Тут буде тітулка

Тут буде лист завдання

Тут буде реферат

ЗМІСТ

[ВСТУП 6](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761582)

[1 ЗАГАЛЬНИЙ РОЗДІЛ 7](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761583)

[1.1 Експертні системи 7](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761584)

[1.1.1Класифікація експертних систем 7](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761585)

[1.1.2 Склад експертної системи 9](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761586)

[1.1.3 Етапи створення експертної системи 10](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761587)

[1.1.4 Переваги та недоліки експертних систем 12](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761588)

[1.1.5 Моделі представлення знань у експертних системах. 14](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761589)

[1.2 Засоби розробки експертних систем 15](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761590)

[1.2.1 Мови програмування 15](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761591)

[1.2.2 Мови інженерії знань 16](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761592)

[1.2.3 Допоміжні засоби для побудови систем 18](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761593)

[1.2.4 Середовище підтримки інструментів 22](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761594)

[1.3 Алгоритм пошуку по дереву в прийнятті рішень 23](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761595)

[2 СПЕЦІЛЬНА ЧАСТИНА 27](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761596)

[ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 28](file:///D:\Maxim\Documents\GitHub\expert-systems\acourse\Курсова_робота_експертні_системи.docx#_Toc104761597)

ВСТУП

# 1 ЗАГАЛЬНИЙ РОЗДІЛ

## Експертні системи

Експертні системи (ЕС) є галуззю прикладного штучного інтелекту (ІІ) та були розроблені спільнотою ІІ у середині 1960-х років. Основна ідея ЕС полягає в тому, що експертиза, яка є великою сукупністю знань з конкретних завдань, передається від людини до комп'ютера. Це знання потім зберігається на комп'ютері і користувачі використовують його для конкретних порад при необхідності. Комп'ютер може зробити висновки та дійти конкретного рішення. Потім як людина-консультант, дає поради та пояснює, якщо необхідно логіку порад [1]. ЕС надає потужні та гнучкі засоби для отримання рішень різних проблем, які часто не можуть бути вирішені іншими, більш традиційними та ортодоксальними методами. Таким чином, їх використання поширюється на багато секторів нашого соціального та технологічного життя, де їх застосування є критично важливим у процесі підтримки прийняття рішень та вирішення проблем [2].

### 1.1.1Класифікація експертних систем

Існує багато способів як можна класифікувати ЕС, ось деякі із цих способів класифікації:

Класифікація експертних систем, що відображає специфіку задач, які вирішуються за допомогою цієї технології [3]:

1. Інтерпретуючі системи призначені для формування опису ситуацій за результатами спостережень чи даними, одержаними від різноманітних сенсорів [3].
2. Прогнозуючі системи призначені для логічного аналізу можливих наслідків заданих ситуацій чи подій [3].
3. Діагностичні системи призначені для виявлення джерел несправностей за результатами спостережень за поведінкою контрольованої системи [3].
4. Системи проектування призначені для структурного синтезу конфігурації об'єктів за заданих обмежень [3].
5. Системи планування призначені для підготовки планів проведення послідовності операцій, що веде до заданої мети [3].
6. Системи моніторингу аналізують поведінку контрольованої системи та порівнюючи отримані дані з критичними точками заздалегідь складеного плану, прогнозують ймовірність досягнення поставленої мети [3].
7. Налагоджувальні системи призначені для вироблення рекомендацій щодо усунення несправностей у контрольованій системі [3].
8. Системи надання допомоги під час ремонту обладнання виконують планування процесу усунення несправностей у складних об'єктах, наприклад у інженерних мережах комунікацій [3].
9. Навчальні системи проводять аналіз знань студентів з певного предмета, знаходять прогалини в знаннях і пропонують засоби для їх ліквідації [3].
10. Системи контролю забезпечують адаптивне управління поведінкою складних людино-машинних систем, прогнозуючи появу можливих збоїв та плануючи дії, необхідні для їх попередження [3].

За типом використання розрізняють:

1. Ізольовані ЕС [4].
2. Гібридні ЕС інтегровані з базами даних та іншими програмними продуктами (додатками) [4].

За рівнем залежності від зовнішнього середовища виділяють:

1. Статичні ЕС, які залежать від зовнішнього середовища [4].
2. Динамічні, що враховують динаміку зовнішнього середовища, та призначені для вирішення завдань у реальному часі [4].

### 1.1.2 Склад експертної системи

Експертні системи відрізняються між собою своєю складністю та кількістю компонентів, але у кожної ЕС є чотири базові компоненти:

1. База знань.
2. Механізм виведення.
3. Модуль набуття знань.
4. Пояснювальний інтерфейс.

База знань (БЗ) містить факти (або затвердження) та правила. Факти – це короткострокова інформація, яка може швидко змінюватись. Правила – це довгострокова інформація про те як генерувати нові факти або гіпотези з того, що зараз відомо. Основна відмінність в тому, що основа знань більш креативна. Факти в базі даних зазвичай пасивні: вони або там є або їх там нема. З іншого боку, база знань активно намагається заповнити інформацію, якої бракує [5].

Механізм виведення – це програма, яка модулює те, яка саме інформація буде виводитись, ґрунтуючись на фактах та правилах, які знаходяться у БЗ. Існують два механізми виведення інформації: прямий ланцюжок - включає міркування від даних до гіпотез, та зворотний ланцюжок - намагається знайти дані для підтвердження або спростування гіпотези. Чистий прямий ланцюжок призводить до несфокусованих питань у системі діалогового режиму, тоді як чистий зворотний ланцюжок має тенденцію бути досить невтомним у своїх цілеспрямованих питаннях [5].

Модуль набуття знань – модуль, до якого залучається експерт зі знань, необхідний для деякої ЕС та інженер зі знань, який має навички у роботі із програмою. Через співпрацю цих людей йде систематизація знань та рішень експерта та доведення їх до мови, яку розуміє програма [5].

Пояснювальний інтерфейс – програма, або їх комплекс, за допомогою якої можливий діалог між користувачем та експертною системою [5].

### 1.1.3 Етапи створення експертної системи

Технологія розробки експертних систем включає шість наступних етапів:

1. Ідентифікація.
2. Концептуалізація.
3. Формалізація.
4. Впровадження.
5. Тестування.
6. Дослідна експлуатація [6].

На етапі ідентифікації процесу розвитку планується прототип системи [7], джерела знання (книги, експерти та методології), цілі (розподіл досвіду, автоматизація рутинних операцій), класи розв'язуваних завдань та багато іншого. Результатом ідентифікації є відповідь на питання, що потрібно зробити і які ресурси потрібно використовувати.

На етапі концептуалізації проводиться поглиблений аналіз проблемної галузі, визначаються методи вирішення проблеми. Цей етап завершується створенням моделі предметної області (програмного забезпечення), що включає основні поняття та відносини [8]. На етапі концептуалізації визначаються такі характеристики проблеми:

1. Доступні типи даних;
2. Вхідні та вихідні дані під задачі загальної проблеми;
3. Застосовані стратегії та гіпотези;
4. Типи відносин між програмними об'єктами, типи використовуваних відносин;
5. Процеси, що використовуються у рішеннях;
6. Зміст знань, що використовуються під час вирішення проблем;
7. Типи обмежень, що накладаються на процеси, що використовуються в рішеннях;
8. Зміст знань, що використовується у поясненнях рішень [8]

На етапі формалізації вибираються і визначаються шляхи уявлення всіх видів знань, формалізуються основні поняття, способи пізнання визначеної інтерпретації, моделювання роботи системи, адекватність цілей, що стосуються системи фіксованого визначення, оцінюється методи вирішення, способи презентації [9]. Вихід стадії формалізації є описом того, як проблеми можуть бути представлені у вибраному або розвиненому формалізмі (фрейми, сценарії, семантичні мережі та ін.) та розуміння методів маніпулювання цим знанням [10].

На етапі впровадження наповнюється база даних з експертом. Процес набуття знань виконується інженером зі знань на основі дії експертів щодо реальних рішень проблем. Мета цього етапу – створення одного прототипу ЕС. В подальшому за цей етап за результатами тестування та дослідної експлуатації створюється кінцевий продукт, придатний для промислового використання[11]. Розробка прототипу складається з програмування компонентів або їх вибір серед відомих програмних інструментів та наповнення бази знань [12]. Таким чином, в даний час процес оцінки ситуації та прийняття рішень є одним з найбільш трудомістких, тому методологія розробки експертних систем вважає наступні потреби найважливішими:

1) Правильна постановка задачі;

2) Систематизація знань для передачі їх в комп'ютерну систему;

3) Розробка інструментів управління знаннями бази даних, індуктивний висновок та спрощений діалог.

### 1.1.4 Переваги та недоліки експертних систем

Експертні системи найбільш поширені у складній проблемній галузі і вважаються альтернативами, що широко використовуються при пошуку рішень, що вимагають наявності спеціальних людських знань. Експертна система також здатна обґрунтовувати запропоновані нею рішення на основі знань та даних, отриманих від попередніх користувачів [13].

Відмінність експертної системи від звичайної системи вирішення завдань у тому, що остання є системою, у якій кодуються програми і структури даних, у той час як для експертної системи жорстко закодовані лише структури даних, і жодна специфічна для проблеми інформація не кодується у структурі програми. Натомість знання про людський досвід фіксуються та систематизуються в процесі, відомому як інженерія знань. Отже, щоразу, коли конкретна проблема вимагає допомоги певного людського досвіду для вирішення, систематизований людський досвід використовуватиметься і оброблятиметься для забезпечення раціонального та логічного рішення. Ця експертна система, що базується на знаннях, дозволяє системі часто додавати нові знання і відповідним чином адаптуватися до нових вимог постійно мінливого та непередбачуваного середовища [13].

Переваги використання експертної системи:

1. Надання послідовних рішень. Вона може давати послідовні відповіді на повторювані рішення, процеси та завдання. Поки основа правил у системі залишається постійною, незалежно від того, скільки разів тестуються подібні завдання, зроблені висновки залишаться тими самими [13].
2. Дає розумні пояснення: може прояснити причини, з яких було зроблено висновок і чому вона вважає найбільш логічним вибір серед інших альтернатив. Якщо є якісь сумніви у постановці певного завдання, вона запропонує користувачам відповісти на деякі питання, щоб опрацювати логічний висновок [13].
3. Долає людські обмеження: ЕС не має людських обмежень і може працювати цілодобово безперервно. Користувачі зможуть часто використовувати її для пошуку рішень. Знання експертів є безцінним активом для компанії. Вона може зберігати знання і використовувати їх, доки вони потрібні організації [13].
4. Легко адаптується до нових умов. На відміну від людей, у яких часто виникають проблеми з адаптацією до нових умов, експертна система має високу адаптивність і може відповідати новим вимогам за короткий період часу. Вона також може отримувати нові знання від експерта та використовувати їх як правила виводу для вирішення нових проблем [13].

Недоліки використання експертної системи:

1. Бракує здорового глузду: у ЕС відсутній здоровий глузд, необхідний для прийняття деяких рішень, оскільки всі прийняті рішення засновані на правилах виведення, встановлених у системі. Вона також не може давати творчі та новаторські відповіді, як люди-експерти у незвичайних обставинах [13].
2. Висока вартість впровадження та обслуговування: впровадження експертної системи в бізнесі буде фінансовим тягарем для невеликих організацій, оскільки воно пов'язане з високими витратами на розробку, а також подальшими періодичними витратами на модернізацію системи для адаптації до нового середовища [13].
3. Складність створення правил виведення: експерти предметної галузі не завжди зможуть пояснити свою логіку та міркування, необхідні для процесу інженерії знань. Таким чином, завдання кодування знань дуже складне і може вимагати великих зусиль [13].
4. Може надати неправильні рішення: ЕС не безпомилкова. При обробці можуть виникнути заблудження через деякі логічні помилки, допущені в базі знань, які потім будуть давати неправильні рішення [13].

### 1.1.5 Моделі представлення знань у експертних системах.

Семантична мережа представляє собою інформаційну модель предметної області (сукупність фактів і тверджень із бази даних) і має вигляд графа, вершини якого відповідають об’єктам предметної області, а дуги- відношенням між ними. Семантичні мережі ще називають асоціативними мережами, оскільки одні вузли в таких мережах асоційовані або зв’язані з іншими. В оригінальній роботі Квілліана людська пам’ять була змодельована як асоціативна мережа, в якій поняття були представлені у вигляді вузлів, а зв’язки показували, як з’єднані ці поняття одне з одним. Згідно з даною моделлю, якщо відбувається симуляція одного вузла в результаті читання слів у реченні, то відбувається активація зв’язків цього вузла з іншими вузлами, і така активність розповсюджується по всій мережі. Наприклад, очевидно, що людина знає тисячі слів, але в її свідомості відображаються лише слова того конкретного речення, яке вона читає [14].

В семантичних мережах використовують два типи зв’язків:

1S-A та A-KIND-OF (АКО), де зв’язок IS-А означає «є екземпляром» даного класу і вказує на конкретний екземпляр деякого класу. В цьому контексті поняття класу близьке доматематичного поняття множини, оскільки означає групу об’єктів. Але множина може містити елементи будь-якого типу, а об’єкти в класі пов’язані певними відношеннями один з одним, тобто не можуть бути взяті довільно. Але при представленні речення у вигляді семантичної мережі не можна обмежитися зв’язками типу IS-А та A-K1ND-OF (АКО), так як такі зв’язки не можна вжити до дієслів та службових частинок речення. А всі слова в реченні неможливо представити у вигляді класів, так як у реченні простежується залежність між різними словами, а не ієрархічна залежність слів від якогось окремого вузла. Проте, чим складніше речення і чим більше слів входить до його складу, тим більше нагромадження зв’язків у мережі і тим незрозуміліша вона для користувача. Головний недолік представлення речення у вигляді семантичної мережі - це відсутність запитальних слів між службовими частинами мови та головними і другорядними членами речення. Тоді на семантичній мережі або взагалі не потрібно показувати зв’язки, що відходять від сполучників та прийменників, або включати їх в один вузол із словами, перед якими ці службові частини мови вживаються [14].

Продукційна модель представляє собою сукупність логічних висловлень, продукційних правил або продукцій. Для даної задачі продукційна модель буде мати вигляд:

«Посилання: експерт не може пояснити процес вирішення задачі

Посилання: експерт не може пояснити прийняття того чи іншого рішення

Заключения: знання експерта не можна представити в експертній системі у вигляді явних знань»

Бачимо, що в даній моделі зміст речення не враховується, має значення тільки форма і зовнішнє представлення. Тобто початковий рядок - набір символів, що представляють вхідні дані, згідно з продукційним правилом, закладеним у експертній системі, перетворюється на вихідний рядок символів [14].

Застосування продукційної моделі для представлення речення має ряд недоліків: по-перше, не відокремлює з даного речення знання, по-друге, отриманий результат залежить лише від наперед заданих правил, і якщо система не знайде відповідного посилання, то результат не буде отримано, по-третє, відсутність стратегії управління, тобто хаотичне розміщення правил у системі, уповільнює роботу на час пошуку необхідного посилання. Застосування продукційної моделі не для складнопідрядного умовного речення, а, наприклад, для складносурядного чи простого, неможливе, так як структура цих речень не дозволяє створити посилання і заключения [14].

Фреймові моделі використовують будь-яке уявлення щодо предмету, об’єкту або стереотипної ситуації. Фрейм являє собою складову структурну одиницю, призначену для опису, це сукупність слотів і заповнювачів, які визначають об’єкт, що розглядається в якості стереотипу. Для даного речення фреймова модель буде виглядати наступним чином:

«Фрейм-поняття: ЕКСПЕРТІ <що робить?, що?, чого?, чого?>;

Фрейм-поняття: ЕКСПЕРТ2 <що робить?, що?, якого?, чого?>;

Фрейм-поняття: НЕ\_МОЖНА\_ПРЕДСТАВИТИ <що?, кого?, в чому?, в якій?, як?, яких?>.

Фрейм-приклад: ЕКСПЕРТ1 <не може пояснити, процес, вирішення, задачі;

Фрейм-приклад: ЕКСПЕРТ2 <не може пояснити, прийняття, того чи іншого, рішення>;

Фрейм-приклад: НЕ\_МОЖНА\_ПРЕДСТАВИТИ <знання, експерта, в системі, експертній, у вигляді знань, явних>»

Фреймові моделі забезпечують упорядковане представлення знань, доступне для розуміння, а також можливість розміщення у якості слотів ознак виклику обчислювальних процедур. Проте застосування фреймової моделі для речення не дає можливість повністю передати його зміст, тобто отримати знання, фрейм описує лише частину складного речення, зміст сполучників «якщо...то» втрачається. Також недоліком фреймових структур є необмежена модифікація і знищення слотів, тому немає можливості створювати складні об’єкти на основі більш простих [14].

Логіко-лінгвістична модель представляє собою вираз або вирази у вигляді термінів природної мови та лінгвістичних змінних. Основою для логіко-лінгвістичних моделей служить логіка предикатів, де предикат - це функція Р, яка приймає значення 0 або 1, і аргументи якої пробігають значення із довільної множини М: Р(х1,х2, ...хn))\4]. Для речення «Якщо експерт не може пояснити процес вирішення задачі і прийняття ним того чи іншого рішення, то знання експерта не можна представити в експертній системі у вигляді явних знань» логіко-лінгвістична модель матиме вигляд:

VxVyVzVr[l ПОЯСИ {х, (у & z)) -» 1ПРЕДСТ(ЗНАН(х), ЕКСЩх), r)J

де х - експерт;

у - процес вирішення задачі;

z - прийняття рішення;

г - явні знання;

ПОЯСИ - предикат, що означає можливість пояснення; ЗНАН(х) - експертні знання; ЕКСП (х) - експертна система; ПРЕДСТ - предикат, що означає можливість представлення. Бачимо, що логіко-лінгвістична модель здійснює контроль цілісності речення, чітко описує зміст речення і забезпечує доступне представлення знань. Недоліком логіко-лінгвістичних моделей є те, що знання важко структурувати, проте для кінцевого користувача ця модель є найбільш зрозумілою при розв’язанні задачі представлення складного речення [14].

## 1.2 Засоби розробки експертних систем

Засоби розробки експертної системи - це системи програмування, які спрощують роботу з побудови експертної системи. Вони варіюються від мови програмування дуже високого рівня засобів підтримки до низького рівня [15].

### 1.2.1 Мови програмування

Найбільш важливими мовами програмування, що використовуються для експертних систем, зазвичай є або проблемно-орієнтовані мови, такі як FORTRAN і PASCAL, або мови обробки символів, такі як LISP і PROLOG. Проблемно-орієнтовані мови призначені для вирішення певних класів завдань: наприклад, FORTRAN має зручні функції для виконання алгебраїчних обчислень і найбільш застосовний до наукових, математичних та статистичних проблем. Мови маніпулювання символами призначені для додатків штучного інтелекту: наприклад, LISP має механізм маніпулювання символами у вигляді структур списку. Список — це просто набір елементів, укладених у круглі дужки, де кожен елемент може бути символом або іншим списком. Найбільш популярною і широко використовуваною мовою програмування для програм експертних систем є LISP. Такі мови маніпулювання символами більше підходять для роботи зі штучним інтелектом, хоча кілька експертних систем були написані проблемно-орієнтованими мовами, такими як FORTRAN та PASCAL. ЕС зазвичай пишуться такими мовами, як LISP і PROLOG або навіть CLIPS (інтегрована виробнича система мовою C, розроблена в середині 1980-х років). Використання цих мов для розробки ЕС спрощує процес кодування. Основними перевагами цих мов у порівнянні зі звичайними мовами програмування є простота додавання, виключення або заміни нових правил та можливості керування пам'яттю [15].

Деякі з характерних рис мов програмування, необхідні для розробки ЕС:

1. Ефективне поєднання цілих та речових змінних.
2. Хороші процедури керування пам'яттю.
3. Великі процедури маніпулювання даними.
4. Інкрементна компіляція.
5. Архітектура пам'яті з тегами.
6. Оптимізація системного середовища
7. Ефективні процедури пошуку.

Мови програмування, такі як LISP, CLIP і т. д., пропонують найбільшу гнучкість розробнику експертної системи і можуть дати вказівки у тому, як представляти знання чи механізми доступу до бази знань. З іншого боку, мови інженерії знань, такі як KAS пропонують невелику гнучкість, оскільки розробник системи повинен використовувати схему управління, визначену готовим механізмом виведення. Однак вони надають рекомендації щодо подання та готові механізми виведення для управління використанням бази знань [15].

### 1.2.2 Мови інженерії знань

Мова інженерії знань - це складний інструмент для розробки експертних систем, що складається з мови побудови експертних систем, інтегрованої в середовище підтримки. Мова програмування - це розробка штучної мови для отримання знань, прийняття/відхилення знань контролю та управління роботою комп'ютера [15].

Мова інженерії знань - це тип мови програмування, призначений для побудови та налагодження експертної системи, мова надає певні можливості для побудови експертної системи; вони гнучкіші, ніж інші мови програмування, у тому, що стосується представлення знань та управління ними. Мови інженерії знань можна віднести або до скелетних систем, або до систем загального призначення. Дослідники ІІ розробили ці мови спеціально для створення експертних систем. Скелетна мова інженерії знань - це просто урізана експертна система, тобто експертна система, з якої видалено її предметно-орієнтовані знання (оболонка), залишивши лише механізм виведення та допоміжні засоби. Мова інженерії знань загального призначення може обробляти безліч різних проблемних областей, таких як вилучення знань, створення логічних висновків або створення інтерфейсу користувача, хоча його використання досить стомлююче. Ці мови відрізняються універсальністю та гнучкістю [15].

Розробники PROSPECTOR позбавили його знань про геологію, щоб перетворити його на KAS, скелетну систему для діагностики та класифікації. Так само система MYCIN для діагностики та лікування бактеріальних інфекцій стала скелетною системою EMYCIN, а консультаційна система CASNET по глаукомі стала скелетною системою EXPERT. Каркасні системи забезпечують структуру та вбудовані засоби, які спрощують та прискорюють розробку системи. Але їм не вистачає спільності та гнучкості; вони застосовні лише до обмеженого класу завдань та значно скорочують можливості проектування експертних систем [15].

EMYCIN

Ця каркасна мова інженерії знань по суті є MYCIN з віддаленими знаннями предметної області. EMYCIN використовує схему представлення знань на основі правил із жорстким механізмом управління зі зворотним ланцюжком, який обмежує його застосування проблемами діагностики та класифікації. Тим не менш, система надає складні засоби пояснення та отримання знань, що явно прискорюють розробку експертної системи. EMYCIN корисна інженерна мова для діагностичних властивостей, тому він використовувався для створення експертних систем у медицині, геології, техніці, сільському господарстві та інших галузях [15].

EXPERT

Ця скелетна мова інженерії знань використовує схему представлення знань, засновану на правилах і має обмежений механізм керування прямим ланцюжком, що робить його придатним для задач діагностики та класифікації. EXPERT має вбудовані засоби пояснення, збирання знань та перевірки несуперечності для прискорення розробки системи. Модуль перевірки несуперечності працює, зберігаючи базу даних репрезентативних прикладів з відомими висновками та використовуючи її для тестування експертної системи після того, як інженер за знаннями додасть нових правил [15].

Якщо випадок не призводить до правильного висновку, EXPERT відображає обґрунтування для цього випадку, щоб інженер знань міг зрозуміти, як нові правила привели до несподіваних результатів. EXPERT використовувався для побудови діагностичних програм у медицині, геології та інших галузях [15].

OPS5

Ця мова інженерії знань загального призначення використовує схему представлення на основі правил, що працює через прямий ланцюжок. Універсальність системи підтримує різноманітні структури представлення даних та управління в рамках однієї програми. OPS5 має потужний засіб зіставлення зі зразком та ефективний інтерпретатор для зіставлення правил із даними, але не має складного середовища підтримки. Вона не має вбудованих механізмів пояснення чи отримання, та має лише мінімальні засоби для редагування та налагодження програм. OPS5 є останньою серед подібних мов, заснованих на правилах, які виникли в результаті роботи в Університеті Карнегі-Меллона з розробки мов програмування для моделювання людського пізнання та пам'яті. OPS5 та ранні мови серії OPS використовувалися для багатьох додатків когнітивної психології, штучного інтелекту та експертних систем [15].

### 1.2.3 Допоміжні засоби для побудови систем

Допоміжні засоби для побудови системи складаються з програм, які допомагають набувати та представляти знання експерта в предметній галузі, та програм, які допомагають проектувати експертну систему, що будується. Ці програми вирішують складні завдання. Багато з них є дослідницькими інструментами, які тільки починають перетворюватися на практичні та корисні допоміжні засоби, хоча деякі з них пропонуються як повноцінні комерційні системи. Порівняно з мовами програмування та мовами інженерних знань було розроблено відносно небагато допоміжних засобів для побудови систем. Ті, що існують, поділяються на дві основні категорії: засоби проектування та засоби придбання знань [15].

Система AGE служить прикладом засобів проектування, а TEIRSIAS, MOLE і SALT є прикладом засобів отримання знань, побудови системи TIMM і засобів уточнення знань SEEK [15].

AGE

Цей програмний інструмент допомагає інженеру зі знань проектувати та створювати експертну систему. AGE надає користувачеві набір компонентів, які, подібно до будівельних блоків, можуть бути зібрані для формування частин експертної системи. Кожен компонент, що є набором функцій INTERLISP, підтримує каркас експертної системи, такий як прямий ланцюжок, зворотний ланцюжок або архітектура Blackboard. Термін Blackboard відноситься до центральної бази даних, що користуються системами для координації та контролю роботи незалежних груп та правил, які називають джерелами знань [15].

Джерела знань спілкуються, записуючи повідомлення на дошці та читаючи повідомлення з інших джерел знань. Ця архітектура була вперше використана в HEARSAY-II, системі розуміння мови, розробленої в середині 1970-х років, і привела до HEARSAY-III, мови інженерії знань для управління кількома джерелами знань. Системи Blackboard є основою сучасної архітектури інтерфейсу користувача. Інженери знань використовували AGE для розробки та створення HANNIBAL, експертної системи, яка виконує оцінку ситуації, інтерпретуючи дані радіозв'язку супротивника. Система використовує інформацію про місцезнаходження та сигнальні характеристики даних для ідентифікації організаційних підрозділів противника та їх зв'язку, бойового порядку [15].

MOLE

Це система набуття знань для евристичного завдання класифікації, такої як діагностика захворювань. Зокрема, він використовується у поєднанні з методом покриття та диференціації завдань. Експертна система MOLE приймає вхідні дані, пропонує набір можливих пояснень або класифікацій, які охоплюють дані, а потім використовує знання, що диференціюють, щоб визначити, яке з них краще. Процес ітеративний, тому що пояснення самі мають бути обґрунтовані доти, доки не будуть встановлені кінцеві причини. MOLE взаємодіє з експертом у предметній області для створення бази знань, яку система, звана MOLE-p, використовує на вирішення проблем [15].

Придбання знань відбувається у кілька етапів:

1. Побудова вихідної бази знань. MOLE просить експерта перерахувати загальні симптоми чи скарги, які можуть вимагати діагностики. Для кожного симптому MOLE запитує перелік можливих пояснень. Потім MOLE в інтерактивному режимі шукає пояснення вищого рівня, доки знайде набір кінцевих причин. У ході цього процесу MOLE створює мережу впливу, аналогічну до мережі переконань. Щоразу, коли подія має кілька пояснень, MOLE намагається визначити умови, за яких одне з пояснень є правильним. Експерт надає знання про те, що гіпотетична подія може бути причиною певного симптому. MOLE намагається вивести попереднє знання, яке говорить, що якщо гіпотетична подія не відбудеться, та симптом обов'язково з'явиться. Це знання дозволяє системі виключити певну гіпотезу на підставі, що певні системи відсутні [15].

2. Уточнення основи знань. Тепер MOLE намагається виявити слабкі місця у базі знань. Один із підходів полягає в тому, щоб знайти дірки та запропонувати експерту їх заповнити. Як правило, важко дізнатися, чи є база знань повною, тому натомість MOLE дозволяє експертам спостерігати за тим, як MOLE-p вирішує приклади завдань. Щоразу, коли MOLE-p ставить невірний діагноз, експерт додає нові знання. Є кілька способів, якими MOLE-p може дійти неправильних висновків. Вона може помилково відкинути гіпотезу, оскільки вважає, що гіпотеза необхідна пояснення будь-якого симптому. Вона може висувати гіпотезу, тому що вона необхідна для пояснення якоїсь незрозумілої гіпотези. Або їй може бракувати диференційного знання для вибору між альтернативними гіпотезами. Наприклад, у лікаря є пацієнт із симптомами A та B. Далі, припустимо, що симптом A може бути викликаний подіями X та Y, а симптом B може бути викликаний Y та Z. MOLE-p може укласти Y, оскільки він пояснює як A , так і B. Якщо експерт вказує, що це рішення було неправильним, то MOLE запитає, які докази слід використовувати, щоб віддати перевагу X та/або Z Y. MOLE використовувався для створення систем, що діагностують проблеми з автомобільними двигунами, проблеми на сталеварних заводах та неефективність вугільних електростанцій. Щоб MOLE була застосовна, кодування знань з погляду охоплення та диференціації має бути виконане заздалегідь [15].

TEIRESIAS

TEIRESIAS був розроблений Девісом у середині 1970-х років у Стенфордському університеті (США) як інструмент для дослідження нових ідей у ​​галузі збору знань та обслуговування баз даних, а не як інструмент для створення експертних систем. TEIRESIAS отримує знання у інтерактивному режимі від експерта. Якщо MYCIN поставив невірний діагноз, то TEIRESIAS проведе експерта з ланцюжка невірних міркувань доти, доки експерт не вкаже, з чого почалися неправильні міркування. Повертаючись до ланцюжка міркувань, TEIRESIAS також взаємодіятиме з експертом, щоб змінити невірні правила або отримати нові правила. Інформація про нові правила не відразу заноситься до MYCIN. Натомість TEIRSIAS перевіряє, чи сумісне нове правило з аналогічними правилами [15].

Наприклад, якщо нові правила описують, як інфекція потрапляє в організм, а в інших прийнятих правилах є умовний елемент, що вказує на портал входу в організм, то нове правило також повинно мати умовний елемент. Якщо в новому правилі не вказано вхідний портал, TEIRESIAS запросить у користувача цю невідповідність. TEIRESIAS має шаблон моделі правил, що складається зі схожих правил, які він знає, та намагається вписати нове правило у свою модель правил. Іншими словами, модель правила - це знання, яке ТІРЕСІАС має про свої правила. TEIRESIAS служив інтерфейсом для експертної системи MYCIN. Програма — це інформація, необхідна для продовження міркувань. Лікар хоче знати, чому програмі потрібна інформація, а також запитує, як програма дійшла висновку, якого, як вона стверджує, прийшла [15].

### 1.2.4 Середовище підтримки інструментів

Це просто програмні пакети, які постачаються з кожним інструментом, щоб зробити його зручнішим для користувача та ефективнішим. Середовища, які можна поділити на: ті, які потрібні для розробки програм експертних систем, такі як засоби налагодження та редактори баз знань, ті, які необхідні для розширення можливостей розроблених програм, такі як засоби введення/виводу та механізм пояснення. Хоча кілька експертних систем підтримують ці інструменти, всі вони підтримують деякі з них. Ці кошти зазвичай доступні як частина інженерної мови та призначені для роботи спеціально з цією мовою[15] .

Допоміжні засоби налагодження: більшість мов програмування містять засоби трасування та розривні пакети. Трасування надає користувачеві відображення роботи системи шляхом перерахування імен всіх запущених правил або відображення імен усіх підпрограм. Пакет break дозволяє користувачеві заздалегідь повідомити програму, де зупинитися, щоб користувач міг зупинити виконання програми безпосередньо перед деякою помилкою, що повторюється, і перевірити поточні значення в базі даних. Усі інструменти експертної системи повинні мати ці функції [15].

Декілька експертних системних інструментів включають автоматичне тестування, дещо незвичніший засіб налагодження, ніж засоби трасування або зламування пакетів. Ця допомога дозволяє користувачеві автоматично тестувати програму на великій кількості контрольних завдань, щоб виявити помилки чи невідповідності у рішеннях [15].

Редактори бази знань: більшість інструментів експертних систем надають механізм редагування бази знань. У найпростішому випадку це звичайний текстовий редактор для зміни правил та даних вручну. Але багато інструментів включають у своє середовище підтримки інші засоби. Наприклад, EMYCIN використовує автоматичний облік. Редактор EMYCIN відстежує зміни, внесені користувачем, та записує відповідну інформацію про подію. Якщо користувач додає або змінює правило, редактор автоматично зберігає дату зміни та ім'я користувача разом із правилом для подальшого використання [15].

Інженери знань зазначають це особливо корисним, коли кілька різних експертів модифікують або уточнюють базу знань [15].

Ще одним поширеним засобом у редакторах баз знань є перевірка синтаксису, коли редактор використовує знання про граматичну структуру мови експертної системи, щоб допомогти користувачам запровадити правила з правильним написанням та форматом. Коли користувач вводить неписьменне правило або команду, редактор вловлює це і пояснює, що не так. Виправлення подібних помилок під час редагування, а не під час тестування системи може значно скоротити час розробки [15].

Надзвичайно корисним, але зазвичай недоступним засобом для редакторів баз знань є перевірка узгодженості, коли система перевіряє семантику або значення правил та даних, щоб побачити, чи не суперечать вони існуючим знанням у системі. У разі конфлікту редактор допомагає користувачеві вирішити конфлікт, пояснюючи, що його викликало, і описуючи способи його усунення [15].

## 1.3 Алгоритм пошуку по дереву в прийнятті рішень

Відомо, що експертні системи моделюють процеси мислення людей експертів під час розв'язання практичних задач предметної області. Слід зазначити, що людина, зазвичай, не розв'язує задачу у тому вигляді, в якому вона від початку сформульована, а представляє її таким чином, щоб її було зручно розв'язувати. Таке переформулювання умов практичних задач називають формалізацією задач. До способів формалізації задач висувають дві основні вимоги [16]:

1) подання задачі має достатньо точно моделювати предметну область експертної системи;

2) подання має давати змогу реалізовувати просте та ефективне розв'язання практичних задач.

Коло задач, які розв'язують за допомогою експертних систем, є дуже широким. Відповідно до їх специфіки використовують різні способи подання, серед яких найбільшого поширення набули наступні:

1. подання у просторі станів;
2. подання у просторі задач;
3. подання у вигляді доведення теорем.

Подання задач у просторі станів містить такі елементи:

1. початковий стан системи;
2. цільовий стан системи;
3. сукупність операторів, за допомогою яких реалізують переходи

між станами системи.

Розв'язання задачі тут полягає у визначенні послідовності переходів між початковим та цільовим станами [16].

У поданні задач у просторі станів широкого поширення набув математичний апарат теорії графів. Тут кожному можливому стану предметної області відповідає певна вершина, а можливим переходам між станами – орієнтовані дуги графу. Крім того, для розв'язання деяких практичних задач кожній дузі графу ставлять у відповідність певний ваговий коефіцієнт, який характеризує, наприклад, витрати ресурсів на реалізацію переходів між станами предметної області. Таким чином, розв'язання задачі у просторі станів зводиться до пошуку в орієнтованому зваженому графі, який моделює стани предметної області [16].

Такий граф називають деревом пошуку. Процес розв'язання задачі полягає у формуванні дерева пошуку, вершина (корінь) якого відповідає початковому стану предметної області. Інші вершини дерева пошуку відповідають всім можливим станам предметної області, впорядкованим за ярусами відповідно до кількості переходів із початкового стану. Ребрам дерева пошуку відповідають можливі переходи між станами предметної області [16].

Подання у просторі задач полягає у багатократному розбитті задач на дрібніші, доки вихідна задача не буде зведена до сукупності елементарних, які мають тривіальній розв'язок, або доки не буде доведено, що задача не має розв'язку. Процес розбиття (декомпозиції) задачі називають редукцією. Очевидно, що спосіб зведення задач до підзадач ефективний у розв'язанні складних комплексних задач, декомпозиція яких породжує взаємно незалежні підзадачі, наприклад, пов'язані із проектуванням електричних станцій, підстанцій, електричних мереж [16].

Для математичного моделювання зведення задач до підзадач широкого застосування також набув апарат теорії графів. Вершинами такого графу виступають вихідна задача та підзадачі, отримані в результаті редукції. Ребрам графу відповідають оператори декомпозиції, які породжують нові підзадачі. Отриманий граф називають графом редукції [16].

У графі редукції розрізняють два види вершин – кон'юнктивні та диз'юнктивні. Кон’юнктивні вершини означають, що для розв'язання задачі необхідно розв'язати всі дочірні підзадачі. Такі вершини часто називають вершинами типу «І». Диз'юнктивні вершини, у свою чергу, позначають розбиття на альтернативні підзадачі. Для розв'язання таких задач достатньо розв'язати будь-яку з дочірніх підзадач. Диз'юнктивні вершини часто називають вершинами типу «АБО» [16].

Подання задач у вигляді доведення теорем полягає у послідовному багатократному переформулюванню умов задачі доки не буде доведена (або спростована) істинність певних положень. Процес розв'язання задачі тут полягає у послідовному виконанні наступних процедур:

1. формують множину вихідних істинних тверджень (аксіом);
2. формують гіпотезу відносно результату розв'язку задачі;
3. аксіоми комбінують між собою за допомогою правил виведення та отримують нові істинні твердження. Якщо під час виконання процедури нові твердження не з'явилися, це свідчить про неможливість розв'язання задачі;
4. перевіряють, чи є серед нових тверджень цільова гіпотеза, або її спростування. Якщо існує, то теорему вважають доведеною та розв'язання задачі закінчують. В інакшому випадку управління передають процедурі алгоритму для формування нових істинних тверджень.

Очевидно, що розглянутий спосіб подання задач є окремим випадком представлення у просторі станів із логічною формалізацією ходу рішення. Дійсно, всі можливі сукупності істинних тверджень, які виникають в процесі розв'язання задачі, визначають можливі стани предметної області, тобто вершини графу простору станів, а правила виведення, за допомогою яких аксіоми поєднуються визначають можливі переходи між станами, тобто ребра графу [16].

Розв'язання задач пошуку у просторі станів полягає у визначенні  
сукупності переходів між вихідним та цільовим станами предметної  
області. Формування та реалізація відповідних алгоритмів базується на  
двох основних положеннях [16].

Граф пошуку задається у пам'яті експертної системи у неявному  
вигляді сукупності можливих станів предметної області та переходів між  
ними. В результаті дерево пошуку будується послідовно, починаючи з  
вихідного стану системи і проглядаючи всі можливі переходи з цього  
стану [16].

Кожний цикл пошуку у просторі станів полягає у виборі одного  
переходу серед множини альтернатив. В теорії систем штучного інтелекту множину можливих альтернативних переходів називають конфліктною множиною. Саме ефективний вибір наступного переходу з конфліктної множини характеризує інтелектуальність експертної системи [16].

Розв'язання задач пошуку у просторі станів часто пов'язано із проблемою комбінаторного вибуху. Комбінаторний вибух полягає у різкому зростанні розмірності конфліктної множини із просуванням по графу станів. Це пов'язано з тим, що кожний цикл пошуку пов'язаний із можливістю реалізації лише одного переходу, який часто породжує багато  
нових альтернатив [16].

Методи неінформативного («сліпого») пошуку передбачають реалізацію повного перебору всіх можливих станів у просторі предметної області, не враховуючи додаткової оціночної інформації щодо ефективності кожного можливого переходу. До цієї групи методів відносять:

• Випадковий пошук;

• Пошук у глибину;

• Пошук у ширину;

• Пошук за алгоритмом Дейкстри.

Випадковий пошук

Стратегія даного методу полягає в тому, що оператора червоного переходу обирають випадково. Пошук виконують доки не буде досягнуто цільового результату [16].

Із плюсів даного методу можна виділити простоту реалізації. Але явним мінусом є те, що випадковий вибір переходу не гарантує збіжність алгоритму. Цей метод можна використовувати для рішення простих задач з малою розмірністю простору станів. Для більш складних задач випадковий пошук займає надзвичайно багато часу, що різко зменшує ефективність [16].

Пошук у глибину

Стратегія даного методу полягає у послідовному розгляданні одного ланцюга переходів між станами предметної області. Якщо під час такого розглядання будуть виявлені тупикові стани або замкнені цикли, то система пошуку відкочується до останньої нереалізованої альтернативи, та починає розглядати цей ланцюг. Пошук продовжується доки не буде досягнуто цільового результату [16].

Пошук у ширину

Стратегія пошуку у ширину полягає у послідовному розгляданні сукупності переходів із початкового стану довжиною в один перехід, два переходи і т.д., доки не буде досягнуто цільового стану системи. При цьому проводиться фільтрація замкнених циклів та тупикових станів у графі простору станів [16].

Пошук за алгоритмом Дейкстри

Особливістю алгоритму Дейкстри є поняття ваги переходу. Пошук шляху у графі станів за цим алгоритмом починають з присвоювання початковій вершині ціни, яка дорівнює нулю, і розкриваючи послідовно всі вершини з найменшими значеннями ціни пошук доходить до цільового стану системи. Такий метод забезпечує мінімізацію втрати ресурсів [16].

Реалізація методів евристичного (направленого) пошуку базується  
на використанні додаткової інформації, яка дозволяє оцінити очікувані  
витрати ресурсів на реалізацію переходів між поточним та цільовим  
станами предметної області. До евристичних методів пошуку відносять:

• Градієнтний алгоритм;

• Алгоритм глобального врахування послідовності цілі;

• А-алгоритм.

Градієнтний алгоритм

Цей алгоритм являє собою вдосконалений алгоритм пошуку у глибину, де розкриття чергової вершини здійснюється з кінця поточного ланцюга пошуку у напряму мінімального значення оцінювальної функції суміжного стану таким чином, щоб якомога ближче наблизитися до цілі на кожному кроці пошуку [16].

Алгоритм глобального врахування послідовності цілі

Цей алгоритм є вдосконаленою версією алгоритму пошуку у ширину, де розкриття чергової вершини здійснюють виходячи з мінімального значення оцінювальної функції відповідного стану [16].

А-алгоритм

А-алгоритм є результатом синтезу алгоритмів глобального врахування цілі та Дейкстри. Тут для кожної вершини формують значення оцінювальної функції, яка дорівнює сумі витрат ресурсів на досягнення поточного стану починаючи із вихідного та оцінювальної функції витрат ресурсів на реалізацію переходів від поточного стану до цільового. Для переходу наступну вершину обирають виходячи із мінімального значення оцінювальної функції [16].

# 2 СПЕЦІЛЬНА ЧАСТИНА

## 2.1 Опис предметної області

Була проведена розробка експертної системи для того, щоб допомогти людині у виборі породи собаки та звузити коло пошуку. За допомогою селекції люди змогли вивести велику кількість порід, які дуже відрізняються як зовнішніми характеристиками, так і поведінкою. Щоб уникнути придбання собаки, яка не буде повністю відповідати вашим очікуванням, була розроблена ця ЕС.

Переваги цієї системи в тому, що за дуже короткий час можливо дізнатися яка саме порода собаки вам підходить відповівши на декілька запитань.

Всього у системі 11 запитань:

1. Якого розміру собаку ви віддаєте перевагу ?
2. Скільки часу ви готові гуляти із собакою ?
3. Чи є у вас досвід догляду за собакою ?
4. Скільки ви плануєте дресирувати собаку ?
5. Для яких цілей, ви хочете собаку ?
6. На скільки активні прогулянки ви хочете?
7. Яка шерсть повинна бути у собаки?
8. Наскільки гучною повинна бути собака ?
9. Чи є у вас алергія на собак ?
10. Чи повинна собака ладити із дітьми ?
11. Чи повинна собака ладити із іншими тваринами ?

У кожного запитання декілька варіантів відповідей. Коли користувач застосунку по вибору породи собак відповідає, то відповіді записуються у об’єкт, в якому 11 властивостей, які відповідають кожному запитанню. Вхідні дані представлені буквеним шифром, який обробляє програма, кожна буква у відповіді співпадає із відповідною властивістю.

Після чого система проводить порівняння із базою знань, перебираючи масив із список порід собак та їх властивостей. При кожному співпадінні рахується кількість балів для кожної собаки, 1 бал – 1 правильна відповідь. Вихідні дані представлені списком із трьох масивів: «Собаки, що ідеально підходять», «Собаки, що майже підходять», «Собаки, які можливо підійдуть». Якщо відбулося 11 збіжностей, то ця собака ідеально підходить для користувача застосунку. При 9-10 балів собака майже підходить для користувача, з можливістю подивитись, які саме властивості не співпали, тому людина може обрати, чи критичні для неї ці властивості собаки, чи ні. Аналогічний функціонал і при набиранні 7-8 балів для собак, які можливо підійдуть.

## 2.2 Моделювання предметної області

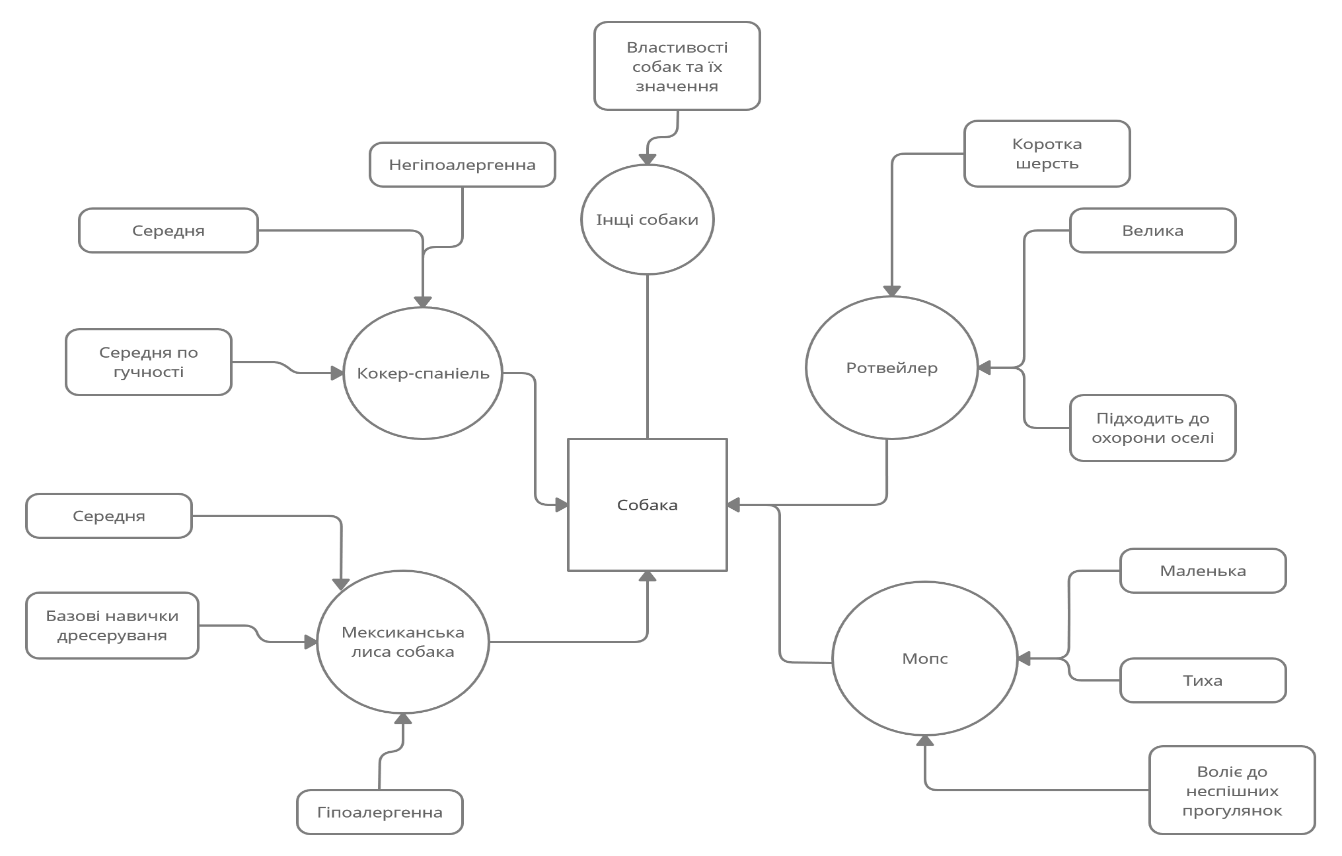
У даному випадку предметна область представлена у вигляді семантичної мережі, бо собаки являються об’єктами одного класу і мають однакові властивості, та різняться лише значенням цих властивостей. Як приклад приблизна модель семантичної мережі експертної системи по вибору собак: 

Рисунок 1. Семантична модель предметної області експертної системи по вибору собак.

## 2.3 Опис програми

Оболонка програми складається із браузерного застосунку, це інтерфейс користувача для зручного використання програми. До інтерфейсу входить 13 сторінок. Головна сторінка, яка пропонує пройти тест на вибір собаки, 11 сторінок із запитаннями та сторінка із результатами. Після проходження остання сторінка відправляє вхідні дані до бази знань.

До складу бази знань входять архів із породами собак та алгоритми роботи із вхідними даними та архівом собак. До архіву собак входить 0 порід собак, масив, у якому записані ці всі породи, та масиви із текстовими властивостями собак, який використовується для того, щоб показати на сторінці результату властивості кожної собаки. Алгоритми роботи експертної системи представлені двома алгоритмами. Перший алгоритм працює із рахуванням балів кожної собаки, та додаванням собаки до відповідного масиву, другий алгоритм висвічує властивості собак та показує, які саме властивості собаки співпали із відповідями користувача, а які ні.

# 3 РОЗРАХУНКОВА ЧАСТИНА

## 3.1 Керівництво користувача

Для початку вибору підходящої породи собаки натисніть кнопку «Розпочати». Після цього ви потрапляєте на сторінку із першим запитанням. Вам запропоновано запитання та варіанти відповідей. Також ви можете побачити під варіантами відповідей 2 кнопки, «Головна» та «Назад». При натисканні кнопки «Головна» ви повертаєтесь на головну сторінку, при натисканні кнопки «Назад», ви повертаєтесь до попереднього запитання, або на головну, в залежності від номеру сторінки. Щоб з’явилась можливість пройти вперед вам необхідно обрати одну із запропонованих відповідей, при натисканні яких з’являється кнопка «Вперед», яка перенесе вас на наступне запитання. Вам буде запропоновано 11 питань, після відповіді на які ви потрапите на сторінку із результатами тесту. На сторінці буде показано три варіанти результатів: «Ідеальні собаки» - собаки які повністю підходять по всім параметрам, які ви обрали, «Майже ідеальні собаки» - собаки які підходять вам по 9-10 параметрам, також у додатковій інформації, можна побачити які саме параметри вам не підійшли, «Собаки, які можливо вам підійдуть» це собаки, які підходять вам по 7-8 параметрах. Тепер у вас є можливість обрати ту собаку, яка вам більше всього підходить із списку запропонованих собак.

3.2 Вимоги для технічного забезпечення.

Для роботи програми потрібно мати

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Turban, E., & Aronson, J. E. (2001). Decision support systems and intelligent systems, sixth Edition (6th ed). Hong Kong: Prentice International Hall.
2. Shu-Hsien Liao. (2005). Expert system methodologies and applications—a decade review from 1995 to 2004. Expert Systems with Applications 28 (2005) 93–103.
3. Питер Джексон (2001). Введение в экспертные системы = Introduction to Expert Systems. — 3-е изд. — М.: Вильямс — С. 624.
4. Страхова О.П. (2014). Экспертные системы в фармации: Методическое пособие для студентов фармацевтического факультета - Запорожье: ЗГМУ, - 36с.
5. Forsyth, R.S. (1984). The Architecture of Expert Systems. In: Forsyth, R.S. (ed.) Expert Systems: Principles & Case Studies. London: Chapman & Hall Ltd., pp. 9-17.
6. Kantureeva M. Zakirova A. (2014) The Methodology of Expert Systems IJCSNS International Journal of Computer Science and Network Security, VOL.14 No.2,
7. Chastikov A.P. Gavrilova T.A. Belov D. L. Development of expert systems. CLIPS environment. – SPb. : BHVPetersburg, 2003. – 608 p.
8. Informatics: The textbook / Under the editorship of the prof. of N. V. Makarova – M: Finance and statistics, 1997.
9. Artificial intelligence: Application in the integrated production systems / Under the editorship of E.Kyyusiak; The Lane with English – M: Mechanical engineering, 1991
10. Artificial intelligence: in 3 books of Book 1. Systems of communication and expert systems. The directory / Under the editorship of E.V.Popova – M: Radio and communication, 1990
11. Page Osuga. Processing of knowledge: The lane with M: World, 1989
12. Priests E.V.Ekspertnye systems: The solution of unformalized tasks in dialogue with the COMPUTER. – M: Science. Hl. edition physical. 1987
13. Advantages and disadvantages of Expert Systems. ilearnlot. URL: <https://www.ilearnlot.com/expert-system-advantages-disadvantages/34332/>
14. Вавіленкова A. АНАЛІЗ МОДЕЛЕЙ ПРЕДСТАВЛЕННЯ ЗНАНЬ В ЕКСПЕРТНИХ СИСТЕМАХ УПРАВЛІННЯ. *Проблеми інформатизаиії та управління.* 2007. С. 22.
15. Subarna D. Construction of an Expert System: 4 Tools | Artificial Intelligence. URL:<https://www.engineeringenotes.com/artificial-intelligence-2/expert-systems/construction-of-an-expert-system-4-tools-artificial-intelligence/35582>.
16. Кацадзе Т. Л. К30 Експертні системи прийняття рішень в енергетиці: навч. посіб. / Т. Л. Кацадзе. – К.: ЛОГОС, 2014. – 173 с. – Бібліогр.: с. 167-173. ISBN 978–966–171–768–7