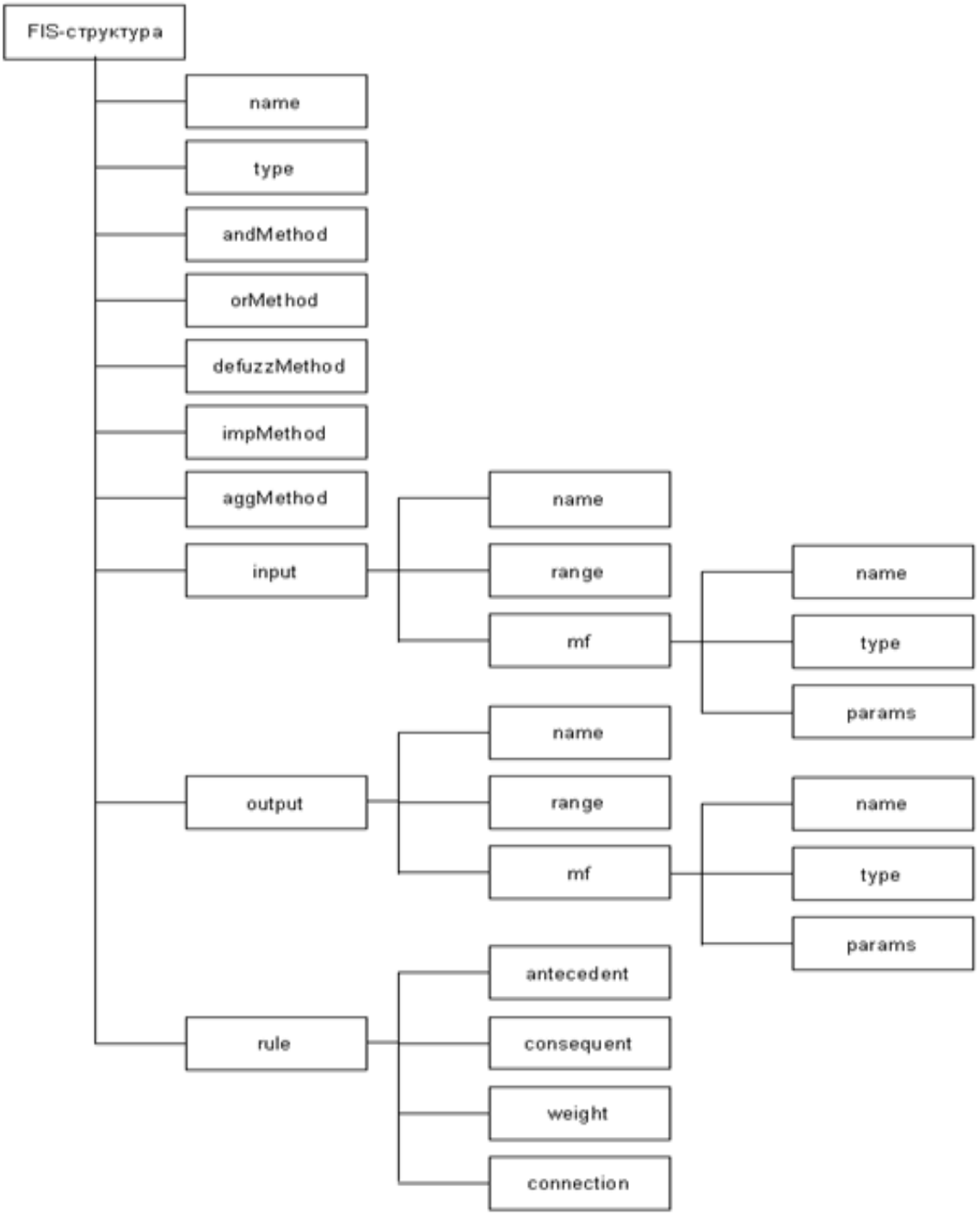


Рисунок 7.2 – Структура FLT

Цей модуль, на додаток до перерахованих можливостей, підтримує всі ключові стадії розробки нечітких моделей згідно з положеннями НЛ в інтерактивному вигляді.
Вбудовані засоби візуалізації формують для користувача можливості створення зрозумілого середовища для всіх етапів моделювання, в тому числі для реалізації Ansis кластеризації та проведення адаптивної нейро-нечіткої настройки при створенні гібридних моделей НЛ і нейронних мереж.

Важливою перевагою FLT є його простота, наочність та інтерактивність, серіалізовані модулем дані можуть бути переглянуті та модифіковані у текстовому вигляді, після чого завантажені у вигляді окремого об'єкта у форматі *.fis, з подальшими можливостями розширення функціоналу за ФОП.
Також модуль імплементує функції створення нечітких моделей через консоль, m-модулі та безпосередньо через форми графічного інтерфейсу з підтримкою налаштування роботи систему візуальним чином через систему діалогових вікон. Підтримуються такі програми та утиліти:

– Редактор властивостей нечіткої системи. Націлений створення заданого користувачем числа вхідних і вихідних змінних із завданням їх назв, реалізовані можливості вибору типів функцій дефазцикації та алгоритму нечіткого логічного висновку, логічних функцій (AND,OR), головне меню для навігації формами інших модулів створення моделі.

– Графічний редактор, створений для розробки та візуалізації ФПР. Практичне використання цієї програми полягає у відображенні форми інтерактивного завдання та коригування форм графіків по всіх заданих лінгвістичних змінних відповідних вхідних та вихідних змінних за допомогою зміни кількості термів та їх параметрів (числові діапазони, форма, накладення один на одного).

– Редактор нечіткої БЗ, що використовується для створення продукційних правил за обраними логічними операціями та завдання вагових значень значущості за кожним із правил з урахуванням їх впливу на вихідну змінну.

– Форма перегляду результатів проведення нечіткого логічного висновку. Може бути використана для візуального опису залежності чисельних значень сукупного впливу всіх вхідних змінних на вихідну цільову змінну в графічному вигляді.

– Форма створення та відображення залежності вихідної змінної від вибраних вхідних змінних у тривимірному вигляді за допомогою різних розмірностей, типів та налаштувань графічних елементів.

2. FisPro (Fuzzy Inference System Professional) — це програма, яка дозволяє створювати системи нечіткого логічного висновку та використовувати їх для міркування, особливо для моделювання фізичної або біологічної системи. Це також дозволяє повністю спроектувати систему нечіткого висновку з числових даних, пов’язаних з досліджуваною проблемою. Приклад інтерфейсу системи наведено на рис.7.3.

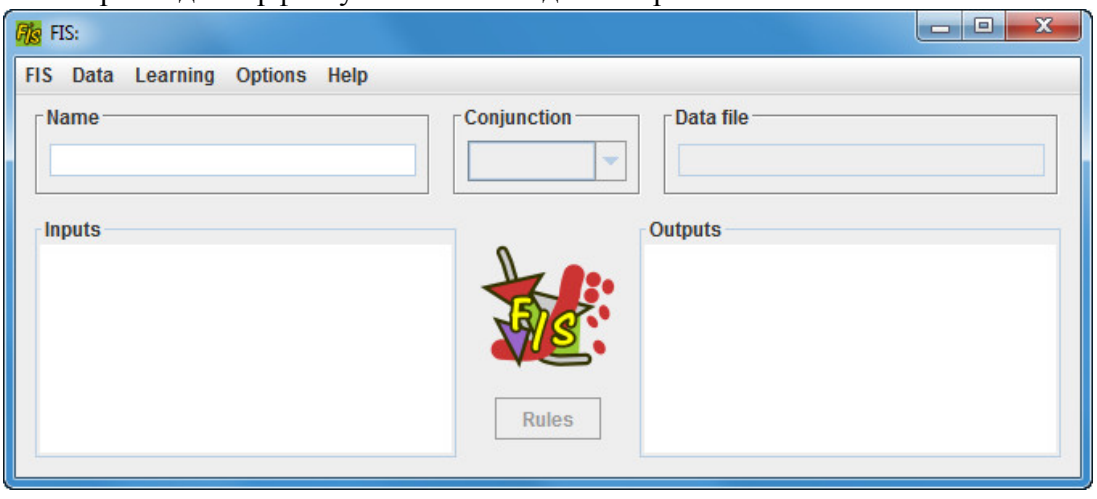


Рисунок 7.3 – Інтерфейс FisPro

3. Пакет прикладних програм Fuzi Calc призначений для виконання швидких підрахункових розрахунків у задачах прийняття рішень, що виникають у різних галузях бізнесу. Пакет забезпечує обчислення у форматі електронних таблиць та зручний для аналізу фінансових показників. Він містить велику кількість стандартних функцій, що використовуються у завданнях аналізу фінансово – господарської діяльності підприємств. За допомогою пакету Fuzi Calc можна визначати показники прибутковості та окупності бізнесу, оцінки заборгованості та прискорювати складання оперативних фінансових звітів та інших документів.

До відомих алгоритмічно-орієнтованих програмних продуктів відносяться пакети CubiCalc та NeuFuz.

Пакет CubiCalc надає набір засобів для побудови нечітких моделей, у тому числі нечіткий редактор для введення, виведення та редагування значень ЛП, відповідних їм нечітких множин та правил; оператори, що дозволяють змінити значення ЛП за допомогою спеціальних модифікаторів (наприклад, <кілька>, <дуже>); вагові коефіцієнти правил. Cubi Calc містить нечіткий словник і базу знань, що редагується. Результати нечіткого виведення подаються у вигляді двовимірних ґрат.

Система NeuFuz поєднує у собі нечіткий висновок та нейронну мережу, яка використовується для автоматичного формування правил та функцій приналежності за прикладами. Нейронна мережа складається з семи шарів і навчається методом зворотного поширення помилки.

Інструментальне середовище Fide дозволяє будувати нечіткі інтелектуальні системи шляхом інтеграції модулів нечіткого виведення, кожен з яких визначений у текстовому файлі, що містить визначення вихідних та вихідних змінних, пов'язаних із нечіткими множинами та правилами. Відмінною рисою системи Fide є креативний модуль, що включає підсистему аналізу та підсистему моделювання.

Середовище для розробки нечітких систем FuzzyTECH містить безліч редакторів, призначених для створення вхідних та вихідних змінних, функцій належності та правил. Це середовище породжує програмні коди інтелектуальних систем мовою C/C++ та інших мовах, орієнтованих спеціальні мікроконтролери .

Інструментальна система TILShell є засобом CASE-технології, призначеним для створення нечітких експертних систем. Ядро системи містить редактори правил та функцій приладдя, а також розвинені засоби для швидкої розробки конкретних додатків, які легко можуть бути стиковані з іншими програмними розробками, зробленими за допомогою інструментарію TIL.

Фазифікація (fuzzification) — це визначення ступеня виконання антецедентів правил. За допомогою фазифікації чіткому значенню ставляться у відповідність ступені його приналежності до нечітких множин.

Фазифікація полягає у перетворенні чітких вхідних величин $x = (x_1, x_2, ..., x_n)$ до нечітких множин $A' = (A'_1, A'_2, ..., A'_n)$. У більшості випадків для цього використовуються синглетонні моделі. Синглетон чіткого значення x_i є нечіткою множиною $A'_i(x, \mu_A(x))$ з функцією належності

$$\mu_A(x) = \begin{cases} 1, x = x_i; \\ 0, x \neq x_i; \end{cases} \tag{8.1}$$

При фазифікації чіткого входу x_i визначають ступені його відповідності кожному лінгвістичному терму $A_{i,j}$, з функціями належності $\mu_{A_{i,j}}(x), j = 1..m_i$. Ці ступені є значеннями функцій належності $\mu_{A_{i,j}}(x_i)$ у точці $x = x_i$, або інакше – значенням $A_{i,j}(x_i), i = 1,...,n$.

Зазвичай компонент системи, що виконує цей процес, називається фазифікатором. На цьому етапі вводиться функція фазифікації або кожна вхідна змінна для вираження пов'язаної з нею невизначеності вимірювань.

Мета функції фазифікації полягає в тому, щоб інтерпретувати вимірювання вхідних змінних, кожна з яких виражається реальним числом, як більш реалістичною нечіткою апроксимацією відповідних реальних чисел.

Більш розпливчасті моделі невизначеностей виявляються на вході і отже це згладжує відповідь системи, що робить його менш чутливим до конкретних вхідних значень і, отже, до невизначеності на вході, такий як шум.

У багатьох випадках вхідні змінні не є чіткими. Тобто застосовується одноразова фазифікація, яка передбачає відсутність шуму на чітких входах, тому вимірювання вхідних змінних використовуються безпосередньо в процесі виведення.

Одноразова фазифікація зіставляє чітке вхідне значення x_0 нечіткому синглтон, тобто нечіткій множині, так що його підтримка зводиться до x_0. Цей тип фазифікації вимагає низьких обчислювальних витрат, так як розрахунок вихідних даних системи спрощений.

Однак одноразова фазифікація не завжди може бути адекватною, особливо в тих випадках, коли вхідні дані пошкоджені шумом.

Для врахування невизначеності в даних необхідна не одноразова фаззифікація. У таких випадках функція фазифікації має вигляд:

$$f_e : [-a, a] \to X \tag{8.2}$$

де X позначає множину всіх нечітких множин, $f_e(x_0)$ - нечітка апроксимація вимірювання x_0.

Далі оброблені методами логіки невиразності вхідного сигналу, механізм виведення використовує базу нечітких правил для виведення нечіткого виведення за допомогою

композиційного правила виведення. Композиційне правило може бути застосоване локально до кожного правила і результуючі нечіткі множини агрегуються для отримання виведеної нечіткої множини.

Структура системи нечіткого логічного виведення наведена на рис.8.1.

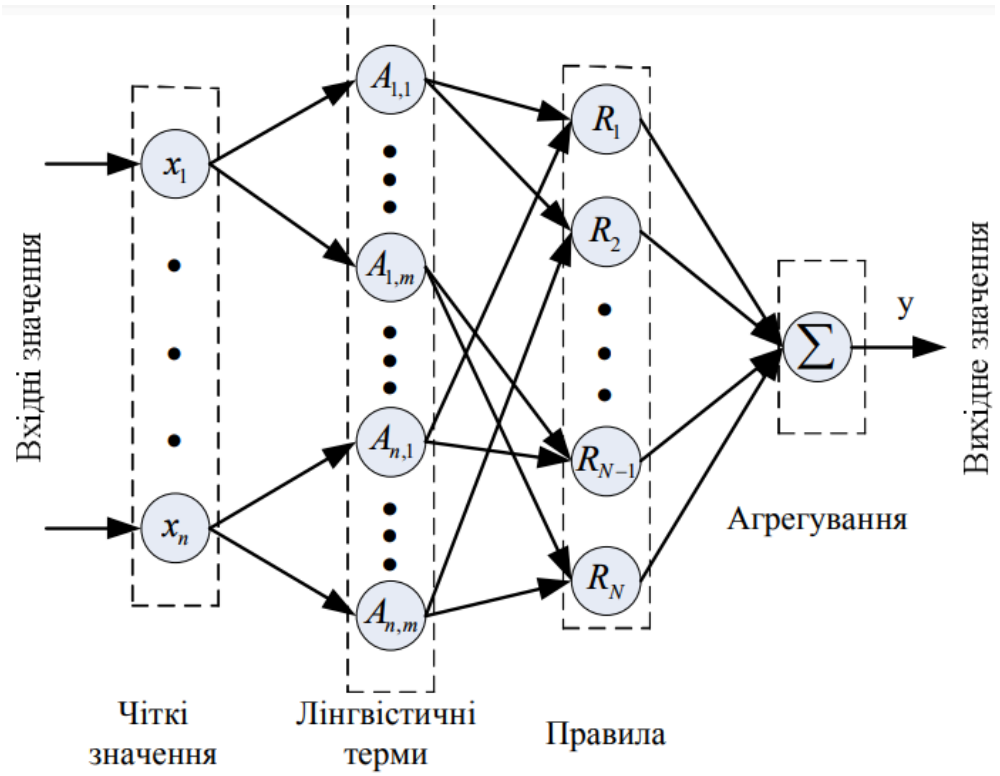


Рисунок 8.1 – Структура системи нечіткого логічного виведення

Алгоритми НЛВ сучасні вчені класифікують за видом використовуваних в завданнях логічних правил, операцій і за методами дефазифікації.

Розглянемо основні методи дефаззифікації.

Центроїдний метод у загальному випадку:

$$z_0 = \frac{\int\limits_{\Omega} zC(z)dz}{\int\limits_{\Omega} C(z)dz} \tag{9.1}$$

Для дискретного варіанта:

$$z_0 = \frac{\sum\limits_{i=1}^{n} \alpha_i z_i}{\sum\limits_{i=1}^{n} \alpha_i} \tag{9.2}$$

Перший максимум (First-of-Maxima). Чітка величина виведення знаходиться як найменше значення, при якому досягається максимум підсумкової нечіткої безлічі (рис. 9.1):

$$z_0 = \min\left\{z \middle| C(z) = \max_U C(U)\right\} \tag{9.3}$$

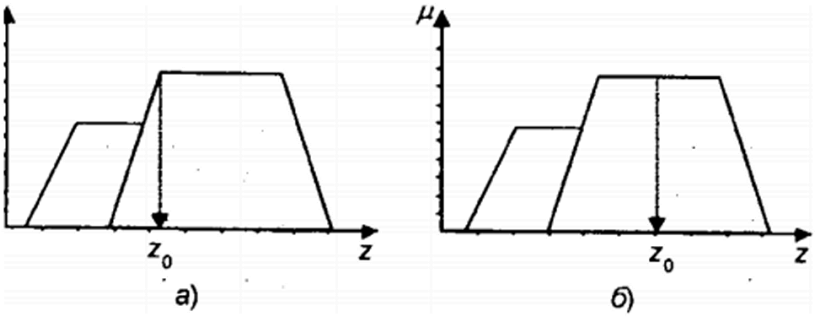


Рисунок 9.1 – Ілюстрація до методів приведення до чіткості: а - перший максимум; б - середній максимум

Середній максимум (Middle-of-Maxima). Чітке значення знаходиться за формулою:

$$z_0 = \frac{\int\limits_{G} zdz}{\int\limits_{G} dz} \tag{9.4}$$

де G - підмножина елементів, яка максимізує C (рис.1.4, б). Для дискретного варіанта (C дискретне):

$$z_0 = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{9.5}$$

Критерій максимуму (Max-Criterion). Чітке значення вибирається довільно серед безлічі елементів, для яких С досягає максимуму:

$$z_0 \in \left\{ z : C(z) = \max_U C(U) \right\} \tag{9.6}$$

Висотна дефазифікація (Height defuzzification). Елементи області визначення Ω, для яких рівень ФП є меншим ніж деяке значення α, в розрахунок не приймаються, і чітке значення розраховується відповідно:

$$z_0 = \frac{\int_{C_\alpha} zC(z)dz}{\int_{C_\alpha} C(z)dz} \tag{9.7}$$

де C_α – нечітка множина - рівня α.

Нечеткое відношення визначається як будь-яке нечеткое підмножество упорядкованих кортежів, побудованих з елементів тех або інших базисних множин – універсумов. При цьому під Кортежем, так само, як і в разі звичайних множин, розуміється довільний набір або список упорядкованих елементів.

Нечітке ставлення. У загальному випадку нечітким k -арним ставленням, заданим на множинах (універсумах) $X_1, X_2, \ldots, X_k$, називається деяке нечітке підмножина декартового твору цих універсумів. Позначимо довільне нечітке ставлення через Q. За визначенням, $Q = \{< x_1, x_2, \ldots, x_k >, \mu_Q(< x_1, x_2, \ldots, x_k >)\}$, де $\mu_Q(< x_1, x_2, \ldots, x_k >)$ – функція приналежності даного нечіткого відношення, що визначається як відображення $\mu_Q : X_1 \times X_2 \times \ldots \times X_k \to [0, 1]$. Тут через $< x_1, x_2, \ldots, x_k >$ позначено кортеж з к-елементів, кожен з яких вибирається зі свого універсуму, тобто $x_1 \in X_1, x_2 \in X_2, \ldots, x_k \in X_k$.

Найпростіший тип нечітких відносин задається як Бінарне нечітке відношення між елементами двох універсальних множин. При цьому форму і вид функції належності нечіткого відношення попередньо ніяких обмежень не накладається.

Розглянемо кілька важливих окремих випадків нечітких відносин.

Порожнє нечітке ставлення. Теоретично нечітких відносин Порожнє Нечітке ставлення визначається як ставлення, яке містить жодного кортежу. Це ставлення позначається через $\varnothing$ і формально визначається як таке нечітке ставлення, функція належності якого тотожно дорівнює 0 на всіх елементах декартового твору його універсумів.

Повне нечітке ставлення Збігається зі звичайним повним ставленням, яке, своєю чергою, дорівнює, за визначенням, декартовим твором відповідних універсумів $X_1 \times X_2 \times \ldots \times X_k$. Функція належності повного нечіткого відношення тотожно дорівнює одиниці всім без винятку кортежів, тобто. $\mu_X(< x_1, x_2, \ldots, x_k >) \equiv 1$.

Бінарне нечітке ставлення дається на базисних множинах X_1, X_2 та визначається як нечітке відношення $Q = \{< x_i, x_j >, \mu_Q(< x_i, x_j >)\}$. Тут $\mu_Q(< x_i, x_j >)$ – функція приналежності бінарного нечіткого відношення, що визначається як відображення $\mu_Q : X_1 \times X_2 \to [0,1]$, а через $< x_i, x_j >$ позначений кортеж із двох елементів, причому $x_i \in X_1, x_j \in X_2$.

Зворотне нечітке ставлення. Якщо поставлено бінарне нечітке відношення Q на декартовому творі $X_1 \times X_2$, то зворотним до нього нечітким ставленням (позначається через Q^{-1}) називається таке бінарне нечітке відношення, яке задано на декартовому творі $X_2 \times X_1$, а функція приналежності якого визначається за такою формулою:

$$\mu_{Q^{-1}}(x_i, x_j) = \mu_Q(x_j, x_i), \quad x_i \in X_2, \quad x_j \in X_1. \tag{10.1}$$

Бінарне нечітке ставлення, задане на одній базовій множині (універсумі) X, визначається як нечітке відношення $Q = \{< x_i, x_j >, \mu_Q(< x_i, x_j >)\}$, де $\mu_Q(< x_i, x_j >)$ – функція приналежності бінарного нечіткого відношення, що визначається як відображення $\mu_Q : X \times X \to [0,1]$. Тут через $< x_i, x_j >$ позначений кортеж із двох елементів, причому $x_i \in X, x_j \in X$.

Насправді використовуються різні способи, якими може бути формально задані ті чи інші нечіткі відносини. Найбільшого поширення їх отримали такі.

У формі Списку з безпосереднім перерахуванням усіх кортежів нечіткого відношення та відповідних цим кортежам значень функції належності: $Q = \{(w_1, \mu_Q(w_1)), (w_2, \mu_Q(w2)), ..., (w_q, \mu_Q(w_q))\}$, де w_i – i-й кортеж $< x_1, x_2, ..., x_k >$ елементів цього відношення; q – число кортежів нечіткого відношення Q. При цьому для скорочення такого запису кортежі з нульовими значеннями функції приналежності у списку зазвичай не вказуються. Зрозуміло, що цей спосіб доцільно застосовувати лише для завдання нечітких відносин із кінцевим та невеликим числом кортежів q.

Приклад 1. Ставлення задано:

$$Q = \{(<1,1>, 0), (<1,2>, 0.1), (<2,1>, 0.5), (<2,2>, 1)\}.$$

Аналітично у формі деякого математичного виразу, що забезпечує можливість обчислення значення функції належності нечіткого відношення для кожного з кортежів. Цей дуже зручний спосіб може бути застосований для завдання нечітких відносин як із кінцевим, так і з нескінченним числом кортежів. При його використанні нечітке відношення записується у вигляді:

$$Q = \{< x_1, x_2, ..., x_k >, \mu_Q(< x_1, x_2, ..., x_k >)\},$$

де $\mu_Q(< x_1, x_2, ..., x_k >) = f(x_1, x_2, ..., x_k)$ – деяка задана функція змінних, що відповідає стандартним вимогам до функцій приналежності.

Приклад 3.2. Ставлення поставлено аналітично:

$$Q = \{< x_1, x_2, x_3 >, \mu_Q(< x_1, x_2, x_3 >) = e^{-(x_1 + 2x_2 + 3x_3)}, x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\}$$

.

Для завдання бінарних відносин, крім перелічених, можуть додатково використовуватися деякі інші способи.

Графічно У формі деякої поверхні чи сукупності окремих точок у тривимірному просторі. При цьому дві координати (незалежні змінні) будуть відповідати значенням елементів з універсумів , а третя координата – функції приналежності зі значеннями з інтервалу [0,1].

У формі Матриці нечітких відносин. При цьому нечітке бінарне відношення з кінцевим числом кортежів описується з використанням матриці, першому рядку якої відповідають перші елементи кортежів, а першому стовпцю – другі елементи кортежів. Елементами матриці є відповідні значення функції належності цього відношення. Якщо бінарне нечітке ставлення задається одному універсумі, то матриця такого відношення є квадратною.

Різні типи нечітких відносин визначаються за допомогою властивостей, аналогічних властивостям звичайних відносин:

1. Нечітке ставлення $\tilde{R}$ на $X \times X$ називається Рефлексивним, якщо $\mu_{\tilde{R}}(x,x) = 1$ $(\tilde{R}: x \approx y)$.

2. Нечітке ставлення $\tilde{R}$ на $X \times X$ називається Антирефлексивним, якщо $\mu_{\tilde{R}}(x,x) = 0$ $(\tilde{R}: x >> y)$.

3. Нечітке ставлення $\tilde{R}$ на $X \times Y$ називається Симетричним, якщо $\mu_{\tilde{R}}(x,y) = \mu_{\tilde{R}^{-1}}(y,x)$ (матриця відношення має бути симетричною).

4. Нечітке ставлення $\tilde{R}$ називається Транзитивним, якщо $\mu_{\tilde{R}}(x,z) \geq \max_{y}\left[\min\left(\mu_{\tilde{R}}(x,y), \mu_{\tilde{R}}(y,z)\right)\right]$ для будь-яких $x,y,z \in X$ и $(x,y),(y,z),(x,y) \in X \times X$.

Якщо $\tilde{R}$ – нечітке відношення «набагато більше», то протилежне йому відношення є нечітким ставленням «набагато менше».

Для нечітких множин є різні узагальнення цих властивостей, наприклад, слабка або сильна рефлексивність та ін.

Види нечітких відносин:

1. Відношення подібності або нечітке ставлення еквівалентності – це нечітке бінарне відношення, що має властивості рефлексивності, симетричності та транзитивності.

2. Нечітке ставлення порядку – рефлексивне, симетричне, транзитивне нечітке бінарне ставлення.

3. Нечітке ставлення подібності – рефлексивне, симетричне нечітке бінарне ставлення.

Лекція 11 «Нечіткі множини в системах управління»

В останній час дуже актуальна задача комп'ютерного управління поведінкою механічних або інших об'єктів. Якщо розглядати ці об'єкти і керувати ними комп'ютером як єдине ціле, то цю задачу можна сформулювати як спробу створення об'єкта, здатного самостійно керувати своїми діями. Речь может идти про регулювання різного ступеня складності - від виконання наступних команд типу «при нажаті кнопки виконати процедуру автоматичного запуску двигуна» до інтелектуального управління, при якому об'єкт може безпілотно (без участі людини) виконувати дії, спрямовані на досягнення визначених цілей.

Нечітка логіка застосовується в системах автоматичного управління для різних цілей і умовно можна виділити дві сфери її практичної реалізації:

- нечіткі логічні регулятори, які функціонують у прямому контурі та виконують функції деякого лінійного перетворювача, у тому числі вони можуть реалізовувати лінійні функції на кшталт П, ПІ, ПІД та ін. регуляторів;

- комбіновані нечіткі регулятори, у яких у прямому контурі функціонують традиційні регулятори, а додатковому контурі є нечіткі системи, які адаптують коефіцієнти посилення регулятора прямого контуру, підлаштовуючи їх до умов функціонування об'єкта управління, що змінюються.

При керуванні складними нелінійними об'єктами зазвичай використовують два підходи. При першому намагаються описати об'єкт за допомогою різних математичних моделей. Однак, у ці моделі входить досить багато емпіричних коефіцієнтів, які, як правило, змінюються у широкому діапазоні та їх ідентифікація є складним науково-технічним завданням. З іншого боку, виникають проблеми забезпечення стійкості обчислювального процесу.

При такому підході системи управління не забезпечують потрібної якості управління і, як наслідок, не вдається отримати готову продукцію з високими споживчими властивостями. Прикладами таких об'єктів є доменна піч у металургії, колона ректифікації при переробці нафти і т.д.

Проте з керуванням такими складними об'єктами людина справляється доволі впевнено. При цьому він використовує показання контрольно-вимірювальних приладів та за евристичними алгоритмами, що ґрунтуються на досвіді та інтуїції, впливає на виконавчі органи об'єкта, домагаючись прийнятних кінцевих результатів. При цьому чим кваліфікованіший оператор, тим кращих результатів він досягає. У цьому можливий другий підхід при управлінні складними об'єктами, коли евристичні алгоритми управління реалізуються з використанням мови нечіткої логіки, яка за своєю структурою близька до природної мови. Вперше цей підхід у вигляді нечіткої системи управління було реалізовано під час управління цементною піччю.

Нижче дається найпростіший приклад нечіткого управління підйомно-транспортним механізмом, коли спеціалізованою мовою формалізуються дії оператора для запобігання розгойдування вантажу на гаку підйомно-транспортного механізму на початку його переміщення, зміні швидкісних режимів або зупинці.

Покажемо попередньо, що об'єкт, яким керує кранівник, є нелінійним динамічним об'єктом. Найпростішою одновимірною математичною моделлю розгойдування вантажу при русі крана є маятник з рухомою точкою підвісу (рис.11.1) Баланс моментів щодо точки О дає

$$M_1+M_2=M_3,$$

де $M_1 = J_r \cdot \ddot{\alpha}(t)$ - момент інерції вантажу щодо точки підвісу;

$M_2 = m\left[V_r(t)\cos\alpha(t)\cdot l\right]_t'$ - момент, що створюється складовою швидкості підвісу, щодо точки підвісу;

$M_3 = mgl\sin\alpha(t)$ - момент, що створюється складовою ваги вантажу, щодо точки підвісу. Після інтегрування та перетворень отримаємо:

$$J_r\cdot\frac{1}{\cos\alpha(t)}\dot{\alpha}(t)+mgl\cdot\frac{1}{\cos\alpha(t)}\int\sin\alpha(t)dt=-mlV_T(t),$$

$$\alpha(t=0)=\alpha_0,\quad \int\sin\alpha(t)dt\,/_{t=0}=\beta_0$$

- початкові умови нелінійного інтегро-диференціального рівняння

Завдання управління нелінійним об'єктом полягає у визначенні швидкості $V_0(t)$ за вимірами його поточної швидкості $V_T(t)$ та кута розгойдування $\alpha(t)$ для того, щоб запобігти небезпечному розгойдування вантажу при зміні швидкісних режимів крана (рис.4.2): початок розгону - т.O_1; кінець розгону т.O_2; початок гальмування т.O_3; кінець гальмування т.O_4. Це завдання вирішується оператором крана евристичним способом. Один із можливих варіантів зміни швидкості $V_0(t)$ наприкінці гальмування (т.O_4.), при якому забезпечується: початок розгону - т.O_1; кінець розгону т.O_2; початок гальмування т.O_3; кінець гальмування т.O_4. Це завдання вирішується оператором крана евристичним способом. Один із можливих варіантів зміни швидкості $V_0(t)$ наприкінці гальмування (т.O_4.), при якому забезпечується: початок розгону - т.O_1; кінець розгону т.O_2; початок гальмування т.O_3; кінець гальмування т.O_4.

Це завдання вирішується оператором крана евристичним способом. Один із можливих варіантів зміни швидкості $V_0(t)$ наприкінці гальмування (т.O_4.), при якому забезпечується min $\alpha(t)$, показано на рис 11.2. При цьому оператор формулює для себе одне з можливих лінгвістичних правил:

R_i: якщо кут $\delta\alpha=\alpha_3-\alpha$, де α_3 - завдання, α - виміряне значення; трохи збільшується за годинниковою стрілкою та похідна кута $\delta\dot{\alpha}$ коливання вантажу трохи збільшується проти годинникової стрілки та швидкість $\delta V_T = V_{T3} - V_T$, де V_{T3} – завдання, V_T – виміряне значення; дорівнює нулю, тоді швидкість V_0 має бути невеликою в негативному напрямку щодо нуля. Визначимо для нечітких лінгвістичних змінних $\delta\alpha$, $\delta\dot{\alpha}$, δV_T, V_0 нечіткі множини з відповідними ідентифікаторами для функцій приладдя $\mu(\delta\alpha)$,$\mu(\delta\dot{\alpha})$, $\mu(V_T)$, $\mu(V_0)$ (рис.11.3). Наприклад, для $\mu(\delta\alpha)$ ці ідентифікатори мають вигляд:

- PM-кут розгойдування позитивний (проти годинникової стрілки) середній;
- PS-кут розгойдування позитивний невеликий;
- ZR - кут розгойдування нульовий;
- NS - кут розгойдування негативний (за годинниковою стрілкою) невеликий;
- NM – кут розгойдування негативний середній.

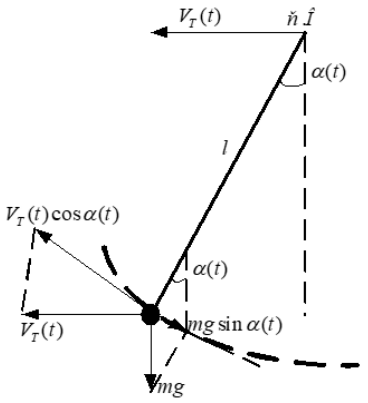


Рисунок 11.1 – Маятник із рухомою точкою про підвіс.

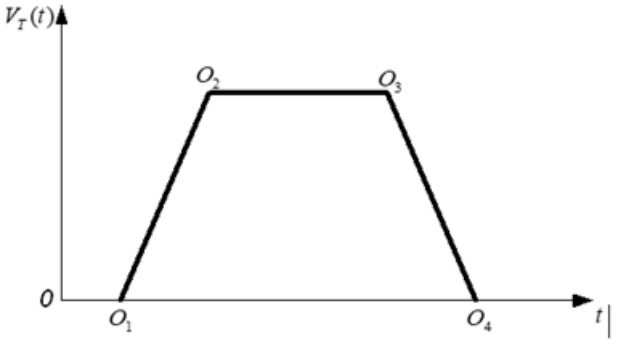


Рисунок 11.2 – Поточна швидкість $V_T(t)$ переміщення крана (т. O_1 - початок розгону; т. O_2 - кінець розгону; т. O_3 - початок гальмування; т. O_4 - кінець гальмування).

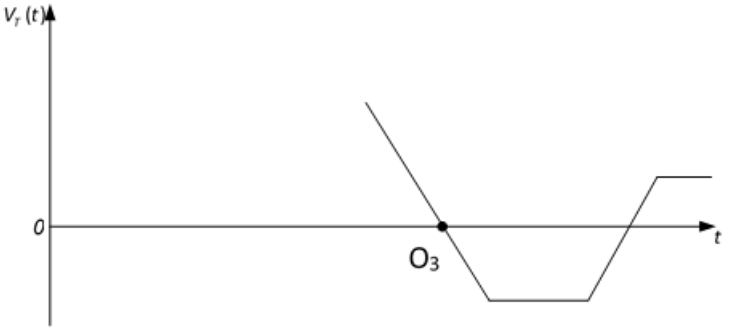


Рисунок 11.3 – Зміна швидкості крана для запобігання розгойдування вантажу.

Аналогічно визначаються ідентифікатори для μ(δα), μ($\delta\dot{\alpha}$), μ(V_0).

З урахуванням заданих нечітких множин база правил евристичного алгоритму керування оператором швидкістю $V_0(t)$ крана при $V_T(t)=0$(т.O_4, рис.11.2) має вигляд:

R_1: якщо δα=NS та $\delta\dot{\alpha}$ =NS та δV_T=ZR, то V_0=ZR

чи

R_2: якщо δα=NS та $\delta\dot{\alpha}$ =ZR та δV_T=ZR, то V_0=NS

чи

R_3: якщо δα=NS та $\dot{\delta\alpha}$ =PS та δV_T=ZR, то V_0=NS

чи

$\{R^i\}_{i=1}^{7}$ R_4: якщо $\delta\alpha$ =ZR та $\dot{\delta\alpha}$ =NS та δV_T =ZR, то V_0 =PS

R_5: якщо δα=ZR та $\dot{\delta\alpha}$ =ZR та δV_T=ZR, то V_0=ZR

чи

R_6: якщо δα=ZR та $\dot{\delta\alpha}$ =PS та δV_T=ZR, то V_0=NS

чи

R_7: якщо δα=PS та $\dot{\delta\alpha}$ =NS та δV_T=ZR, то V_0=PS.

Аналогічні засади правил можуть бути записані т.т. О1-О3 зміни швидкісних режимів крана (рис.11.2). Після їхнього об'єднання отримаємо формалізоване уявлення евристичного алгоритму, за допомогою якого оператор управляє швидкість переміщення крана для запобігання небезпечному розгойдування вантажу. Реалізація цього алгоритму як нечіткого контролера у прямому контурі управління дозволяють виключити оператора крана і передати його функції нечіткою системі регулювання.

Наведений приклад показує перевагу нечіткої системи регулювання нелінійним об'єктом у порівнянні з традиційною системою регулювання простотою своєї технічної реалізації. Блок схема нечіткої системи регулювання зображено на рис.11.4. Подібні нечіткі системи регулювання можуть бути розглянуті для підйомно-транспортних механізмів завантаження ракет у підводні човни, шахти, пускові платформи і т.д. з урахуванням впливу неконтрольованих факторів типу: вітрові навантаження, що впливають на вантаж; хвилювання поверхні води, що переміщує місце розташування посадки ракети і т.д. У всіх цих випадках дії досвідченого оператора можуть бути формалізовані і далі реалізовані як нечітка система регулювання.

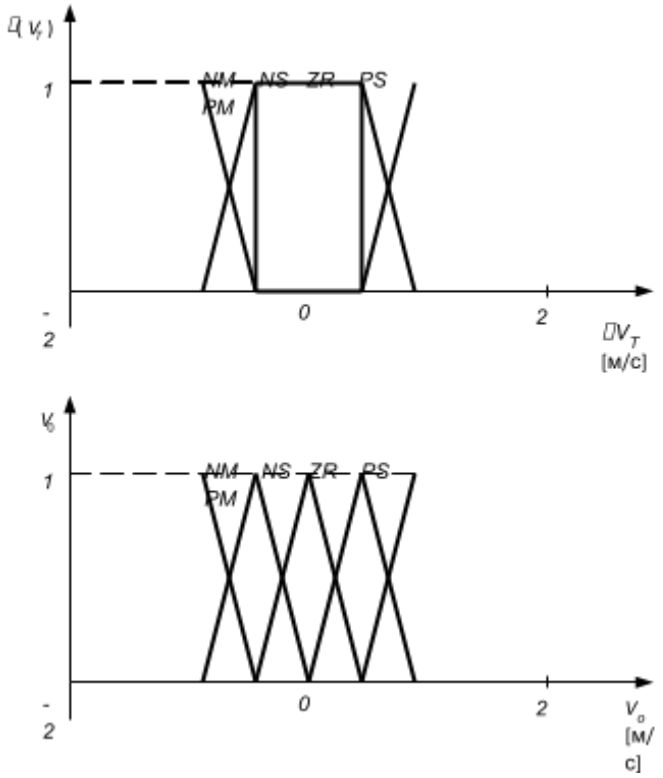


Рисунок 11.4 – Функції приналежності нечітких множин

Нечітке управління застосовується підтримки температури всередині «сорочки» хімічного реактора, зробленого за принципом термоса (рис.4.6). Хімічні реактори такого типу використовуються в медицині, хімії, у виробництві косметики і т.д., коли необхідно забезпечити плавність зміни температури в часі та одночасно рівномірність її зміни всередині реактора.

Температура всередині реактора управляється шляхом зміни температури робочого тіла всередині сорочки реактора. При управлінні за класичною схемою зазвичай використовують ПІД-регулятор. Однак використання нечіткого управління може підвищити точність регулювання, забезпечуючи таким способом вихід більш якісної продукції.

Основне завдання управління полягає у підтримці заданої температури всередині реактора, яка може змінюватися під впливом неконтрольованих збурень. Наприклад, в результаті додавання матеріалу 1 реактор може відбуватися хімічна реакція з виділенням або поглинанням тепла, тому необхідно відповідно або знижувати, або підвищувати температуру всередині «сорочки», швидко відновлюючи задану температуру всередині реактора. Метод керування з використанням ПІД-регулятора не може забезпечити автоматичне вирішення цього завдання через вплив неконтрольованих збурень. Як правило, при такому способі управління оператор візуально контролює температуру і потім змінює завдання ПІД-регулятора по режимних картах. Однак використання в системі управління блоку обробки нечітких висловлювань дозволяє повністю автоматизувати процедуру поточного налаштування ПІД-регулятора (рис.11.5).

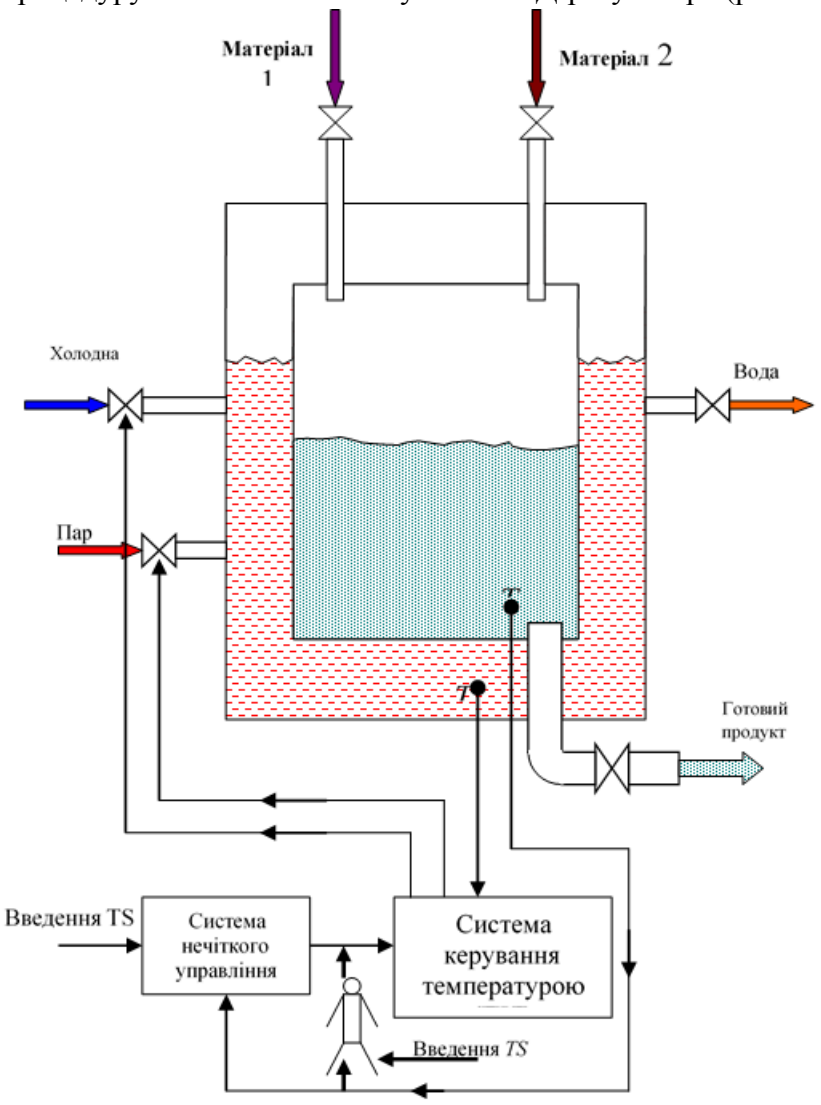


Рисунок 11.5 – Приклад нечіткого управління на базі ПІД для система керування температурою хімічного реактора (жирні стрілки – класичний контур керування температурою)

PyFuzzy була першою бібліотекою загального призначення для розробки FIS з використанням мови програмування Python (версія 2.7). Це залежить від середовища виконання ANTLR 3. PyFuzzy дозволяє керувати всім об'єктів, необхідних для побудови FIS і створення багатьох типів нечіткі множини. Він також підтримує експорт і спільний доступ до FIS за допомогою засобів файлів мови нечіткого керування (FCL). Файли FCL реалізують старий стандарт IEC 61131 (IEC61131-7), який був розроблений для додатків нечіткого управління і залишався протягом багатьох років єдиним де фактичний стандарт для представлення FIS.

Незважаючи на повноту, PyFuzzy тепер застарів і більше не підтримується, а Python 2 – більше не підтримується офіційно.

На сьогоднішній день програмне забезпечення, розроблене з використанням Python 3, включає ScikitFuzzy і Fuzzylab. Scikit-Fuzzy — API нечіткої логіки призначений для роботи в стеку scipy, який пропонує функції та класи для підтримки моделювання нечітких систем. Fuzzylab - це а нещодавно опублікована бібліотека Python 3, заснована на Octave Fuzzy Logic Toolkit, призначений для створення логічних контролерів. Обидві ці бібліотеки не підтримують визначення власного членства функції (надають лише попередньо реалізовані форми), ані системи висновків TakagiSugeno довільного порядку. Наразі ScikitFuzzy підтримує лише висновок Mamdani, тоді як Fuzzylab підтримує системи висновку Мамдані та Такагі-Сугено 0-го порядку. Крім того, обидві бібліотеки не забезпечують інтерфейс, близький до природної мови для визначення FIS. Наприклад, ані Fuzzylab, ані ScikitFuzzy не дозволяють визначати нечіткі правила як рядки тексту, написані природною мовою.

У своєму основному API Scikit-Fuzzy приймає префіксну нотацію (тобто оператори передують операндам), щоб визначити антецеденти правил. Розроблений пізніше API (розроблений для впровадження систем керування) також доступний, але він потребує маніпуляцій об'єктів лінгвістичної змінної та індексувати їх за допомогою лінгвістичних терміни для визначення нечіткого правила.

Більшість останніх програм із відкритим кодом для нечіткої логіки спрямовані на машинне навчання, класифікація та регресійний аналіз або рішення системи підтримки. Серед найпопулярніших можна знайти FuzzyR, інструментарій загального призначення для нечітких міркувань, реалізований у R і допоміжні FIS типу 1 і 2; FuzzyLite, набір бібліотек C++, розроблених для нечіткого керування; Fispro, C++ програмне забезпечення з графічним інтерфейсом користувача (GUI), розроблене для керованих даними додатків FIS та їх автоматичного навчання з набору даних; Juzzy, набір інструментів на основі Java для обробки типу 2 нечітких наборів і JT2FIS, бібліотеки класів Java для визначення ІНФ типу 2 інтервалу.

Octave Fuzzy Logic Toolkit є компонентом обчислювального середовища Octave для розробки та експлуатації FLC використовуючи мову сценаріїв Octave або формат FIS, обидва з яких здебільшого сумісні з аналогами Matlab. Обчислювальне середовище та інструментарій Octave безкоштовні з відкритим вихідним кодом, під ліцензією GNU General Public 3. Окрім керування нечіткою логікою, інструментарій забезпечує кластеризацію нечітких даних. Поточна версія інструментарію це 0.4.5, випущена у 2014 році. Набір інструментів містить наступны функції для FLC контролери: Мамдані та Такагі Сугено, функції приналежності: трикутник, трапеція, дзвін, гаусів, гаусів добуток, пі-форма, сигмоїдна різниця, сигмоїдний добуток, сигмоїд, s-форма, z-форма, постійна, лінійна та спеціальні функції. T-норми: мінімальна, добуток, обмежена різниця, різкий продукт, продукт Ейнштейна, продукт Хамахера та спеціальні функції. S-норми: максимальні, алгебраїчні сума, обмежена сума, різка сума, сума Ейнштейна, сума Хамахера, і спеціальні функції. Дефазифікатори: центроїд, бісектриса, найменший з максимумів, середнє з максимумів, найбільший з максимумів, середньозважене, зважена

сума та спеціальні дефазифікатори. Перелік програмних бібліотек для реалізації нечіткої логіки наведено у таблиці 12.1.

Таблиця 12.1 – Перелік програмних бібліотек для реалізації нечіткої логіки

Name	Language	Latest Release	Description
FuzzyLite	C + +	2017	Колекція бібліотек C++, розроблених для нечіткого керування, сумісних зі стандартом FCL
FisPro	C + +	2019	Програмне забезпечення загального призначення з графічним інтерфейсом користувача, призначене для полегшення навчання систем нечіткого висновку з даних
Juzzy	Java	2013	Набір інструментів на основі Java, що реалізує нечіткі міркування типу 2
JFML	Java	2018	Бібліотека Java, що реалізує стандарт FML
FuzzyR	R	2019	Набір інструментів R із графічним інтерфейсом користувача для проектування систем нечіткого висновку типу 1 і 2
Fuzzy Toolbox for Matlab	Matlab	1994	Набір інструментів загального призначення, реалізований у середовищі Matlab
Fuzzy Logic Toolbox	Matlab	2020	Комерційно розповсюджений інструментарій, забезпечений графічним інтерфейсом і доступний у середовищі Matlab
PyFuzzy	Python 2	2014	Бібліотека Python 2, сумісна зі стандартом FCL. Розробку бібліотеки було припинено, і Python 2 більше не підтримується офіційно. Ця бібліотека залежить від середовища виконання ANTLR 3
Fuzzylab	Python 3	2019	Бібліотека Python на основі Octave Fuzzy Logic Toolkit
Scikit-Fuzzy	Python 3	2019	API загального призначення, призначений для роботи в стеку scipy, пропонуючи класи та методи для підтримки визначення нечітких систем
Py4JFML	Python 3	2019	Обгортка Python для бібліотеки Java JFML

Однією з найновіших і найцікавіших реалізацій у нечіткому співтоваристві є JFML бібліотека, єдина бібліотека з відкритим кодом (реалізована на Java), включає в себе останній розроблений стандарт для представлення FIS, стандарт IEEE 1855-2016, який визначає нову розширену мову розмітки W3C під назвою Fuzzy Markup Language (FML). Примітно, що також була випущена оболонка Python для JFML, що дозволяє визначити FML-сумісні FIS за допомогою скриптів Python і бібліотеки JFML. Однак слід зазначити що це рішення потребує фреймворку Py4J, який необхідний для інтерпретатора Python для динамічного доступу до об’єктів Java у віртуальній машині Java.

Можна завантажити вихідний код з GitHub:

```
git clone https://github.com/pcingola/jFuzzyLogic.git
```

Щоб створити, треба запуститит ant з папки проекту.

jFuzzyLogic надає кілька параметрів командного рядка, корисних для розробки. Розглянемо кілька типових прикладів використання.

Запуск jFuzzyLogic без жодного параметра командного рядка покаже просту довідку.

```
java -jar jFuzzyLogic.jar

Usage: java -jar jFuzzyLogic.jar [options]

Options:
   file.fcl                        : Load FCL file and show
memebership functions.
  -c file.fcl                      : Compile. Generate C++
code from FCL file (to STDOUT)
  -e file.fcl in_1 in_2 ... in_n : Load FCL file, assign
inputs i_1, i_2, ..., i_n and evaluate.
  demo                             : Run a demo exmaple
(tipper.fcl)
```

Проста демонстрація доступна за допомогою демо параметра командного рядка. Він покаже анімацію прикладу tipper.fcl:

```
java -jar jFuzzyLogic.jar demo
```

Використовуючи файл FCL, jFuzzyLogic покаже всі функції членства для всіх вхідних і вихідних змінних.

```
java -jar jFuzzyLogic.jar myfile.fcl
```

Ви також можете вказати вхідні змінні за допомогою параметра командного рядка -e, jFuzzyLogic обчислить і покаже дефазифіковані вихідні змінні, а також ступінь підтримки кожного правила. Після параметра командного рядка -e повинні вказуватися значення вхідних змінних.

```
java -jar jFuzzyLogic.jar -e tipper.fcl 8.5 9
FUNCITON_BLOCK tipper
  VAR_INPUT         food = 8.500000
  VAR_INPUT      service = 9.000000
  VAR_OUTPUT         tip = 24.999984
  RULE_BLOCK No1
    Support   Rule name   Rule
    0.000000  1           IF (service IS poor) OR (food IS
rancid) THEN tip IS cheap;
    0.000000  2           IF service IS good THEN tip IS
average;
    0.750000  3           IF (service IS excellent) AND
(food IS delicious) THEN tip IS generous;
```

Мова нечіткого керування (FCL) — це мова, визначена в специфікації IEC 61131, частина 7. Очевидно, що наявність стандартної мови для програмування нечітких систем має багато переваг. До FCL кожна реалізація бібліотеки нечіткої логіки мала різні способи визначення функцій членства, правил тощо. Більшу частину часу вам доводилося вивчати API, які були несумісні від однієї системи до іншої. Крім того, використання API для створення систем є громіздким і схильним до помилок.

FCL просто знімає всі ці проблеми. Крім того, FCL досить простий у вивченні, і більшість людей вивчають основи, подивившись на кілька прикладів.

Наступний приклад коду FCL для класичної проблеми "самоскид". Проблема полягає в тому, як розрахувати чайові в ресторані (очевидно, це іграшковий приклад тривіальної задачі, еквівалентний прикладу "привіт, світ!" в інших мовах програмування):

```
     FUNCTION_BLOCK tipper    // Block definition (there may be
more than one block per file)
     // Define input variables
     VAR_INPUT
     service : REAL;
     food : REAL;
     END_VAR
     // Define output variable
     VAR_OUTPUT
     tip : REAL;
     END_VAR
     // Fuzzify input variable 'service': {'poor', 'good' ,
'excellent'}
     FUZZIFY service
     TERM poor := (0, 1) (4, 0) ;
     TERM good := (1, 0) (4,1) (6,1) (9,0);
     TERM excellent := (6, 0) (9, 1);
     END_FUZZIFY
     // Fuzzify input variable 'food': { 'rancid', 'delicious' }
     FUZZIFY food
     TERM rancid := (0, 1) (1, 1) (3,0) ;
     TERM delicious := (7,0) (9,1);
     END_FUZZIFY
     // Defzzzify output variable 'tip' : {'cheap', 'average',
'generous' }
     DEFUZZIFY tip
     TERM cheap := (0,0) (5,1) (10,0);
     TERM average := (10,0) (15,1) (20,0);
     TERM generous := (20,0) (25,1) (30,0);
     METHOD : COG;       // Use 'Center Of Gravity'
defuzzification method
     DEFAULT := 0;       // Default value is 0 (if no rule
activates defuzzifier)
     END_DEFUZZIFY
     // Inference rules
     RULEBLOCK No1
     AND : MIN;     // Use 'min' for 'and'
     ACT : MIN;     // Use 'min' activation method
     ACCU : MAX;    // Use 'max' accumulation method
     RULE 1 : IF service IS poor OR food IS rancid THEN tip IS
cheap;
     RULE 2 : IF service IS good THEN tip IS average;
     RULE 3 : IF service IS excellent AND food IS delicious THEN
tip IS generous;
     END_RULEBLOCK
     END_FUNCTION_BLOCK
```

Щоб запустити це треба завантажити файл FCL tipper.fcl і виконати таку команду:

```
java -jar jFuzzyLogic.jar tipper.fcl
```

Ця команда розбере та завантажить код FCL і покаже функції членства (продовжуйте читати наступний підрозділ для отримання додаткової інформації). Приклад використання jFuzzyLogic наведено на рис.12.1.

Щоб використовувати jFuzzyLogic у своїх проектах, треба додати jFuzzyLogic.jar як один зі своїх JAR-файлів (бібліотек). Це робиться так само, як і з будь-яким іншим файлом JAR.

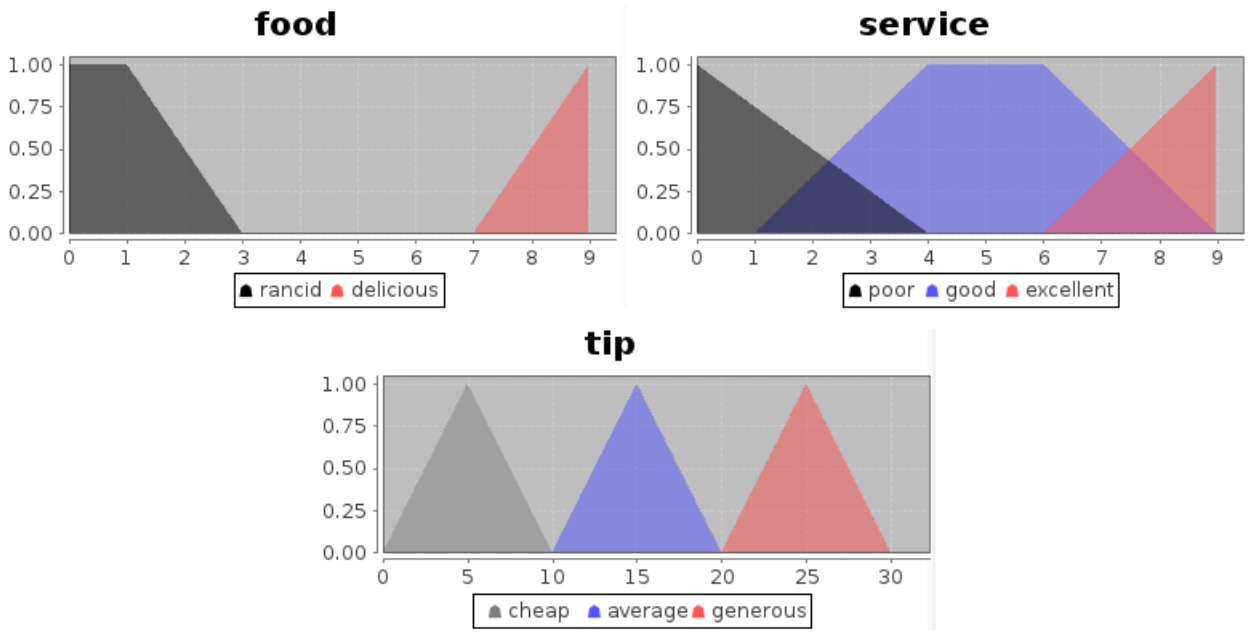


Рисунок 12.1 – Приклад використання jFuzzyLogic

Лекція 13 «Можливості бібліотеки Fuzzylab »

Fuzzylab це бібліотека нечіткої логіки Python, заснована на Octave Fuzzy Logic Toolkit 0.4.6, вважається переважно сумісним із MATLAB набором інструментів нечіткої логіки для Octave.

Репозиторій бібліотеки на GitHub наведено на рис.13.1.

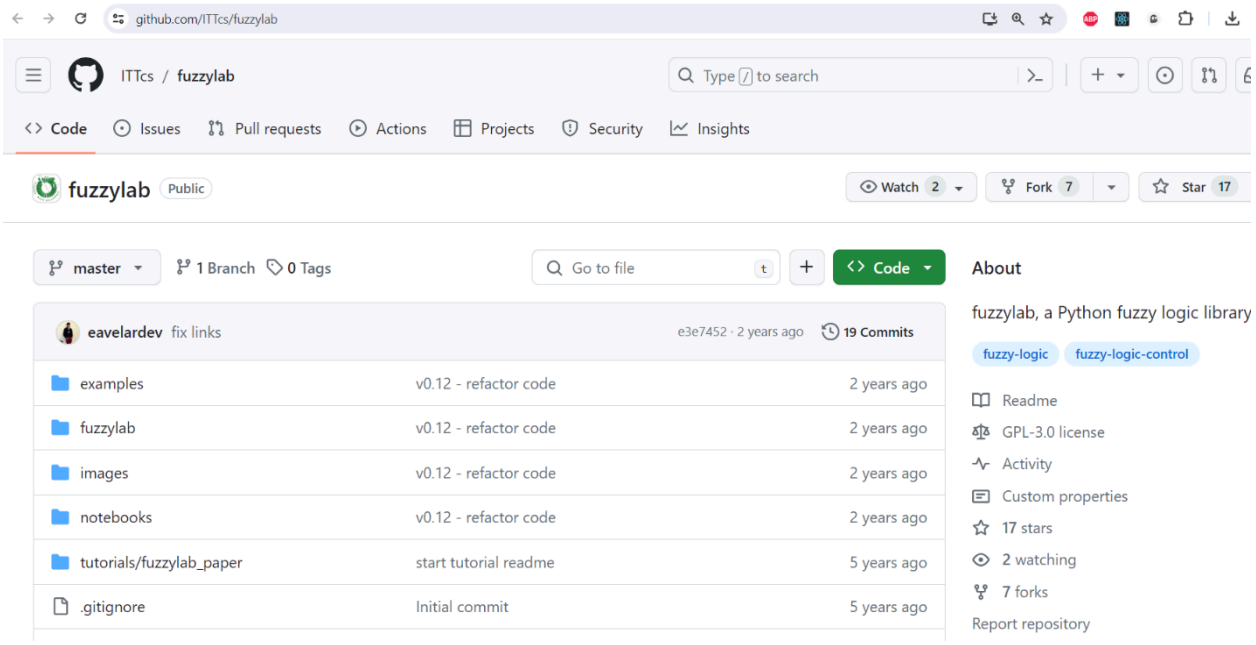


Рисунок 13.1 – Репозиторій бібліотеки на GitHub

Спосіб використання функцій fuzzylab базується на функціях Matlab R2022a Fuzzy Logic Toolbox.

Fuzzylab використовує матрицю, яку потрібно ввести користувача, щоб визначити повну базу правил.

Установка бібліотеки виконується наступним чином

```
pip install fuzzylab
```

Результат імпорту бібліотеки у Jupiter Notebook наведено на рис.13.2.

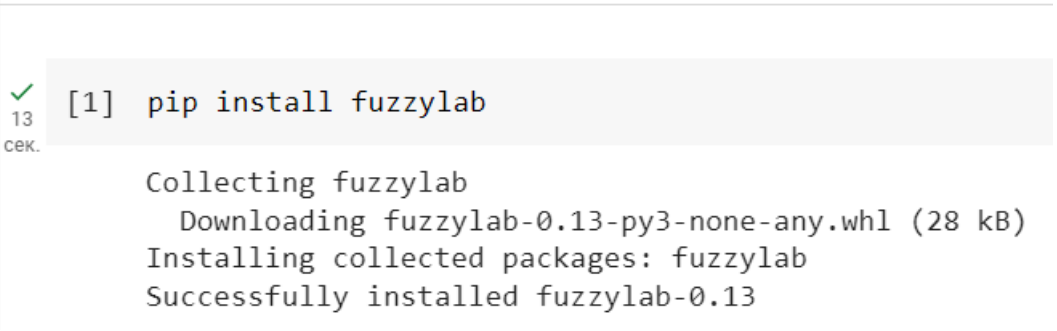


Рисунок 13.2 – Результат імпорту бібліотеки у Jupiter Notebook

Використання. У папці прикладів є кілька фрагментів коду та еквівалентний код matlab для початкової точки використання бібліотеки.

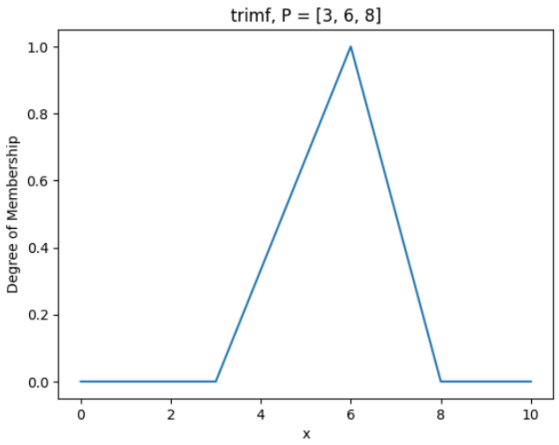
Ми використовуємо fl як псевдонім для fuzzylab.

```
import fuzzylab as fl
Example
import fuzzylab as fl
import matplotlib.pyplot as plt
x = fl.arange(0, 0.1, 10)
y = fl.trimf(x, [3, 6, 8])
plt.plot(x,y)
plt.title('trimf, P = [3, 6, 8]')
plt.xlabel('x')
plt.ylabel('Degree of Membership')
plt.ylim([-0.05, 1.05])
plt.show()
```

Також ми можемо використовувати стиль коду MATLAB (рис.13.3).

```
from fuzzylab import arange, trimf
from matplotlib.pyplot import plot, title, xlabel, ylabel,
ylim, show

x = arange(0, 0.1, 10)
y = trimf(x, [3, 6, 8])

plot(x,y)
title('trimf, P = [3, 6, 8]')
xlabel('x')
ylabel('Degree of Membership')
ylim([-0.05, 1.05])
show()
```

```
import fuzzylab as fl
import matplotlib.pyplot as plt

x = fl.arange(0, 0.1, 10)
y = fl.trimf(x, [3, 6, 8])

plt.plot(x,y)
plt.title('trimf, P = [3, 6, 8]')
plt.xlabel('x')
plt.ylabel('Degree of Membership')
plt.ylim([-0.05, 1.05])
plt.show()
```




```
from fuzzylab import arange, trimf
from matplotlib.pyplot import plot, title, xlabel, ylabel, ylim, show
x = arange(0, 0.1, 10)
y = trimf(x, [3, 6, 8])
plot(x,y)
title('trimf, P = [3, 6, 8]')
xlabel('x')
ylabel('Degree of Membership')
ylim([-0.05, 1.05])
show()
```

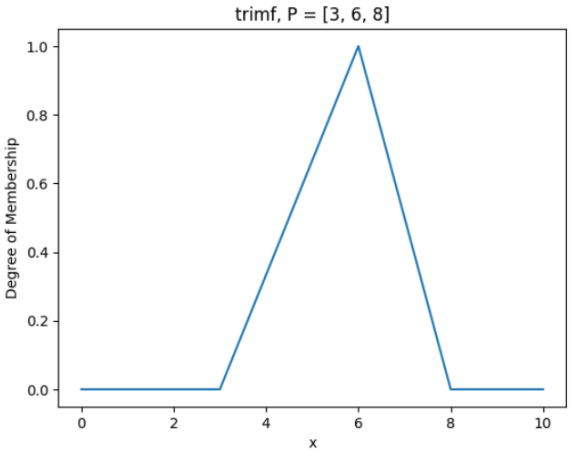


Рисунок 13.3 – Результат застосування бібліотеки у Jupiter Notebook

Від користувача також вимагається пов’язати значущий лінгвістичний термін з кожним нечітким набором. Визначені нечіткі набори потім використовуються в створення конкретних об'єктів LinguisticVariable, забезпечених їх власні імена, надані користувачем. Якщо нечіткі множини задані означає лише послідовність точок, Simpful автоматично визначає межі всесвіту дискурсу, використовуючи мінімальне та максимальне значення серед перших координат усіх точки, які визначають усі нечіткі множини. Навпаки, якщо нечіткий набір визначається за допомогою спеціальних функцій, які користувач може вказати дійсний інтервал значень для об’єкта LinguisticVariable за допомогою необов'язкового аргументу universe_of_discourse.

Визначення цього інтервалу потрібно для використання методу draw() класу LinguisticVariable, який можна використати для побудови нечітких наборів, таким чином

дозволяючи швидко перевірити та налагодити їх впровадження. Заслуговує на увагу нечіткість як на основі точок, так і на основі функцій набори можуть бути використані одночасно у визначенні однієї лінгвістичної змінної. Щоб полегшити його використання, також надає клас AutoTriangle(), який повертає об’єкт LinguisticVariable що підрозділяється на визначену користувачем кількість нормалізованих трикутних нечітких наборів.

Кожне правило має використовувати імена змінних і лінгвістичні терміни, які були визначені в об’єктах LinguisticVariable.

У випадку систем Такагі–Сугено, послідовність правил має також використовувати рядки, пов’язані з вихідними чіткими значеннями або з функціями виводу, визначеними користувачем. В цьому випадку, користувач повинен визначити функції, які використовуються в висновку як слідує:

1. Функції 0-го порядку (тобто постійні функції) визначаються як «виводити чіткі значення».

2. Для систем Такагі–Сугено вищого порядку користувач може визначити «функції виводу» як рядки тексту, що включають лінгвістичні змінні. Ці функції оцінюються під час виконання використовуючи функцію eval() Python, яка аналізує вираз, заданий як рядковий аргумент, і виконує його як код в рамках програми.

Необхідно виводити чіткі значення, виводити функції та виводити нечіткі набори мають асоційований значущий і унікальний рядок для їх ідентифікації, які будуть використані у визначенні нечітких правил.

Лінгвістичні змінні, нечіткі правила, вихідні чіткі значення, вихідні функції та вихідні нечіткі набори додаються до об’єкта нечіткої системи, який реалізує весь FIS. Враховуючи вхідні значення для змінні, що з’являються в антецедентах нечітких правил, методи, що реалізують висновок Мамдані або Такагі–Сугено, можуть бути викликається для однієї або кількох змінних, що з’являються в консеквенті нечітких правил, щоб отримати їх остаточний результат. В якості альтернативи користувач може викликати метод inference() класу FuzzySystem, щоб дозволити Simpful вибрати та використати найбільш відповідний метод висновку (наприклад, Мамдані або Такагі–Сугено).

Результати висновку повертаються користувачеві як ключ-значення пари всередині словника, де ключі представляють імена змінних.

У лістингу 1 проблема чайових моделюється як FIS Мамдані. створюється об'єкт нечіткої системи. Нечіткі множини та лінгвістична змінна «Сервіс», вони містить три нечітких набори, «погано», «добре» і «відмінно», починаючи від 0 до 10. Далі лінгвістична змінна якості їжі: визначено, використовуючи два нечітких набори, «прогірклий» і «смачний». Вихідна змінна «Tip» та її нечіткі набори визначені далі.

Усі нечіткі множини, використані в цьому прикладі, є трикутними (отже, використання попередньо реалізованої функції Triangular_MF), за винятком нечітких наборів, які є прикладом трапецієподібного набору (Trapezoidal_MF).

```
    A simple fuzzy inference system for the tipping problem
    # Create a fuzzy system object
    FS = FuzzySystem()
    # Define fuzzy sets and linguistic variables
    S_1   =   FuzzySet(function=Triangular_MF(a=0,   b=0,   c=5),
term=”poor”)
    S_2   =   FuzzySet(function=Triangular_MF(a=0,   b=5,   c=10),
term=”good”)
    S_3   =   FuzzySet(function=Triangular_MF(a=5,   b=10,   c=10),
term=”excellent”)
```

```
    FS.add_linguistic_variable(”Service”,
LinguisticVariable([S_1, S_2, S_3], concept=”Service quality”,
universe_of_discourse=[0,10]))
    F_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=10),
term=”rancid”)
    F_2 = FuzzySet(function=Triangular_MF(a=0, b=10, c=10),
term=”delicious”)
    FS.add_linguistic_variable(”Food”, LinguisticVariable([F_1,
F_2], concept=”Food quality”, universe_of_discourse=[0,10]))
    # Define output fuzzy sets and linguistic variable
    T_1 = FuzzySet(function=Triangular_MF(a=0, b=0, c=10),
term=”small”)
    T_2 = FuzzySet(function=Triangular_MF(a=0, b=10, c=20),
term=”average”)
    T_3 = FuzzySet(function=Trapezoidal_MF(a=10, b=20, c=25,
d=25), term=”generous”)
    FS.add_linguistic_variable(”Tip”, LinguisticVariable([T_1,
T_2, T_3], universe_of_discourse=[0,25]))
    # Define fuzzy rules
    R1 = ”IF (Service IS poor) OR (Food IS rancid) THEN (Tip IS
small)”
    R2 = ”IF (Service IS good) THEN (Tip IS average)”
    R3 = ”IF (Service IS excellent) OR (Food IS delicious) THEN
(Tip IS generous)”
    FS.add_rules([R1, R2, R3])
    # Set antecedents values
    FS.set_variable(”Service”, 4)
    FS.set_variable(”Food”, 8)
    # Perform Mamdani inference and print output
    print(FS.Mamdani_inference([”Tip”]))
```

Лекція 14 «Гібридизація м’яких обчислень»

Подібно до того, як нечіткі множини розширили рамки класичної математичної теорії множин, нечітка логіка вторглася практично в більшість методів Data Mining, наділивши їх новою функціональністю.

Нині є кілька алгоритмів реалізації процедури нечіткого висновку, які відрізняються характером використовуваних правил, типами імплементованих логічних операцій та математичними підходами до процедури дефазифікації.

У практичному та прикладному сенсі доцільність використання НМ та НЛ полягає в їх поєднанні з іншими системами та підходами в рамках штучного інтелекту.

Типові нечіткі системи на сьогоднішній день володіють набором недоліків, до яких у тому числі належать складності створення та формалізації логічних правил та ФПР за допомогою пошуку та залучення фахівців та експертів аналізованої предметної галузі. Завдяки використанню методик створення адаптивних та гібридних нечітких систем стає можливим автоматизація цього процесу шляхом системного підбору всіх параметрів у процесі навчання моделей на зазначених експериментальних даних.

Гібридні нечіткі системи є комбінацією нечіткої логіки з іншими методами та технологіями, такими як нейронні мережі, генетичні алгоритми, логіка ведення журналів, експертні системи тощо. Мета таких систем — використовувати переваги різних методів для створення більш ефективних і потужних рішень у порівнянні із використанням одного методу.

Ось кілька прикладів гібридних нечітких систем:

1. Гібридні нечіткі нейромережні системи. Комбінація нечіткої логіки та нейронних мереж може покращити здатність моделювання та навчання систем. Наприклад, можна використовувати нечітку логіку для інтерпретації результатів нейронної мережі або впроваджувати нечіткі правила до архітектури нейромережі.

2. Генетичні нечіткі системи. Генетичні алгоритми можуть бути використані для оптимізації параметрів нечіткої системи. Генетичні алгоритми можуть підбирати оптимальні функції належності та правила, що дозволяє покращити продуктивність системи.

Приклад структурної схеми гібридної нейро-нечіткої системи із застосуванням генетичного алгортму наведено на рис.14.1.

3. Еволюційні нечіткі системи. Використання еволюційних алгоритмів, таких як генетичні алгоритми або алгоритми оптимізації, для створення та покращення структури нечітких правил та функцій належності.

4. Нечітко-мережевий підхід. Поєднання принципів нечіткої логіки та нейромережевих методів для створення систем, здатних адаптуватися до умов, що змінюються, і навчатися на основі даних.

5. Гібридні нечіткі експертні системи. Комбінування нечіткої логіки з методами експертних систем для більш точних та інтерпретованих рішень. Експертні системи можуть надавати знання та досвід, а нечітка логіка – способи обробки невизначеності.

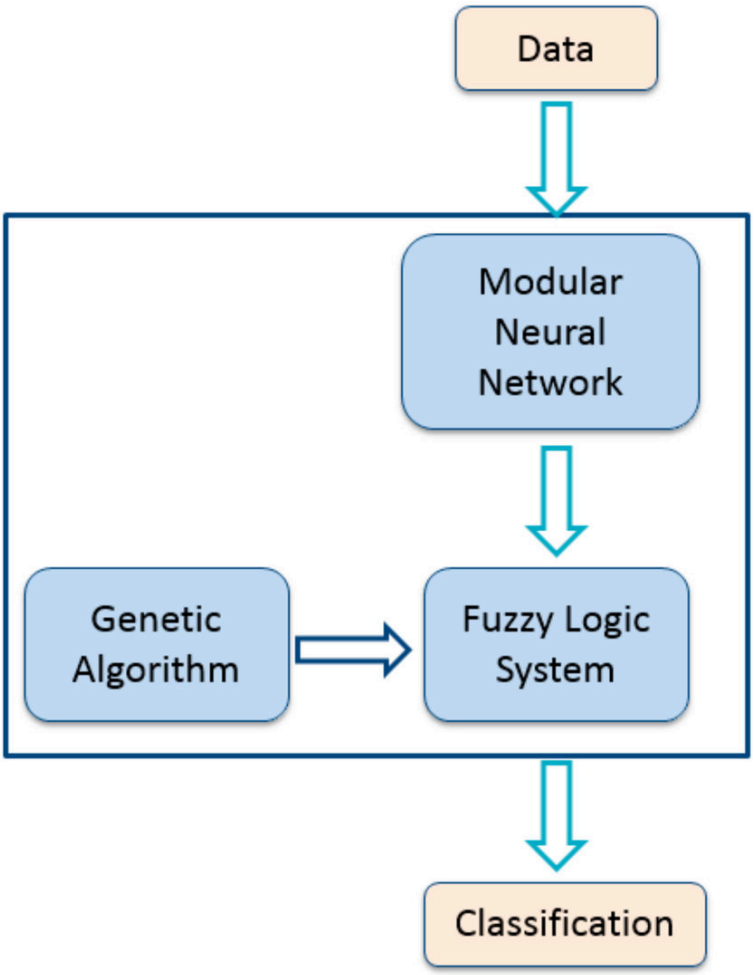


Рисунок 14.1 – Структурна схема гібридної нейро-нечіткої системи із застосуванням генетичного алгортму

Схема використання штучної нейромережі в складі нечіткої системи управління даними наведена на рис.14.2.

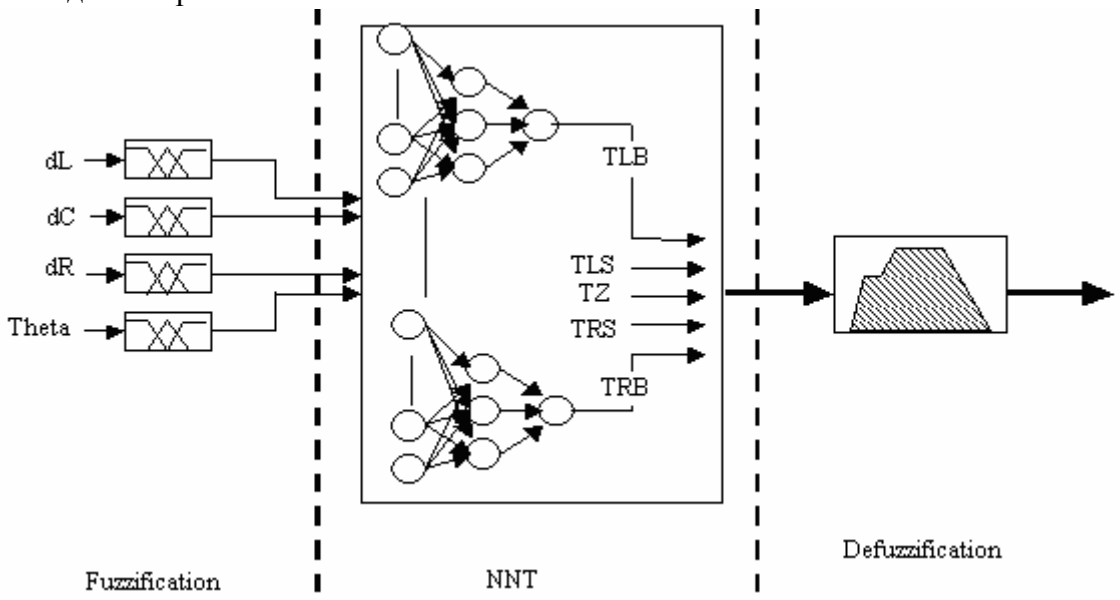


Рисунок 14.2 – Схема використання штучної нейромережі в складі нечіткої системи управління даними

6. Гібридні нечіткі системи управління. Використання нечіткої логіки разом із класичними методами управління підвищення ефективності системи управління за умов змінних і невизначених зовнішніх чинників.

Гібридні нечіткі системи часто застосовуються у складних та динамічних середовищах, де невизначеність, зміни та адаптація до змінних умов є ключовими аспектами. Ці системи дозволяють об'єднати переваги різних методів та створити більш гнучкі та ефективні рішення.

Використовувані алгоритми навчання нечітких систем з елементами адаптивності є складнішими і вимогливішими до обчислювальних ресурсів, на відміну алгоритмів зворотного поширення помилки, їх реалізація передбачає:

- формування пов'язаних лінвістичних змінних;
- динамічне коригування функцій приналежності.

Перша з зазначених завдань належить до виду перебірної процедури аналізу даних, друга дозволяє провести оптимізацію залежностей функцій у межах безперервних просторів. У цій ситуації формується деяке протиріччя, що виникає через необхідність створення наборів нечітких правил за спеціальними ФПР, а під час проведення самих етапів нечіткого висновку формування БП. У такій ситуації, коли набір нечітких правил генерується автоматизованим необхідним є процес забезпечення їхньої повноти та несуперечності.

Проаналізуємо актуальні напрями застосування НЛ у зв'язці з іншими завданнями та гібридними технологіями.

1. Нечіткі запити до реляційних сховищ та баз даних (БД), цільова орієнтованість даного підходу полягає у формулюванні гнучких запитів щодо вибірки даних природною мовою, розширюючи можливості SQL мови. що неможливе при використанні стандартного механізму запитів. Реалізація цього підходу можлива шляхом залучення апарату нечіткої реляційної алгебри та програмних розширень для використовуваних СУБД.

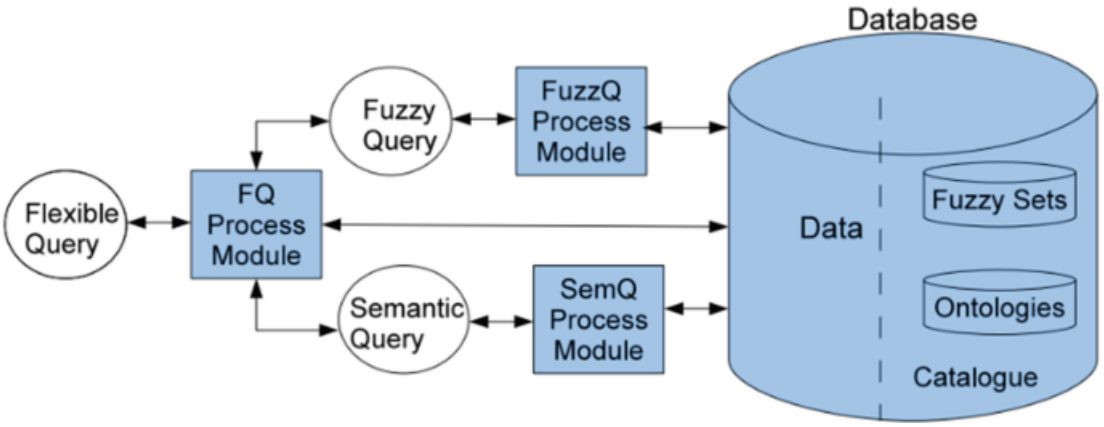


Рисунок 14.3 – Приклад побудови модуля формування нечітких запитів до БД

2. Нечіткі асоціативні правила, напрямок фокусується на виявленні у БД прихованих закономірностей у форматі лінгвістичних висловлювань різних рівнів деталізації. Для імплементації цього підходу застосовуються механізми нечітких транзакцій, оцінки ступеня достовірності асоціативних правил.

3. Методика побудови нечітких когнітивних карт, що застосовується для побудови моделей, що описують різні причинно-наслідкові взаємозв'язки між сутностями або концептами заданої предметної галузі. Перевагою даного підходу в порівнянні зі звичайними методами формування когнітивних карт є оперування гнучкішою структурою правил у вигляді нечіткого орієнтованого графа, в якому окремим НМ відповідають його вузли. Орієнтованість ребер графа формує та задає різні причинно-наслідкові зв'язки між

позначеними сутностями та концептами, виділяючи та враховуючи значення ваг їх зв'язків між собою.

Активне застосування цей підхід отримав як основу моделювання ієрархічних систем різного ступеня складності завдяки високому ступеню наочності уявлення зв'язків елементів системи та простоті інтерпретації специфіки взаємодії між ними. Складністю підходу є формалізований процес формування підсумкової когнітивної карти, а також необхідність проведення додаткових обчислювальних експериментів для оцінки адекватності моделі реальної системи. Описані проблеми можуть бути вирішені за допомогою використання та програмної імплементації алгоритми цільової побудови когнітивних карток в атематичному режимі на базі різнобічного аналізу вибірки вхідних даних.

4. Нечітка кластеризація, що допускає приналежність одного об'єкта до кількох окремих кластерів з різною мірою (вагою). Особливістю даного підходу є гнучкіша інтерпретація даних особливо для граничних об'єктів вибірки, що знаходяться рівновіддалено від декількох кластерів. Типовим прикладом реалізації цього підходу є алгоритм c-means та його модифікована версія у формі алгоритму Густафсона-Кесселя.

Лекція 15 «Прикладні сфери застосування нечіткої логіки»

Нечітка логіка застосовується у різних галузях і надає ефективні методи моделювання невизначеності та розмитості даних. Ось кілька прикладів практичного застосування нечіткої логіки:

1. Нечіткі системи управління (Fuzzy Control Systems).

Нечітка логіка широко використовується у системах управління, особливо у сферах, де складно виразити чіткі правила. Приклади включають кліматичні системи, автомобільні трансмісії, стабілізацію камер у фотоапаратах тощо.

2. Автоматизовані транспортні системи.

У транспортних системах нечітка логіка може використовуватися для оптимізації керування світлофорами, адаптації швидкості руху транспортних засобів, керування системами безпеки тощо.

3. Пилогазові системи контролю.

Нечітка логіка застосовується в системах контролю за пилогазовими викидами у промисловості. Вона допомагає адаптуватися до змін у навколишньому середовищі та ефективно керувати процесами очищення повітря.

4. Кліматичні системи.

Нечітка логіка використовується для управління кліматичними системами, де потрібна адаптація до умов, що змінюються, таким як температура, вологість і вітер.

Схема системи контролю клімату у теплиці наведена на рис.15.1.

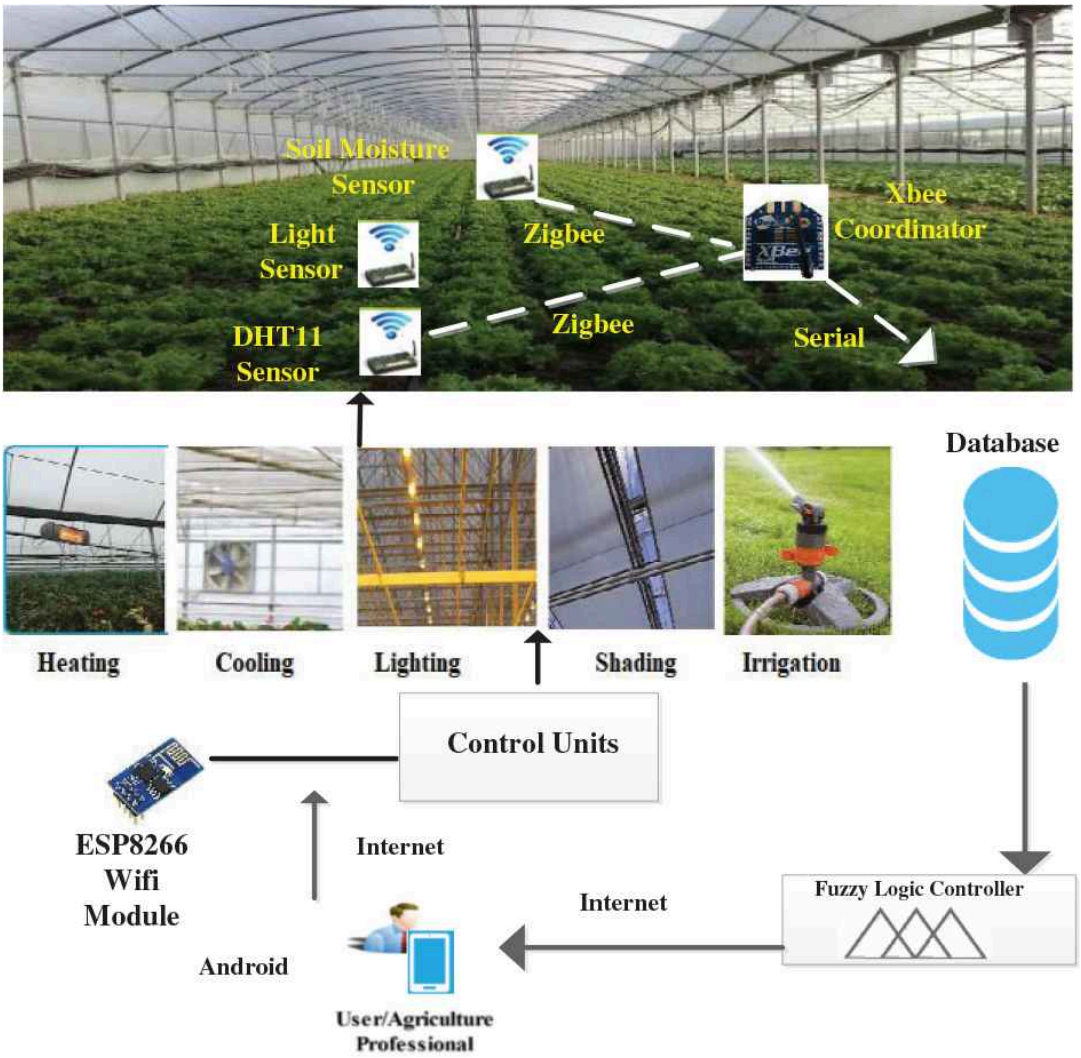


Рисунок 15.1 – Схема системи контролю клімату у теплиці

5. Побутові пристрої.

У побутових пристроях, таких як пральні машини, посудомийні машини та кондиціонери, нечітка логіка застосовується для автоматичного регулювання параметрів роботи залежно від умов та переваг користувача.

Пральна машина є чудовим прикладом розуміння того, як працює нечітка логіка в ШІ. Розглянемо базову систему нечіткого керування, яка регулює споживання води пральною машиною, час прання, швидкість віджимання та процес прання.

У цьому випадку вхідними параметрами є кількість одягу, ступінь забруднення та вид забруднення. Тоді як кількість одягу впливатиме на кількість випитої води, кількість бруду залежатиме від прозорості води, а вид бруду можна визначити за тим, як довго колір води залишається незмінним.

Система нечіткої логіки визначала б час прання одягу, враховуючи ці три основні параметри за допомогою логіки «якщо-тоді».

6. Фінансова аналітика.

У фінансовій аналітиці нечітка логіка може використовуватися для моделювання та аналізу фінансових даних, а також для прийняття рішень за умов невизначеності.

7. Медична діагностика.

У медичній області нечітка логіка може допомогти у створенні систем діагностики, де класифікація симптомів та захворювань може бути розмитою та нечіткою.

Схема системи медичної діагностики на базі використання моделей нечіткої логіки наведено на рис.15.2.

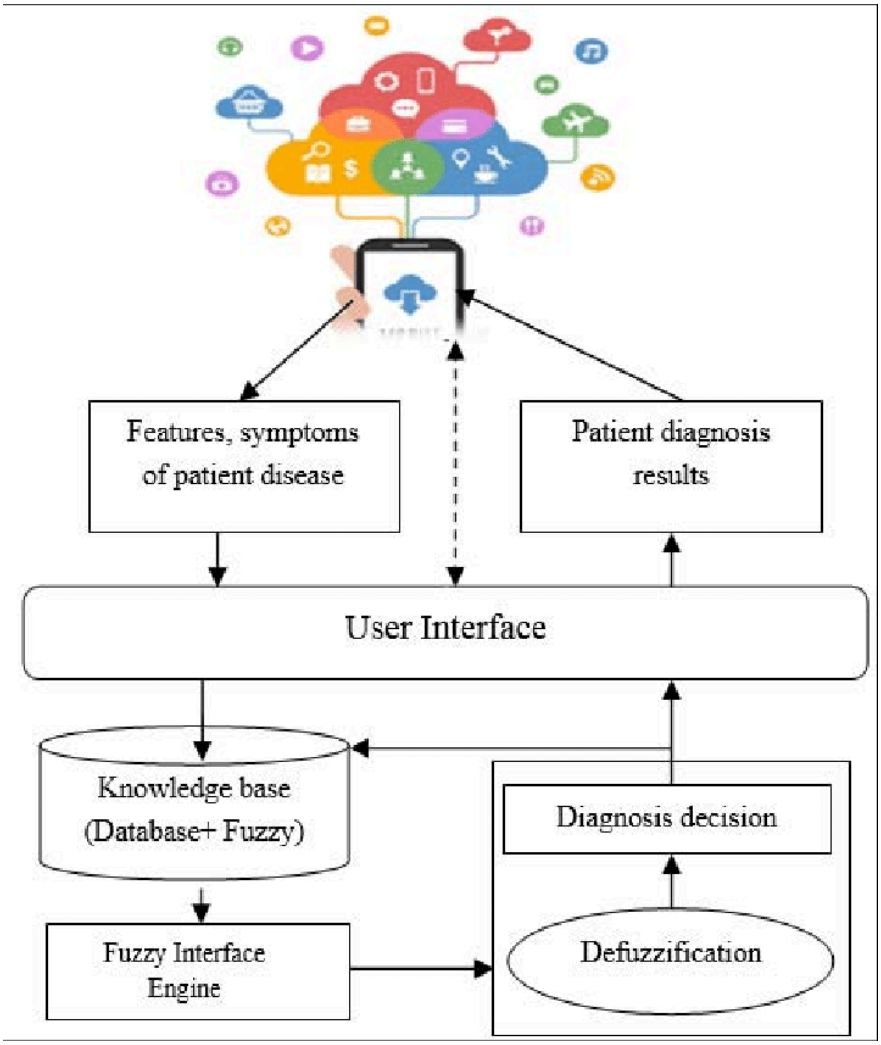


Рисунок 15.2 – Схема системи медичної діагностики на базі використання моделей нечіткої логіки

8. Автомобільна промисловість.

В автомобільній промисловості нечітка логіка застосовується для керування трансмісією, антиблокувальною системою гальм (ABS), системою керування стабілізацією (ESP) та іншими системами, де потрібна адаптація до змінних умов на дорозі.

Ці приклади демонструють широкий спектр областей, де нечітка логіка успішно застосовується для вирішення завдань, пов'язаних з невизначеністю та розмитістю даних.

Приклади застосування кейсів на базі використання нечіткої логіки у різних компаніях наведено у таблиці 15.1.

Таблиця 15.1 – Приклади застосування нечіткої логіки

Продукт	Компанія	Застосування нечіткої логіки
Антиблокувальна система гальм	Nissan	Для керування гальмами у небезпечних випадках, які залежать від швидкості автомобіля, прискорення, швидкості коліс та прискорення.
Авто та transmission	NOC/Nissan	Нечітка логіка використовується для управління впорскуванням палива та запаленням на основі положення дросельної заслінки, температури охолоджувальної води, обертів за хвилину і т.д.
Авто двигун	Honda, Nissan	Для вибору передачі в залежності від навантаження двигуна, стилю водіння та дорожніх умов.
Копіювальна машина	canon	Використовується для регулювання напруги барабана в залежності від густини зображення, вологості та температури.
Круїз контроль	Nissan, Isuzu, Mitsubishi.	Використовується для налаштування дросельної заслінки, щоб встановити швидкість та прискорення автомобіля.
Посудомийна машина	Matsushita	Використання для регулювання циклу очищення, стратегій полоскання та миття залежить від кількості посуду та кількості їжі, що подається на посуді.
Управління ліфтом	Fujitech, Mitsubishi Electric, Toshiba	Використовується для зменшення часу очікування залежно від пасажиропотоку.
Гольф діагностична система	Maruman Golf	Використовується для вибору гольф-клубу на основі переваг гравця в гольф та статури.
Фітнес-менеджмент	Omron Company	Нечіткі правила, які передбачаються для перевірки придатності своїх співробітників.
Управління піччю	Nippon Steel	Використовується для цементного змішування
Мікрохвильова піч	Mitsubishi Chemical	Встановлює потужність лунок та стратегію приготування.
Кишеньковий комп'ютер	Hitachi, Sharp, Sanyo, Toshiba	Розпізнає рукописні символи
Плазмове травлення	Mitsubishi Electric	Встановлює час та стратегію травлення

Література

1. laoui Mohamed. Fuzzy TOPSIS: Logic, Approaches, and Case Studiesю- CRC Press, 2021. — 217 p.
2. Beliakov Gleb, James Simon, Wu Jian-Zhang. Discrete Fuzzy Measures. - Springer, 2020. — 245 p.
3. Carter J., Chiclana F., Khuman A., Chen T. (Eds.). Fuzzy Logic: Recent Applications and Developments. - Springer, 2021. — 270 p.
4. Chaira T. Fuzzy Set and Its Extension: The Intuitionistic Fuzzy Set. - Hoboken: Wiley, 2019. — 296 p.
5. Farhadinia B. Hesitant Fuzzy Set: Theory and Extension. -New York: Springer, 2021. — 162 p.
6. https://www.guru99.com/ru/what-is-fuzzy-logic.html
7. https://www.quora.com/What-are-good-real-world-examples-of-fuzzy-logic-being-used
8. https://www.simplilearn.com/fuzzy-logic-in-ai-article
9. Alpay Ö. The Control of Greenhouses Based on Fuzzy Logic Using Wireless Sensor Networks / Ö. Alpay, E. Erdem // International Journal of Computational Intelligence Systems. – 2018. - №12. – PP. 190 – 203
10. https://jfuzzylogic.sourceforge.net/html/manual.html