Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра програмування та захисту інформації

Експертні системи

Методичні вказівки
до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп’ютерна інженерія",
122 "Комп’ютерні науки та інформаційні технології",
125 "Кібербезпека"

Кіровоград 2016

Міністерство освіти і науки України
Кіровоградський національний технічний університет
Кафедра програмування та захисту інформації

Хох В.Д., Мелешко Є.В., Дрєєв О.М.

Експертні системи

Методичні вказівки
до виконання лабораторних робіт студентами
денної та заочної форми навчання спеціальностей
123 "Комп’ютерна інженерія",
122 "Комп’ютерні науки та інформаційні технології",
125 "Кібербезпека"

Затверджено
на засіданні кафедри
програмування та
захисту інформації
Протокол №13
від 15.02.16

Кіровоград 2016

Експертні системи. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 123 "Комп'ютерна інженерія", 122 "Комп'ютерні науки та інформаційні технології", 125 "Кібербезпека" / Укл.: В.Д. Хох, Є.В. Мелешко, О.М. Дрєєв – Кіровоград: КНТУ, 2016. – 35 с.

Дані методичні вказівки адресовані майбутнім розробникам програмного забезпечення, фахівцям з захисту інформації та з аналізу даних, можуть використовуватися також студентами інших спеціальностей. Вони включають в себе основи проектування експертних систем та баз знань, містять основні теоретичні положення та практичні завдання, необхідні для засвоєння навчального матеріалу.

Укладачі: Хох В.Д.,
Мелешко Є.В., канд. техн. наук, доцент
Дрєєв О.М., канд. техн. наук

Рецензенти: Смірнов О.А., доктор техн. наук, професор
Якименко Н.М., канд. фіз.-мат. наук, доцент

ЗМІСТ

Лабораторна робота №1

Тема: Продукційне представлення знань

Мета: Ознайомитися з загальними принципами побудови експертних систем на основі продукційного представлення знань

Теоретичні відомості

Продукційна модель представлення знань, як видно з назви, базується на використанні продукцій. У продукцій немає єдиної, окремої теорії як і немає чіткого математичного апарату, тому реалізації експертних систем на основі продукцій можуть значно відрізнятися. В загальному вигляді продукція це – кортеж <i, C, R, F>, де:

i – унікальний ідентифікатор продукції (наприклад порядковий номер у базі знань);

C – ядро продукції. Ядро продукції найчастіше має вигляд імплікації – $f_1 \wedge f_2 \wedge f_3 \Rightarrow F$;

R – правило використання продукції. Класично, має вигляд кон'юнкції – $f_1 \wedge f_2 \wedge f_3$. В тому випадку, коли кон'юнкція набирає значення true, продукцію можна буде використовувати в прийнятті рішень;

F – набір функцій, які повинні бути використані, або набір даних для логічного виводу, або набір даних для фактологічної бази.

Розглянемо приклад побудови експертної системи, яка повинна керувати налаштуваннями температури в душі, керуючи вентилями крана. Результатом роботи системи повинна бути тепла вода, температура якої визначена завдяки експертній оцінці.

Спочатку визначимо факти, якими буде оперувати наша система:

f_1 – чи відкритий вентиль гарячої вводи;

f_2 – чи відкритий вентиль холодної вводи;

f_3 – чи повністю відкритий вентиль гарячої води;

f_4 – чи повністю відкритий вентиль холодної води;

f_5 – чи вода гаряча;

f_6 – чи вода холодна;

f_7 – чи вода тепла;

f_8 – крок відкриття вентилю.

На етапі визначення фактів, а отже і формування фактологічної бази системи, варто, для початку, визначити об’єкти, з якими буде працювати система, потім визначити їх властивості, а з них обрати ті, які необхідні для функціонування системи. Не варто відкидати властивості об’єктів, які було визначено, але не обрано для фактологічної бази. Продукційні системи досить легко масштабувати на етапі проектування, тому варто залишити всі дані про потенційні об’єкти фактологічної бази.

Почнемо формувати базу правил нашої продукційної системи:

<0, «$f_1 \wedge f_5$», «$\neg f_4$, $\neg f_7$», «ВідкритиВентильХолодноїВодиНа(f_8)»>

<1, «$f_2 \wedge f_6$», «$\neg f_3$, $\neg f_7$», «ВідкритиВентильГарячоїВодиНа(f_8)»>

<2, «$f_1 \wedge f_2 \wedge f_5$», «$f_3$, $\neg f_7$», «ЗакритиВентильГарячоїВоди()»>

<3, «$f_1 \wedge f_2 \wedge f_6$», «$f_4$, $\neg f_7$», «ЗакритиВентильХолодноїВоди()»>

На даному етапі чотирьох продукцій повинно вистачити. Промовимо продукцію: «Продукція нуль, **ЯКЩО** (f_1)Відкритий вентиль гарячої води **І** (f_5)Вода гаряча **ТОДІ** (блок F) Викликати функцію з параметром f_8 – *ВідкритиВентильХолодноїВодиНа*(f_8) **ЛИШЕ ТОДІ І ТІЛЬКИ ТОДІ, ЯКЩО** (блок R) (f_3)Вентиль гарячої води **НЕ** відкритий повністю і (f_7)Вода **НЕ** тепла».

Перейдемо до алгоритму та архітектури системи, яка повинна обробляти продукції. Загальну архітектурну схему системи зображено на рисунку 1.1, а алгоритм вирішувача на рисунку 1.2.

Архітектурно система складається з наступних елементів:

– "Система логічного вводу" – елемент, задача якого переробити вхідні дані у зрозумілий для системи формат. Однією з можливих функцій є розподіл даних. Наприклад, якщо деякі дані є параметрами фактологічної бази, а деякі параметрами, які конфігурують роботу вирішувача, – система логічного вводу застосує їх відповідно до їх ролі у системі.

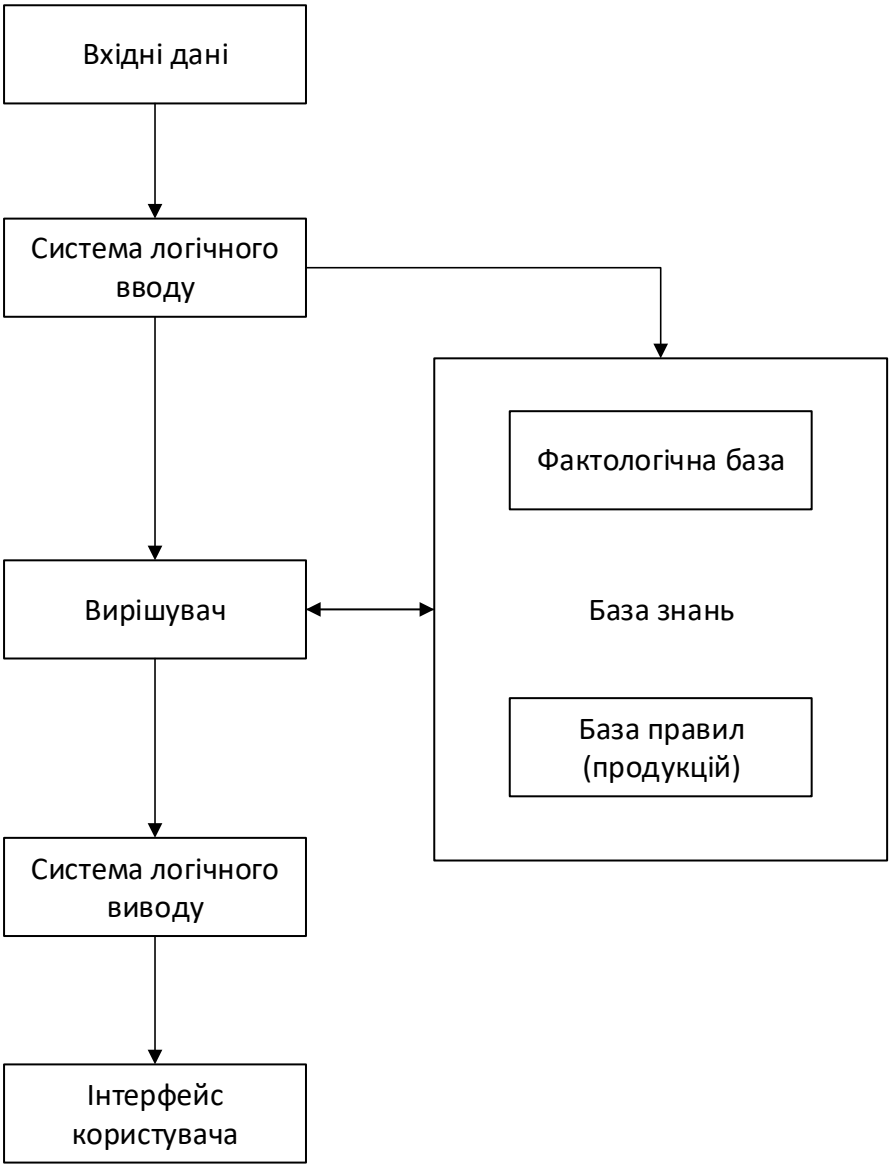

Рисунок 1.1 – Загальна схема архітектури експертної системи на основі продукційного представлення знань

– "База знань" – базу знань можливо розділити на дві частини: фактологічна база та база правил. Цей елемент характерний для експертних систем загалом; в залежності від виду представлення знань його вигляд буде відрізнятись. Для експертних систем з продукційним представленням знань характерне розділення бази на фактологічну (також носить назву - робоча пам’ять) і на базу продукцій.

– "Фактологічна база" – або робоча пам’ять є відображенням того, що знаходиться у полі зору експертної системи, які факти вона використовує в своїх розрахунках. У цій же пам’яті відбуваються зміни значень фактів.

– "База правил (продукцій)" – зазвичай, база даних, в якій певним, в залежності від типу та особливостей експертної системи, чином зберігаються продукції.

– "Система логічного виводу" – система, або підсистема, в задачі якої входить підготовка результатів обчислення експертної системи для зручного користувачу формату відображення даних.

– "Вирішувач" – основний елемент експертної системи. Задача вирішувача обробити дані з робочої пам’яті (фактологічної бази) і застосувати їх відповідно правилам у базі продукцій (правил), тобто, фактично задача вирішувача – конвертувати дані в знання.

Зупинимось на "вирішувачі". Алгоритм роботи вирішувача з реалізованим у ньому «наївним» алгоритмом обробки продукцій зображено на рисунку 1.2.

Якщо продукційна система складається з невеликої кількості продукцій і фактів – не варто витрачати час на алгоритми оптимізації, можна одразу перейти до побудови «наївного» алгоритму обробки продукцій. Суть «наївного» алгоритму полягає у послідовному переборі продукцій і виконанні тих, які можна використати згідно результатів обчислення блоку R. Після того як продукція виконана, алгоритм переходить до початку списку продукцій.

Спробуємо відтворити алгоритм словесно, крок за кроком. Але спочатку варто зупинитися на тому, що для роботи даної експертної системи необхідні деякі зовнішні модулі, які визначать момент, коли вода стане теплою, і змінять відповідний факт у робочій пам’яті системи, вони ж повинні повідомити системі, яка зараз вода, холодна або гаряча. Взаємодія з експертною системою відбувається завдяки модифікації її робочої пам’яті.

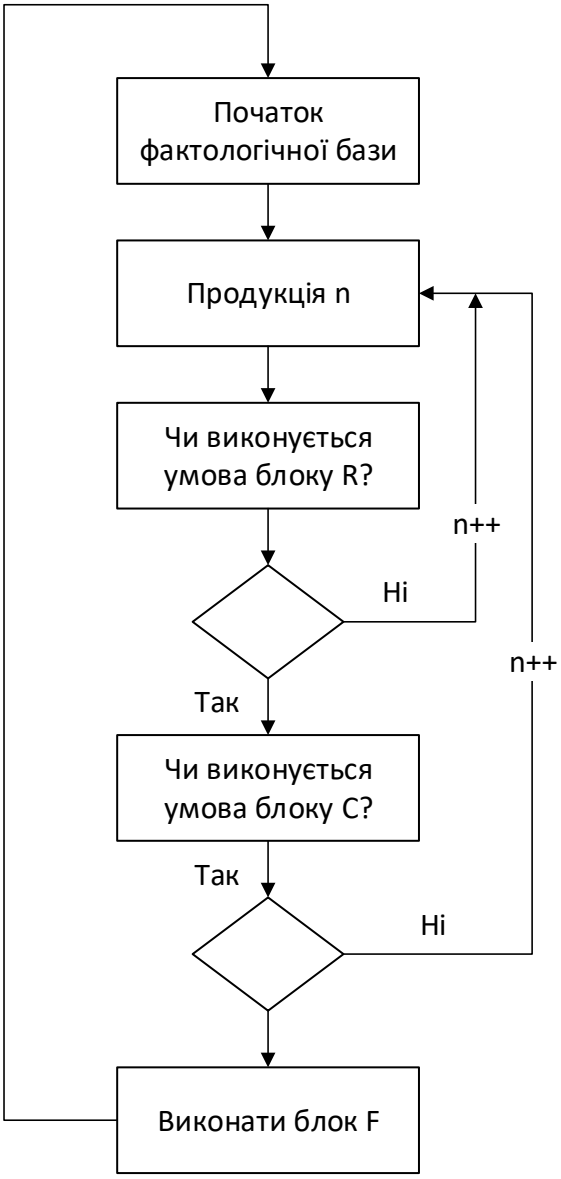

Рисунок 1.2 – Загальна схема алгоритму вирішувача

Отже, встановимо робочу пам’ять в певний стан:

f_1 = true;

f_2 = true;

f_3 = false;

f_4 = false;

f_5 = false;

f_6 = true;

f_7 = false;

f_8 = 1.

Алгоритм формує список продукцій і переходить до першої, підставляючи замість назв фактів їх значення з робочої пам’яті (фактологічної бази). Спочатку це застосовується до блоку R. Отже:

1. Продукція «0».
2. Блок R = «$\neg f_3$, $\neg f_7$» = «**NOT** false **AND NOT** false» = true.
3. Переходимо до ядра продукції(оскільки R = true) і намагаємося його активувати C = «$f_1 \wedge f_5$» = «true **AND** false» = false. Ядро продукції не буде активовано.
4. Переходимо до наступної продукції.
5. Продукція «1».
6. Блок R = «$\neg f_4$, $\neg f_7$» = «**NOT** false **AND NOT** false» = true
7. C = «$f_2 \wedge f_6$» = «true **AND** true» = true
8. Оскільки C = true – переходимо до блоку F, викликаємо функцію «*ВідкритиВентильХолодноїВодиНа*(f_8)» з параметром фактологічної бази f_8 Нехай, в цей момент вентиль став повністю відкритий, а вода залишилася холодною, в цей момент буде змінено факт f_4 = true.
9. Оскільки ядро продукції було активовано – алгоритм переходить на початок списку продукцій.
10. Продукція «0».
11. R = R = «$\neg f_3$, $\neg f_7$» = «**NOT** false **AND NOT** false» = true.
12. C = C = «$f_1 \wedge f_5$» = «true **AND** false» = false.
13. Продукція «1»
14. R = «$\neg f_4$, $\neg f_7$» = «**NOT** true **AND NOT** false» = false.
15. Продукція «2»
16. R = «f_3, $\neg f_7$» = «false **AND NOT** false» = false.
17. Продукція «3»
18. R = «f_4, $\neg f_7$» = «true **AND NOT** false» = true.
19. C = «$f_1 \wedge f_2 \wedge f_6$» = «true **AND** true **AND** true» = true.
20. F = *ЗакритиВентильХолодноїВоди*() – В цей момент функція закриває вентиль холодної води - f_2 = false, а оскільки єдиним вентилем

відкритим залишається вентиль гарячої води, вода стає не холодною а гарячою f_5 = true, f_6 = false. Ядро продукції було активовано – алгоритм виконує перехід в початок списку продукції.

21. Продукція «0»
22. R = «$\neg f_3$, $\neg f_7$» = «**NOT** false **AND** **NOT** false» = true.
23. C = «$f_1 \wedge f_6$» = «true **AND** true»
24. F = *ВідкритиВентильХолодноїВодиНа*(f_8)
25. Перехід до продукції «0»

Як можна побачити, тепер система перейде в цикл, до того моменту поки f_4 = true, але з цього можна буде зробити висновок що холодної води взагалі немає, або система буде відкривати вентиль холодної вводи на f_8 доти, доки f_7 = true, у цьому випадку у всіх продукцій блок R перестане виконуватись, і система буде пробігати по списку нічого не змінюючи оскільки досягнуто цільового стану.

Завдання: Розробити експертну систему на основі продукційного представлення знань. Остання цифра залікової книжки – кількість продукцій (якщо менше 5, тоді 5). Передостання цифра залікової книжки – кількість фактів (якщо менше 5, тоді 10).

Контрольні питання:

1. Опишіть загальну схему архітектури ЕС на основі продукційного представлення знань.
2. Що таке продукція? З яких елементів вона складається?
3. Що представляє собою ядро продукції?
4. Що таке база знань?
5. Що таке фактологічна база?
6. Як працює вирішувач в продукційній ЕС? Якою є загальна схема алгоритму його роботи?

Лабораторна робота №2

Тема: Експертні системи з нечіткою логікою

Мета: Ознайомитися з загальними принципами нечіткої логіки та програмно реалізувати

Теоретичні відомості

Нечітка логіка дозволяє зручно та відносно швидко вирішити питання, які не здатна вирішити традиційна булівська\формальна логіка. Наприклад, у разі виникнення конфлікту, коли декілька логічних виразів стають істинними і неможливо визначити який з них більш релевантний. Вирішення подібного досягається за допомогою використання у процесі вирішення функцій фазифікації та дефазифікації, реалізація яких, в свою чергу, можлива за допомогою використання нечітких змінних.

Розглянемо детально нечітку змінну. Нечітка змінна – це кортеж, що має наступний вигляд <N, L, A>, де:

N – ім’я змінної;

L – область визначення змінної;

A – нечітка множина на L.

В даній лабораторній роботі у якості нечіткої множини на області визначення змінної буде розглянуто деяку лінгвістичну змінну. Розглянемо її.

Лінгвістична змінна є кортеж виду <N, T, L, F, B> де:

N – ім’я змінної;

T – множина термів(наприклад: {«Холодно», «Прохолодно», «Тепло», «Спека»});

L – область визначення;

F – функція генерування нових термів за допомогою логічних зв’язок та модифікаторів. Наприклад «Дуже тепло» або «Досить прохолодно»;

F – функція встановлення лінгвістичної змінної з термів множини T.

Розглянемо приклад побудови нечіткої системи, яка повинна оцінити придатність їжі до вживання певним організмом.

Нехай, у нас є три нечіткі змінні: «Придатність їжі», «Смакові якості», «Корисність».

Корисність буде визначати наскільки безпечно вживати дану їжу. Нехай змінна буде визначатися у діапазоні від 0 до 100: якщо x<50, то їжа більше шкідлива ніж корисна. Змінна буде мати вигляд: <«Корисність», [0, 100], A> де A = {«Небезпечно(0, 25)», «Небажано(26, 50)», «Мало корисно(51, 75)», «Корисно(76, 100)»}.

Смакові якості будуть визначатися за таким самим принципом, отже: <«Смакові якості», [0, 100], B> де B = {«Гидка(0, 25)», «Несмачно(26, 50)», «Досить смачно(51, 75)», «Дуже смачно(76, 100)»}.

Придатність їжі буде визначатись за формулою (Корисність + Смакові якості) / 2, і, в свою чергу, буде визначати можливість використання їжі. Принцип визначення змінної залишається: <«Придатність їжі», [0, 100], C> де C = {«Непридатна(0, 25)», «Придатна за певних обставин(26, 50)», «Досить придатна(51, 75)», «Придатна(76, 100)»}.

Додатково визначимо змінну, яка б характеризувала ступінь ситості організму. Нехай: <«Рівень ситості», [0, 100], D>, де D = {«Небезпечно для життя(0, 25)», «Голодний(26, 50)», «Зголоднів(51, 75)», «Ситий(76, 100)»}.

Сформулюємо правило харчування організму. Для цього використаємо нечітку множину, в яку помістимо правила, які б ми могли використати у алгоритмі. Вигляд та механізм в даному описі надано лише для прикладу і не обов’язковий для реалізації у такому вигляді. Сформуємо множину R = {«Рівень ситості[0,25] && Придатність їжі[0, 100]», «Рівень ситості[26,50] && Придатність їжі[26, 100]», «Рівень ситості[51,75] && Придатність їжі[51, 100]», «Рівень ситості[76,100] && Придатність їжі[76, 100]»}.

Перейдемо до алгоритму, який буде працювати з цими даними. Його зображено на рисунку 2.1:

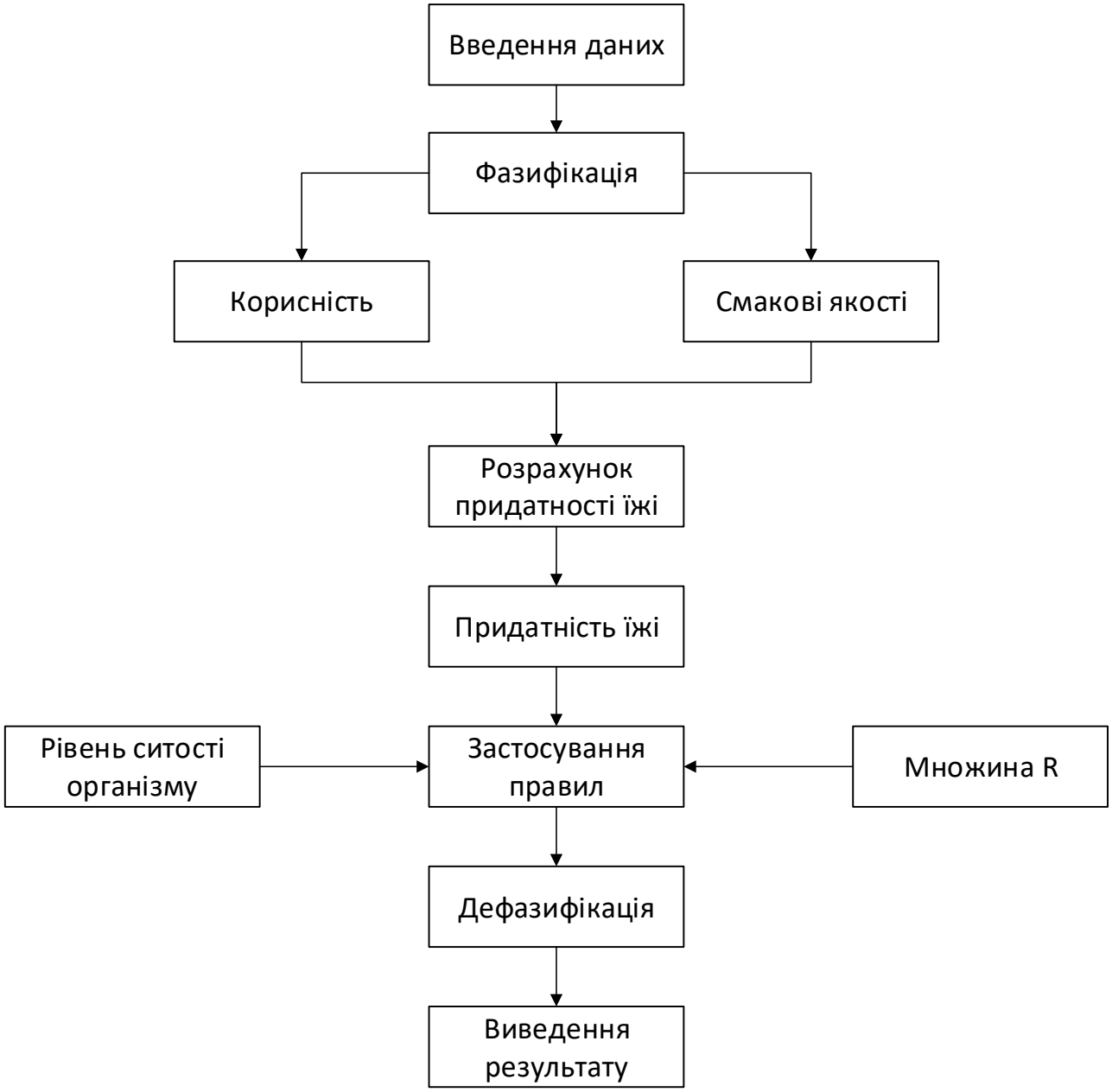

Рисунок 2.1 – Схема алгоритму пошуку рішення

Розглянемо кроки алгоритму:

– Введення даних – користувач, або модуль програми «готує їжу», надаючи їй певних характеристик, у нашому прикладі «корисність» та «смакові якості».

– Фазифікація – даний етап характерний саме для алгоритмів, що використовують нечітку логіку при прийнятті рішення. В різних алгоритмах та в залежності від потреб – етап фазифікації може значно відрізнятися у своїй реалізації. Але суть даного процесу найчастіше спільна і полягає у

конвертуванні вхідних даних до вигляду нечітких змінних. Варто розуміти, що різні алгоритми можуть дуже по-різному визначати вигляд нечітких даних, до того ж на вигляд етапу фазифікації впливає і вигляд вхідних даних.

– Далі, на основі отриманих вхідних даних ми розраховуємо придатність їжі до вживання.

– Після розрахунку придатності ми можемо почати застосовувати правила. На цьому ж етапі відбуваються спроби активувати правило.

– Дефазифікація – процес, обернений до фазифікації, і також як фазифікація є залежним від специфіки алгоритму. В даному алгоритмі ціллю дефазифікації є перетворення параметрів «Ступінь ситості» та «Придатність їжі» до булівських (true або false) значень в правилах виконання їх умови (як можна побачити з вигляду правил, після обробки умова буде виглядати як - «bool && bool»). Таким чином, результатом роботи дефазифікатора в даному алгоритмі повинен стати, спочатку, вираз – if (bool and bool), а потім дефазифікатор повинен обчислити значення цього виразу.

– Вивід даних: якщо остаточним результатом дефазифікатора є true, виводимо, що їжу було з’їдено, у випадку false – ні.

Завдання:

Розробити алгоритм годування організму за прикладом наведеного та з використанням принципів нечіткої логіки відповідно до параметрів свого *варіанту.

*передостання цифра залікової книжки – кількість параметрів, що характеризують їжу, остання цифра залікової книжки – кількість параметрів, що характеризують стан організму.

Контрольні питання:

1. Що таке нечітка логіка? Які задачі вона дозволяє вирішувати?
2. Що собою представляє нечітка змінна?
3. Що собою представляє лінгвістична змінна?
4. Яким чином відбувається пошук рішення в ЕС з нечіткою логікою?
5. Що таке фазифікація? Що таке дефазифікація?

Лабораторна робота №3

Тема: Семантична модель представлення знань

Мета: Ознайомитися з загальними принципами побудови експертних систем на основі семантичної моделі представлення знань

Теоретичні відомості

Семантичну мережу найлегше зобразити у вигляді графу, вершини якого є певними сутностями, а ребра між ними зображують відношення між ними. Для прикладу побудуємо семантичну мережу для робота, якому необхідно пройти лабіринт і прийняти деякі рішення відносно об'єктів, що йому зустрінуться.

Для прикладу, припустимо, що робот переміщується в деякому дискретному просторі. Об'єкт займає чітко визначену позицію і займає чітко визначений простір, який дорівнює 1. Переміститися робот може теж на 1 за один раз. Робот має пам'ять, яка може запам'ятати лише один крок, тобто робот пам'ятає, звідки він прийшов, але лише попередній крок.

Опишімо сутності, з якими доведеться працювати роботу:

– Стінка.
– Попередня клітинка (попередній крок робота; клітинка, з якої він прийшов).
– Порожня клітинка (клітинка, в яку робот може перейти).
– Ваза.
– Енергетичний блок (роботу потрібна підзарядка).
– Собака (являє загрозу для робота).

Тепер опишімо стани, в які робот може переходити:

– Заряджений (батареї робота повністю заряджені).
– Нормальний (батареї робота трохи розряджені).
– Розряджений (батареї робота розряджені – рух неможливий).

Тепер необхідно продумати, як відноситься робот в кожному зі станів до кожного з об'єктів. Для цього побудуємо граф, який і буде нашою семантичною

мережею. Граф зображено на рисунку 3.1, а на рисунку 3.2 зображено схему лабіринту для робота. Кольори клітинок у схемі лабіринту на рисунку 3.2 означають наступне:

- червона клітинка – собака;
- фіолетова клітинка – ваза;
- жовта клітинка – попередня клітинка;
- чорна клітинка – стіна;
- біла клітинка – порожня клітинка;
- зелена клітинка – робот;
- блакитна клітинка – енергетичний блок.

Спробуємо відтворити поведінку робота в ситуації, зображеній на рисунку 3.2. Нехай зір робота дозволяє бачити одночасно 4 клітинки: так у його полі зору одна клітинка – стіна, одна – собака, одна попередня клітинка і одна порожня. В залежності від ПЗ, яке надсилає семантичній мережі дані, у семантичну мережу може в першу чергу прийти інформація про порожню клітинку і робот проігнорує собаку і перейде в порожню клітинку, а може прийти інформація про собаку, в такому випадку робот зробить крок у попередню клітинку (див. рисунок 3.1) і почне рух у зворотному напрямку, оскільки з двох сторін буде оточений стінами, а позаду руху буде попередня клітинка, в яку він переходить, тільки якщо бачить собаку чи вазу, а попереду його руху будуть порожні клітини. Робот переходить від стану до стану в залежності від зовнішнього ПЗ, яке керує ним; до складу семантичної мережі ці переходи не входять, але описані. Енергетичний блок, зображений блакитним, робот проігнорує, якщо перебуває у зарядженому стані, і використає, якщо знаходиться у розрядженому або нормальному стані.

Для пошуку зручних методів запису та зберігання семантичної мережі у своїй програмі варто звернутися до теорії графів, а саме до методів подання графів, і, в залежності від побудованої семантичної мережі, обрати найзручніший спосіб.

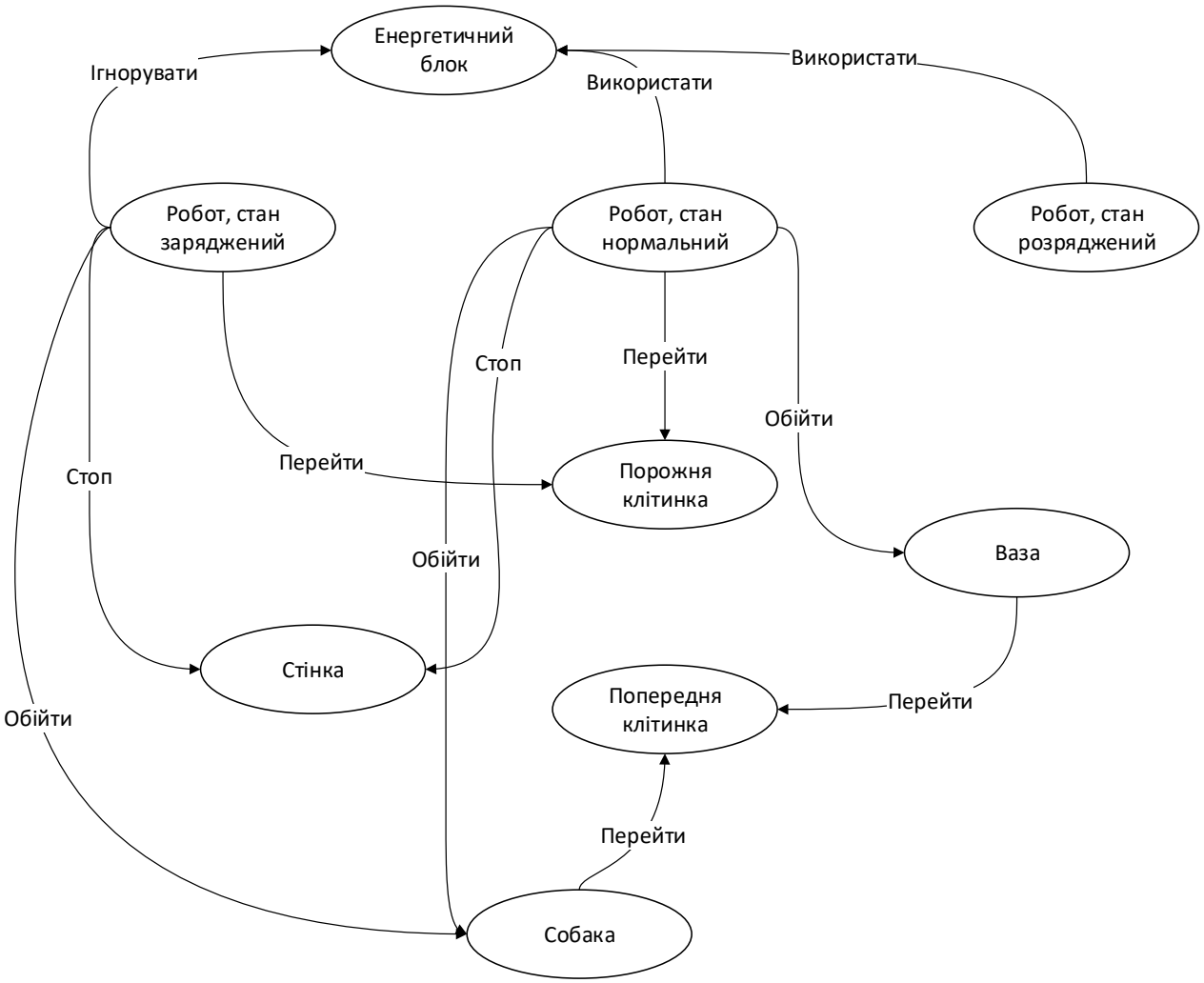

Рисунок 3.1 – Граф семантичної мережі

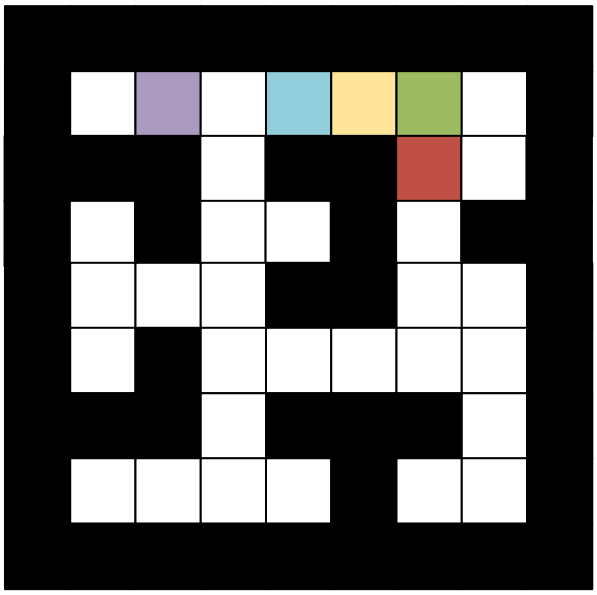

Рисунок 3.2 – Вигляд лабіринту для робота

Завдання: Розробити експертну систему на основі семантичної мережі. Остання цифра залікової книжки – кількість сутностей (якщо менше 5, то 5), передостання + остання цифра залікової книжки – кількість ребер (якщо менше 10, то 10).

Контрольні питання:

1. Що таке семантична мережа?
2. З яких елементів складається семантична мережа?
3. Які задачі можна вирішувати за допомогою семантичних мереж?

Лабораторна робота №4

Тема: Фреймова модель представлення знань

Мета: Ознайомитися з загальними принципами побудови експертних систем на основі фреймової моделі представлення знань

Теоретичні відомості

Фреймова модель представлення знань виникла в процесі вирішення проблеми значного ускладнення продукційних систем та недостатньої інформативності семантичних мереж. Фреймова модель дозволяє використовувати продукції, а за архітектурою схожа на семантичні мережі.

Фрейм можна записати як кортеж <I, N, S> - такий вигляд буде мати дуже простий фрейм.

Розглянемо детальніше:

I – унікальний ідентифікатор, може бути числом – порядковим номером.

N – ім’я фрейму, може бути списком певних імен, на зразок тегів для пошуку. Ім’я фрейму, найчастіше, ідентифікує його семантичну належність до чогось і не обов’язково має бути унікальним.

S – це множина слотів або полів фрейму. Кожне поле чітко та однозначно ідентифікує певну властивість сутності, яку описує фрейм. В полях можуть додатково або самостійно розміщатись прикріплені функції, поле також може містити посилання на інший фрейм або групу фреймів. Важливо зауважити, що в полі фрему може бути реалізована і продукція. Логічним є додавання до слоту певних властивостей – наприклад, значення за замовчуванням або мінімальне та максимальні значення поля, або інформацію чи є поле обов’язковим для заповнення.

Побудуємо просту фреймову модель, яка буде вирішувати, так звану, «проблему пінгвінів». Проблема полягає у тому, що найхарактернішою особливістю усіх птахів є можливість літати, на що пінгвіни, як відомо, нездатні (існують й деякі інші птахи, що нездатні літати, але для прикладу опису проблеми, чомусь, обрали пінгвінів), і коли, наприклад, намагатися

навчити експертну систему на основі семантичних мереж класифікувати тварин до певних видів, то поява пінгвіна серед цих тварин вимагає додаткових вузлів у мережі, які дозволять уникнути виключних ситуацій. В продукційних системах такої ситуації може взагалі не виникнути, оскільки в такій системі знання легко масштабувати і сутність може сприйматись не сталою кількістю властивостей, а може взагалі не сприйматися системою як окрема, самостійна сутність, а бути лише набором фактів, зв'язаних лише тим, що вони опинились в полі зору експертної системи під час прийняття рішень.

Тепер, коли проблему окреслено, почнемо будувати перші фрейми, які будуть наслідувати усі фрейми, які визначено певним типом тварин:

– <0, «Ссавці», S{«Має діафрагму[за замовчуванням true]», «Кількість шийних хребців[за замовчуванням 7]», «Наявність м'ясистих губ[за замовчуванням true]»}>;

– <1, «Плазуни», S{«Кількість відділів хребту[за замовчуванням 5]», «Холоднокровний [за замовчуванням true]»}>;

– <2, «Птахи», S{«Вміють літати[за замовчуванням true]», «Наявність пір'я[за замовчуванням true]», «Наявність дзьоба[за замовчуванням true]»}>.

Тепер, коли у нас є базові фрейми, ми можемо їх наслідувати і, в залежності від необхідності, додавати свої або змінювати значення існуючих слотів в базових фреймах. Зобразимо фреймову модель на рисунку 4.1.

З схеми стає зрозумілим, що для фреймової моделі характерна певна ієрархія, але це далеко не завжди так – в залежності від реалізації та особливостей вимог ієрархія може бути відсутньою. Більш того, наявність ієрархії скоріше вийняток, ніж правило.

Розглянувши схему уважніше, помітимо в деяких слотах слово «is», а за ним назву певного фрейма. Слово «is» було обрано навмання лише для того, щоб відмітити той факт, що далі буде йти назва фрейму, з якого поточний фрейм буде наслідувати властивості. Фрейм 7 наслідує 3, але перевантажує одну з його властивостей, визначаючи її як «Небезпечний [false]». Є певна схожість з об'єктно-орієнтовними методами програмування.

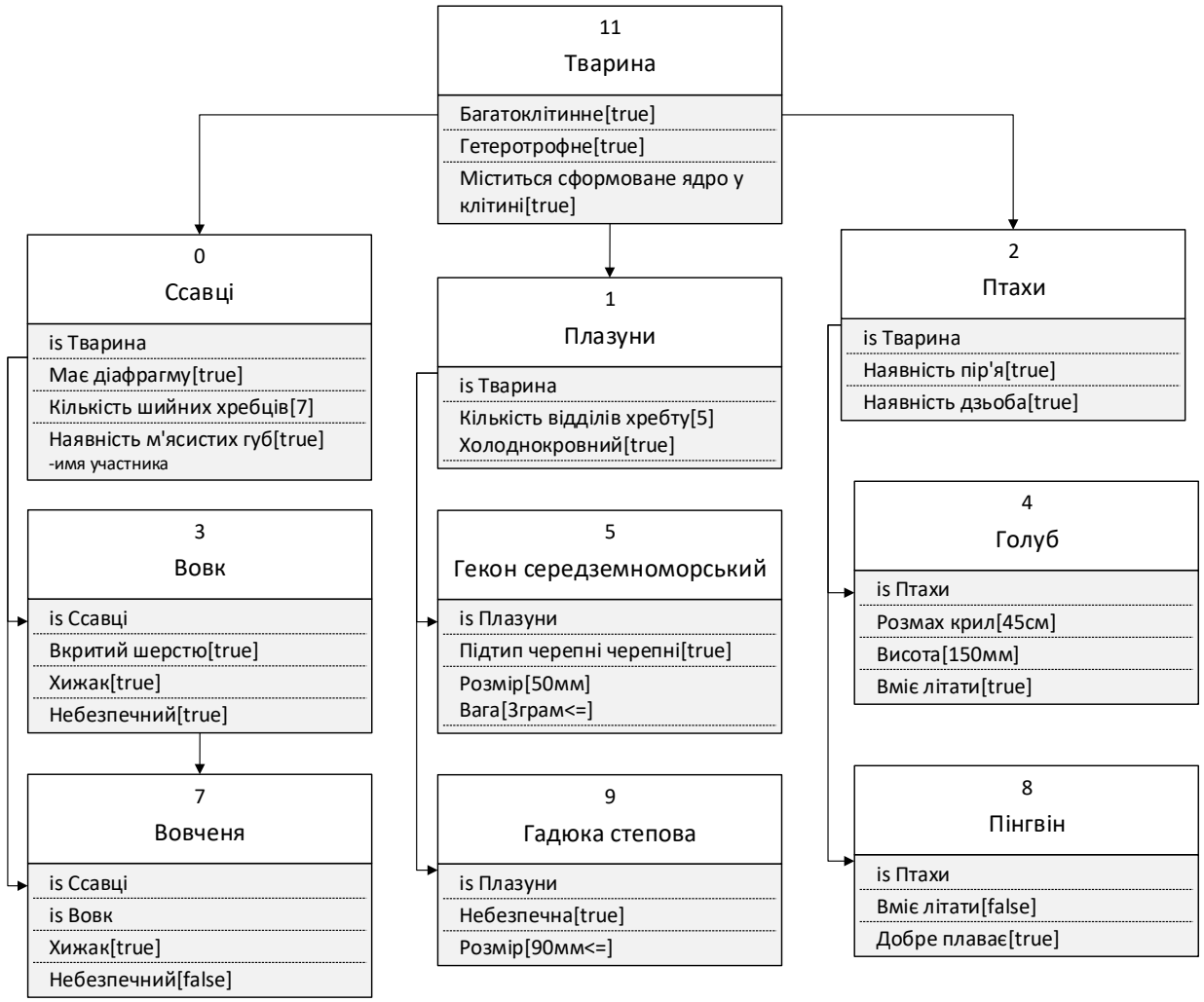

Рисунок 4.1 – Схема фреймової моделі

Що ж стосується «Проблеми Пінгвінів», то завдяки фреймам її можливо вирішити «в лоб», тобто, просто прибрати з одного з базових фреймів властивість, яка викликає колізії і, якщо вона все ж важлива, визначити її у всіх критичних фреймах, або зробити ще один-два додаткових базових фреймів. Можливий й інший метод вирішення, наприклад, перевантаження як з фреймами 3 та 7, але це повинно бути погоджене з алгоритмом обробки бази знань. До того ж у слот фрейма може бути додано певний функціонал, наприклад, назва функції, яку необхідно виконати у разі, якщо властивості фрейма відповідають поточним вхідним даним.

Універсального алгоритму, як наприклад «наївний», для продукцій у фреймовій моделі немає. Але наведемо приклад над розробленою у даній лабораторній роботі базою знань (рисунок 4.1).

Нехай, експертна система намагатиметься вгадати, якого звіра загадав користувач, задаючи прості питання про нього. В такому випадку найпростішим буде перейти до найвищого у ієрархії фрейму, надати можливість користувачеві ввести параметри з фрейму, підтвердити факт, що це тварина, після чого надавати можливість користувачеві вводити параметри випадково з усіх наступних фреймів, які наслідують фрейм, параметри якого збігаються з введеними даними користувача, і так далі. На прикладі з розробленою базою знань діалог користувача та системи може виглядати наступним чином (С: - система, К: - користувач):

– С: Це – багатоклітинне?(Так\Ні)
– К: Так.
– С: Гетеротрофне?(Так\Ні)
– К: Так.
– С: У клітині міститься сформоване ядро?(Так\Ні)
– К: Так.
– (Фрейм 11 можна вважати виконаним, оскільки дані фрейма збігаються з відповідями користувача, після цього система повинна знайти всі фрейми, які наслідують поточному, такі дані можуть бути записані у відповідному слоті фрейма, який унаслідується, це залежить від архітектури. В нашому випадку система знайде 3 фрейми, які явно наслідують фрейм 11 – 0, 1, 2)
– С: Має діафрагму?(Так\Ні)
– К: Ні.
– (Перехід до наступного фрейму)
– С: Холоднокровний?(Так\Ні)
– К: Ні.
– (Перехід до наступного фрейму)
– С: Має пір’я?(Так\Ні)
– К: Так.
– (Система може не знати про кількість фреймів у поточній вибірці, тому продовжує опитування з поточним фреймом)

– С: Має дзьоб?(Так\Ні)

– К: Так.

– (Фрейм підтверджено, виконується наступна вибірка – це фрейми 4 та 5, оскільки лише вони явно наслідують фрейм 2(Птахи))

– С: Вміє літати?(Так\Ні)

– К: Ні.

– (Оскільки ми йдемо за найпростішим шляхом – система бачить лише поточний фрейм, не запам'ятовує відповіді користувача, то, отримавши відповідь користувача, яка не збігається з поточним фреймом, вона перейде до наступного. Таким чином, якщо поточним фреймом був 4, система перейде до 8 і задасть таке ж питання)

– С: Вміє літати?(Так\Ні)

– К: Ні.

– С: Добре плаває?(Так\Ні)

– К: Так.

– С: Це «Пінгвін»

Ефективність системи, що використовує фреймову модель представлення знань, прямо залежить від вправності розробника бази знань – когнітивного інженера. В одному з слотів фрейма можуть бути вбудовані певні дані, які будуть виведені або використані у разі співпадіння поточних даних з описаними у слотах фрейма.

Завдання: розробити експертну систему на основі фреймової моделі представлення знань. Якщо остання цифра залікової книжки менше 5, то система не повинна мати ієрархії, якщо більше – ієрархія повинна бути. Сума останніх 2 цифр залікової книжки – кількість фреймів у системі (якщо менше 10, то 10).

Контрольні питання:

1. Що собою представляє фреймова модель представлення знань?
2. З яких елементів складається фрейм у фреймовій ЕС? Що таке слот (або поле фрейму)?

Лабораторна робота №5

Тема: Мультиагентні експертні системи

Мета: Ознайомитися з основними принципами побудови мультиагентних експертних систем

Теоретичні відомості

Багато- або мультиагентні інтелектуальні системи на даний момент дуже популярні, розповсюджені і є предметом особливої уваги серед розробників та науковців. Різноманітність реалізацій та принципові розбіжності цих реалізацій обумовлені відсутністю особливих правил по створенню даних систем, а також можливістю об’єднувати різні методи представлення знань, реалізуючи агентів з різними видами баз знань.

Для прикладу, змалюємо хід розробки мультиагентної експертної системи для процедурної побудови лабіринту, який було наведено в лабораторній роботі №3 –"Семантична модель представлення знань". Для початку, виділимо з системи потенційних агентів – такими можуть бути:

– Агент – гравець, який ходить по лабіринту.
– Поле.
– Собака.
– Енергетичний блок.
– Ваза.

І додамо два елементи, яких не було у згаданій лабораторній роботі:

– Спаун-точка для агента-гравця (місце виникнення агента-гравця).
– Майстер-наглядач.

Наступний крок – визначення ролей у системі кожного агента. В даному випадку можна сказати, що кожен агент – це генератор певних елементів, окрім агента-гравця та майстра-наглядача. Таким чином:

– Агент "поле" генерує об’єкт "поле", який являє собою певним чином розміщені порожні комірки та комірки стіни. В задачі цього агента входить не лише довільно розмістити комірки, але й розмістити таким чином, щоб з точки

X_{min},Y_{min} можливо було дістатися в комірку X_{max},Y_{max} нашого поля. Дане питання можливо вирішувати різноманітними методами, один з яких – сприймати поле як матрицю розміром X_{max} на Y_{max}, де 1 це стінка, а 0 це порожня комірка. За цією матрицею можливо побудувати граф і матрицю суміжності графа і визначити, чи можливо пройти від однієї точки до іншої (див. теорія графів, матриця суміжності). Тут не будуть розглядатися певні алгоритми вирішення даної проблеми, але правила для агента розробити необхідно, тому умовою для розміщення пустої комірки буде така умова – «коли у матриці суміжності її значення буде 1 або більше», що означає, що у даної комірки є хоча б одне ребро для переходу в іншу. До того ж у графі не повинно бути петель.

– Агент "спаун-точка" – повинен розмістити точку так, щоб вона обов’язково була в порожній комірці.

– Агент "собака" – повинен розміститися в порожній комірці і повинен бути як мінімум за одну порожню клітинку від точки спауну.

– Агент "енергетичний блок" – повинен розміститися в порожній комірці.

– Агент "ваза – повинен розміститися в порожній комірці як мінімум за одну комірку від точки спауну.

– Агент "майстер-наглядач" – повинен розіслати команди іншим агентам на генерування своїх об’єктів, лише певну кількість разів. Для спаун-точки – одну, для інших – таку кількість разів, як зазначено, наприклад, в конфігураційному файлі. До того ж "майстер-наглядач" повинен надсилати команди агентам лише після того, як завершив свою роботу агент "поле", і мати в своїй пам’яті матрицю розміщення всіх елементів, що була згенерована агентами, для того, щоб відповідати на питання агентів чи є певні елементи в певних координатах. «Майстер-наглядач» повинен командувати агенту-гравцю робити наступний крок, а також повідомляти чи не повинен він з’явитися у точці спауну.

– Агент "гравець" повинен з’являтися в точці спауну, робить крок за командою майстра-нагядача. Поведінка агента в полі визначається правилами, що

описані в лабораторній роботі №3 –"Семантична модель представлення знань".

На рисунку 5.1 зображено схему взаємодії агентів у системі.

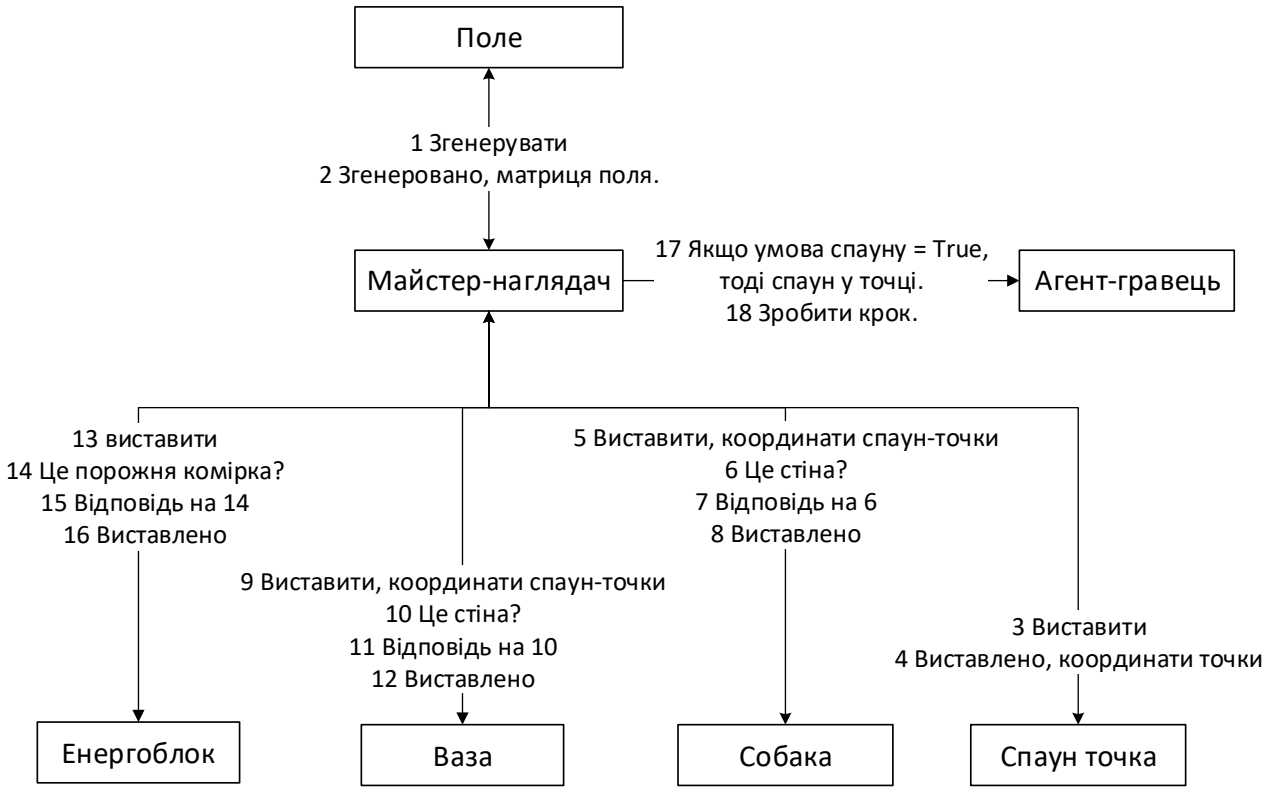

Рисунок 5.1 – Схема взаємодії агентів у системі

Відповіді і запити для зручності пронумеровано.

Кожен з агентів може бути реалізований зі своєю моделлю представлення знань, в залежності від необхідності та особливостей реалізації.

Завдання: Розробити мультиагентну інтелектуальну систему. Остання цифра залікової книжки – кількість агентів у системі (якщо менше 5, то 5). Серед агентів повинно бути розподілено не менше 3-х різних моделей представлення знань.

Контрольні питання:

1. Що собою представляє мультиагентна ЕС?
2. Яким чином можна представити схему взаємодії агентів у мультиагентній ЕС?

Лабораторна робота №6

Тема: Експертні системи на основі результатів когнітивного моделювання

Мета: Ознайомитися з основними принципами побудови експертних систем на основі результатів когнітивного моделювання

Теоретичні відомості

З розвитком методів побудови експертних систем та постійним збільшенням потужності обчислювальної техніки перед експертними системами почали ставити задачі, для вирішення яких до недавнього часу навіть і не розглядалися варіанти їх використання.

Такі задачі найчастіше виникають, коли предметна область, в якій працює система, складається з декількох інших предметних областей, в яких існують свої певні експертні системи. Така система чимось схожа на мультиагентну, але відрізняється взаємозв'язками між «агентами», а також тим, що предметні області не відокремленні одна від одної.

В когнітивному моделюванні використовується поняття *фактор*. Кожен фактор може бути з будь-якої предметної області і може впливати на будь-яку кількість інших факторів з будь-якої іншої предметної області. Предметні області не виділяються в окремі елементи системи. Таким чином когнітивне моделювання можна розділити на два етапи:

– Виділення з предметних областей факторів.
– Визначення впливу фактору на інші фактори системи.

Неможливо визначити чіткий порядок цих етапів: немає сенсу додавати в систему фактор, який не впливає на інші; неможливо визначити вплив фактору на інші, не виділивши його з предметної області.

Розробимо модель експертної системи з використанням принципів когнітивного моделювання. Нехай система-модель повинна відображати стан певної країни. Країна – система складна, основними показниками країни є економічні. Економічний стан країни формується на основі багатьох факторів, з

багатьох різних предметних областей. Наприклад, при зростанні якості медицини зростає кількість здатних до роботи людей, зростають і економічні показники. Зростання економічних показників дасть змогу знизити певні податки, що дозволить поліпшити економічне становище робітників, стимулюватиме бізнес, оскільки надлишок грошей працівникам треба буде кудись витратити. В свою чергу, розвиток бізнису призведе до покращення якості соціальних послуг. Покращення якості соціальних послуг може призвести до потоку мігрантів (якщо соціальні послуги до цього не були жахливими). Поток мігрантів призведе до конкуренції серед робочої сили, а отже і до збільшення її якості. Також, наприклад, можна зменшити податкове навантаження на певні види приватного бізнесу, що теж позитивно відобразиться на економічному становищі країни, і так далі.

Як бачимо, в такій складній системі як країна, навіть, невеликі зміни призводять до великої кількості наслідків (якщо ці зміни реальні). Для складних систем варто притримуватись певних правил, певної системи, яка найбільш підходить для поточної задачі. Одним з універсальних методів побудови можуть бути об’єктно-орієнтовні принципи, вони дозволять зберігати певний визначений рівень деталізації моделі. Таким чином виділимо для початку основні 3 об’єкти, які нас цікавлять (побудова даної моделі наведена тут для прикладу когнітивного моделювання і може ніяк не відноситись до реальних економічних процесів):

– Медицина.
– Технології.
– Освіта.

Отже, зараз в нас є три об’єкти, які в контексті моделі і на даному етапі її побудови можна назвати факторами. Назвемо їх для зручності факторами якості медицини, факторами якості технологій та факторами якості освіти. Важко не погодитись, що із зростанням кожного з цих факторів будуть зростати і економічні показники країни. Зобразимо це на схемі (рисунок 6.1):

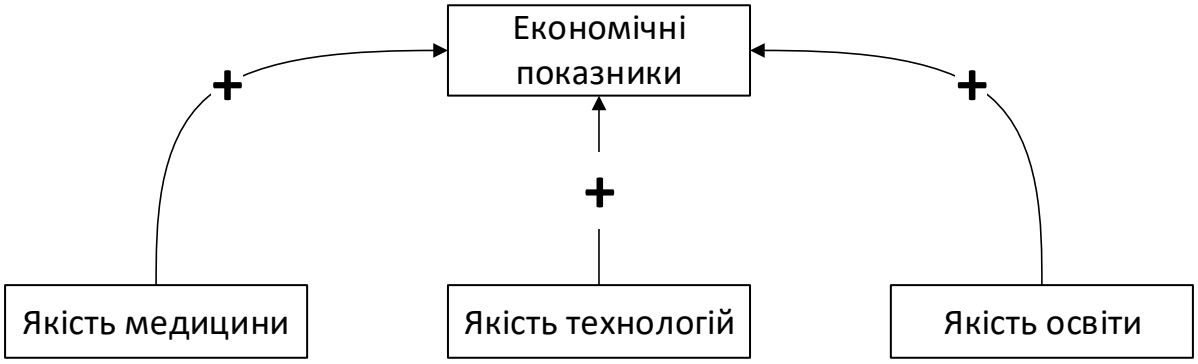

Рисунок 6.1 – Вплив трьох факторів на економічні показники країни

Але дана модель занадто проста і, хоч вона логічна, користі ніякої не несе. Тому необхідно збільшити деталізацію кожного фактору. Для цього виділимо з кожної предметної області 3 додаткових фактори:

1. Якість медицини:
   - Якість медперсоналу.
   - Якість медпрепаратів та апаратів.
   - Вартість.
2. Якість технологій:
   - Якість наукового потенціалу.
   - Якість співробітників.
   - Вартість.
3. Якість освіти:
   - Якість персоналу.
   - Якість студентів.
   - Вартість.

Зобразимо тепер ці 9 факторів на схемі (рисунок 6.2):

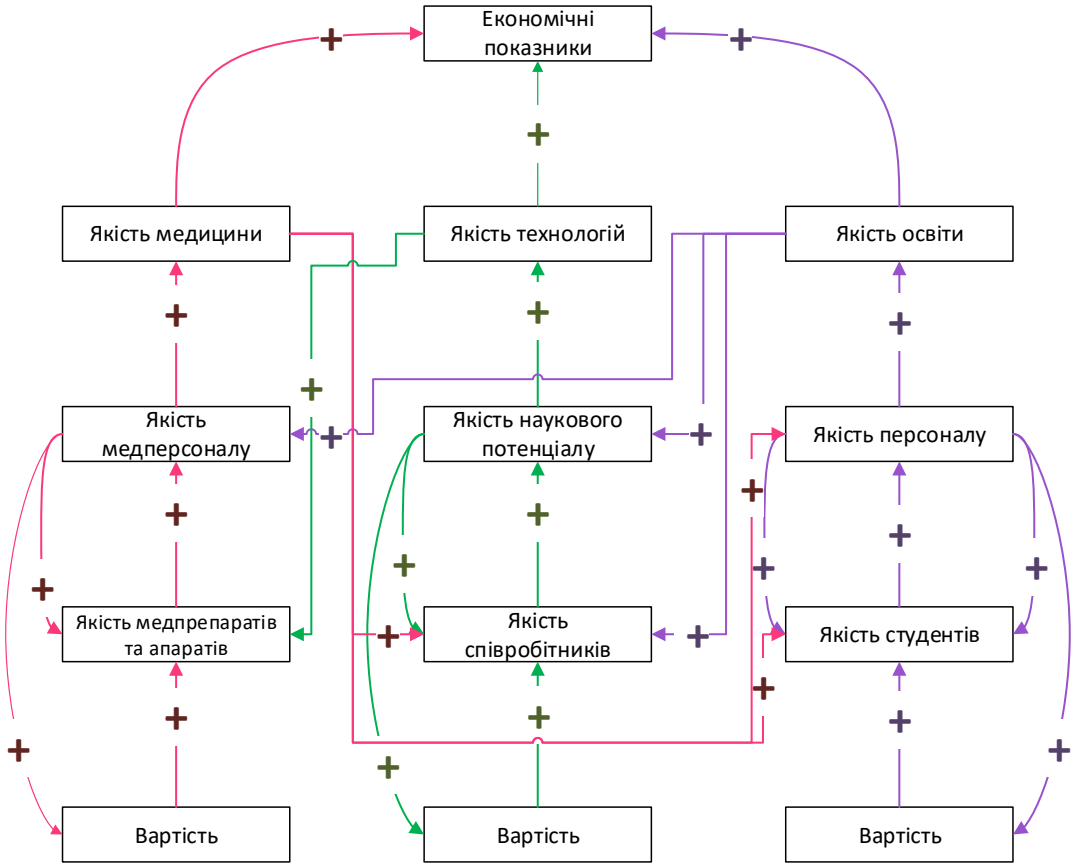

Рисунок 6.2 – Взаємодія дев'яти факторів

Як видно зі схеми тепер, завдяки більшому рівню деталізації можна встановити взаємозв’язок між 3 предметними областями, чітко визначити ті фактори, які більше чи менше впливають на систему в цілому.

Продивившись уважно схему, можна зробити досить логічний висновок, що було б доречно додати ще один фактор в кожну з предметних областей – "доступність", тоді збільшення вартості буде негативно впливати на доступність, і доступність в різних предметних областях по-різному буде впливати на загальну систему. Так, зниження доступності освіти призведе до підвищення якості освіти, а, отже, і до підвищення наукового потенціалу, але може знизити якість співробітників загалом, оскільки співробітників з необхідним рівнем освіти може просто не вистачати, але ж, з іншого боку, всі ті співробітники, які мають необхідний рівень освіти, мають його не лише на паперах, отже, в даному випадку маємо одвічну боротьбу між якістю та

кількістю. Дану проблему можна вирішити двома способами: 1) додаванням нових факторів, 2) або введенням «ваги» впливу факторів, що, в свою чергу, можливо зробити використовуючи методи на зразок SWOT-аналізу.

Завдання: Розробити експертну систему, використовуючи методи когнітивного моделювання, що містить від 5 до 20 факторів. (Остання цифра залікової книжки * передостання цифра залікової книжки) – кількість необхідних факторів (якщо менше 5, то 5; якщо більше 20, то 20). Тип експертної системи – довільний.

Контрольні питання:

1. Що таке когнітивне моделювання?
2. На які етапи можна розділити когнітивне моделювання?
3. Як здійснюється когнітивне моделювання?
4. Що означає поняття "фактор" в когнітивному моделюванні?
5. Яким чином на основі результатів когнітивного моделювання можна побудувати експертну систему?

Лабораторна робота №7

Тема: Методи одержання знань. Інженерія знань

Мета: Ознайомитися з методами збору знань

Теоретичні відомості

Незважаючи на обраний тип чи типи моделей представлення знань в експертній системі, базу знань необхідно заповняти знаннями, які б ефективно нею використовувались. Для цього, звісно ж, задіюють експертів в певній проблемній області. Але експерт, найчастіше, не зможе передати свої знання в тому форматі, який необхідний розробникам ЕС. Для вирішення цієї проблеми в команді розробників необхідна посада інженера по знанням, або когнітивного інженера.

Основна, та найчастіше єдина, задача когнітивного інженера полягає у зборі та формулюванні знань у необхідному для розроблюваної ЕС форматі. Оскільки сам інженер по знанням не є експертом або спеціалістом у предметній області, що досліджується, а експерти предметної області, зазвичай, не є спеціалістами у розробці ЕС, інженеру доведеться організувати свою роботу таким чином, щоб дати змогу експертам самостійно визначити та надати необхідні для розробки бази знань знання, але, в той же час, не будучи ні спеціалістом, ні експертом в предметній області, що досліджується, інженер повинен контролювати якість отримуваних знань та даних. В зв'язку із складністю проблеми, для її вирішення було розроблено велику кількість методів по збору інформації, і, у зв'язку з різноманітністю варіантів даної проблеми, набір методів вирішення проблеми може значно відрізнятися.

Загалом, методи пошуку знань поділяються на дві великі групи:

– **Методи пошуку явних знань.**

– **Методи пошуку знань прихованих (неявних).**

Про явні знання відомо експертам, про неявні експерти можуть не здогадуватись.

На рисунку 7.1 зображено схему класифікації методів пошуку/виділення явних знань.

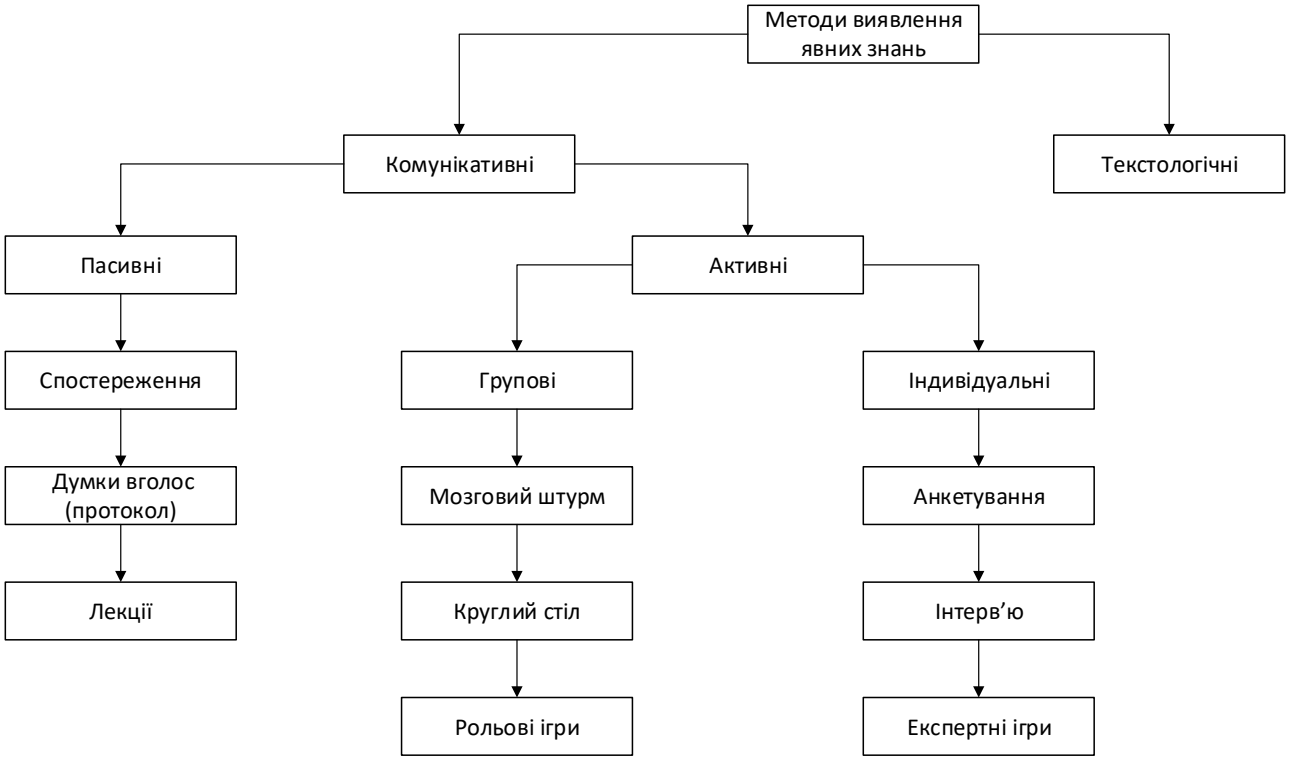

Рисунок 7.1 – Класифікація методів виявлення явних знань

Методи пошуку явних знань поділяються теж на дві групи:

– **Комунікативні.**

– **Текстологічні.**

Комунікативні методи засновані на анкетуванні, спілкуванні з експертом та групами експертів. Поділяються на пасивні та активні; при використанні пасивних методів головну роль виконує експерт, при активних – когнітивний інженер. При розробці баз знань, зазвичай, ці методи комбінуються. Активні методи, в свою чергу, поділяються на групові та індивідуальні. При використанні групових методів знання отримують від групи експертів. На практиці більш розповсюдженими є індивідуальні методи, коли знання отримують від одного експерта за 1 раз.

Текстологічні методи опираються на вивчення підручників, документів та спеціальної літератури з предметної області.

Серед методів пошуку неявних знань можна назвати:

– **Метод багатовимірного оцінювання** – заснований на методах статистики. Полягає в оцінюванні схожості експертних оцінок певного об’єкта. Результати опрацювання представляються у вигляді точок деякого координатного простору.

– **Метод метричного оцінювання** – створення нових класів з використанням метричних відстаней. Цей тип обробки даних орієнтовано на максимальне зближення числових значень двох матриць. Важливо нагадати, що даний метод не виявляє нових знань, об’єктів, а виявляє нові простори, а отже його результати необхідно інтерпретувати як відновлену структуру розміщення точок у площині, на координатній площині чи в просторі.

– **Метафоричний підхід** – орієнтований на виявлення прихованих знань з практичного досвіду експерта та заснований на порівнянні об’єктів предметної області з абстрактними об’єктами зі світу метафор. У результаті застосування даного методу можливо виявити нові властивості об’єктів, що аналізуються, а також виявити ставлення експерта до цих об’єктів. При застосуванні даного методу експерт виходить за рамки об’єктивності та діє з точки зору свого суб’єктивного уявлення.

– **Метод репертуарних решіток** – даний метод широко використовуваний у психології. Полягає у заповненні експертом матриці, стовпцями якої є певні будь-які об’єкти, рядками матриці є «конструктори», які представлені біполярними ознаками, що можуть бути звуками, характерною поведінкою об’єкта, кольором і тощо. Іншими словами, конструктор – це така характеристика об’єкта, яку можливо використовувати для розподілу об’єктів на класи. Наприклад «холодний - гарячий», «провідний – безпровідний», «мережевий – локальний» тощо.

Всі знання, отримані від експертів, в результаті повинні відповідати метрикам цілісності та метрикам експертної системи загалом.

Завдання: Розробити експертну систему довільного типу, але під час формування її бази знань необхідно використати знання експертів або експерта в певній предметній області. Знання в експерта повинні бути зібрані з використанням як мінімум трьох методів збору явних знань та одного методу збору неявних знань.

Контрольні питання:

1. Які існують методи одержання знань?
2. В чому полягає робота когнітивного інженера?
3. Що таке явні знання? Що таке неявні знання?
4. Які існують методи виявлення явних знань?
5. Які існують методи виявлення неявних знань?

СПИСОК ЛІТЕРАТУРИ

1. Болотова Л. С. Системы искусственного интеллекта: модели и технологии, основанные на знаниях / Л. С. Болотова – Москва: «Финансы и Статистика», 2012. – 663

2. Терано Т. Прикладные нечеткие системы. : Пер. с яп. / Терано Т., Асаі К., Сугено М., Чернишова Ю. М.: «Мир» - Москва, 1993. 184 с.: іл.

3. Нікольський Ю. В. Дискретна математика: підруч. [для студ. вищ. навч. закл.] / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина – Київ : «Видавнича група BHV», 2007. – 368 с.: іл.

4. Глибовець М.М., Олецький О.В. Штучний інтелект: Підручник для студентів, що навчаються за спец. "Комп'ютерні науки" та "Прикладна математика". – К.: Вид. дім "КМ Академія", 2002. – 366 с.

5. Триумфгородских М.В. Дискретная математика и математическая логика для информатиков, экономистов и менеджеров. Учебное пособие для ВУЗов. – М.: Диалог-МИФИ, 2011. –180 с.

6. Нікольський Ю.В., Пасічник В.В., Щербина Ю.М. Дискретна математика. Підручник для вищих навчальних закладів. – К.: Видавнича група BHV, 2007. – 368 с.

7. Лямец В.И., Успенко В.И. Основы общей теории систем и системный анализ. Учебное пособие – Харьков: «БУРУН и К», Киев: ООО «КНТ», 2015. – 304 с.