В. І. Месюра, А. А. Яровий, І. Р. Арсенюк

ЕКСПЕРТНІ СИСТЕМИ

Частина 1

Міністерство освіти і науки України
Вінницький національний технічний університет

В. І. Месюра, А. А. Яровий, І. Р. Арсенюк

ЕКСПЕРТНІ СИСТЕМИ

Частина 1

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів спеціальності “Інтелектуальні системи прийняття рішень”. Протокол №6 від 29 грудня 2005 р.

Вінниця ВНТУ 2006

УДК 330:004.891
М 53

Р е ц е н з е н т и:

В. П. Кожем'яко, доктор технічних наук, професор
О. В. Осадчук, доктор технічних наук, професор
Л. І. Тимченко, доктор технічних наук, професор
Г. Б. Ракитянська, кандидат технічних наук, доцент

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Месюра В. І., Яровий А. А., Арсенюк І. Р.
М53 **Експертні системи. Частина 1. Навчальний посібник. – Вінниця: ВНТУ, 2006.– 114 с.**

В першій частині посібника розглядаються основні поняття та перспективи розвитку прикладних систем штучного інтелекту в контексті їх використання в технологіях експертних систем. Проаналізовано методи подання й обробки знань, особливості побудови експертних систем, загальні відомості про методи набуття знань та інструментарії для розробки експертних систем. Здійснено теоретичний аналіз основних принципів та методів подання знань та їх моделей в експертних системах.

Посібник розроблений у відповідності з планом кафедри інтелектуальних систем та програмою дисципліни "Експертні системи" та призначений для студентів, що вивчають експертні системи і споріднені їм дисципліни, а також для всіх бажаючих ознайомитися з основами побудови прикладних систем, основаних на поданні й обробці знань.

УДК 330:004.891

ЗМІСТ

ВСТУП

Обчислювальна техніка на даному етапі завоювала ключові позиції в багатьох сферах людської діяльності. Крім класичних застосувань, пов'язаних з виконанням інженерних і економічних розрахунків, розробкою автоматизованих систем керування, створенням інформаційно-пошукових систем і т.д., зараз успішно розвивається науковий напрямок, що має назву «штучний інтелект». Глобальною метою вказаного наукового напрямку є створення інтелектуальних автоматизованих систем, що виконують ті ж функції, що й творча особистість, в крайньому разі, їх найпростіші прояви.

За останні роки із загального напрямку «штучний інтелект» виділився новий науковий напрямок, пов'язаний зі створенням експертних систем, призначених для вирішення задач експертного оцінювання ситуацій у різних предметних областях. При побудові експертних систем основними є питання: „Які знання повинні бути в них подані та у якій формі?” Структура знань залежить від сфери їх використання і може мати досить складний характер. Така структура містить у собі різні факти із предметної області, взаємозв'язків між ними, правил дій і т.д. Вона також повинна містити в собі знання, що стосуються способу включення знань до експертної системи. Складність і різноманітність структур знань спонукали до розробки різноманітних способів подання знань, з яких варто виділити якнайбільш поширені: логічні моделі, продукційні та фреймові системи, семантичні мережі, модель дошки оголошень. Кожний спосіб подання має свої переваги та недоліки й тяжіє до певної структури знань. В останні роки все більше почали використовуватися моделі подання знань, що поєднують зазначені способи.

Особливість систем подання знань полягає в тому, що вони моделюють діяльність людини, яка часто здійснюється у неформальному вигляді. Якщо методи вирішення обчислювальних задач основуються на чітких алгоритмах, обґрунтованість яких базується на понятті збіжності, то моделі подання знань мають справу з інформацією, що одержується від експертів, яка часто має якісний, і до того ж, суперечливий характер. Проте, у силу специфіки функціонування комп’ютерних систем, подібна інформація повинна бути приведена до однозначного нормалізованого вигляду. Це здійснюється на основі ідей багатозначної логіки, теорії нечітких множин і аналогічних математичних моделей.

У той же час моделі подання знань є предметом досліджень і розробок у середовищі вузькоспеціалізованих фахівців, в основному програмістів і математиків, у той час як потреба в таких моделях відчувається практично у всіх предметних областях. Це пояснюється необхідністю побудови експертних систем, структура яких нерозривно пов'язана з формами подання знань, обумовлених у свою чергу особливостями предметної області.

У першій частині посібника систематично викладені основні положення теорії експертних систем у контексті теорії штучного інтелекту, сфери компетентності використання експертних систем, методологія їх проектування, а також теоретично обґрунтоване глобальне поняття «знань», наведено моделі подання знань. Велика увага приділяється порівняльному аналізу моделей, аналізу їх „сильних” і „слабких” сторін, огляду галузей застосування. Важливу роль у книзі відіграють приклади, що пояснюють структури моделей, правила виведення й т.д. Все це робить книгу корисною не лише для фахівців, але й для осіб, що не є фахівцями в області експертних систем, але зацікавлених у їх використанні. Посібник деякою мірою відбиває сучасний стан досліджень в області експертних систем, а також точку зору наукової громадськості на методи подання й використання знань, принципи побудови експертних систем.

1 ОСНОВИ ТЕОРІЇ ЕКСПЕРТНИХ СИСТЕМ. ВЗАЄМОЗВ'ЯЗОК ЕКСПЕРТНИХ СИСТЕМ ТА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Проблематика штучного інтелекту. Напрямки розвитку систем штучного інтелекту

Класичне тлумачення поняття інтелекту було запропоноване вченим А.Тюрінгом: проблема формулюється в термінах імітаційної гри, основаної на популярній грі "Англійська вітальня". Людина А і інтелектуальна машина В розміщуються в різних кімнатах. Екзаменатор С знаходиться в третій кімнаті і, ставлячи запитання, повинен визначити, хто йому відповідає: людина чи машина. Якщо він не може розпізнати, хто є хто, це означає, що машина інтелектуальна. Звичайно, екзаменатор повинен розробити певну стратегію діалогу, яка дозволила б йому оцінити, чи здатний випробовуваний об'єкт пізнавати і аналізувати, чи тільки повторювати фрази, що запам'ятались.

Розвиток штучного інтелекту (ШІ) як наукового напряму став можливим тільки після створення ЕОМ. Це відбулося в 40-х роках XX століття. В цей же час вчений Н.Вінер створив свої основоположні роботи з нової науки – кібернетики [2,3,5].

Термін ШІ (artificial intelligence) був запропонований в 1956 році на семінарі з аналогічною назвою в Стенфордському університеті. Семінар був присвячений розробці логічних, але не обчислювальних задач. Незабаром після визнання ШІ самостійною галуззю науки, відбулося розділення на два основні напрями: нейрокібернетику і кібернетику "чорного ящика".

Основна ідея нейрокібернетики: єдиний об'єкт, здатний мислити, - це людський мозок. Тому будь-який пристрій, що здатен мислити, повинен певним чином відтворювати його структуру. Таким чином нейрокібернетика орієнтована на апаратне моделювання структур, подібних структурі мозку. Звичайні комп'ютери для реалізації цього принципу не підходять, зважаючи на обмежені ресурси, тому цей напрям орієнтований на створення трансп'ютерів – паралельних комп'ютерів з великою кількістю процесорів і нейрокомп'ютерів, що моделюють структуру нервових структур мозку людини.

У основі кібернетики "чорного ящика" лежить принцип, протилежний нейрокібернетиці. Не має значення, яка структура "мислячого" пристрою. Головне, щоб на задані вхідні дії він реагував так, як людський мозок. Цей напрям орієнтований на пошуки алгоритмів вирішення інтелектуальних задач на існуючих моделях комп'ютерів.

На теперішній час перелік досліджень та дисциплін із штучного інтелекту постійно збільшується [1,6]. Деякі з них, які безпосередньо

певним чином мають відношення та впливають на розвиток експертних систем, зображені на рис. 1.1.

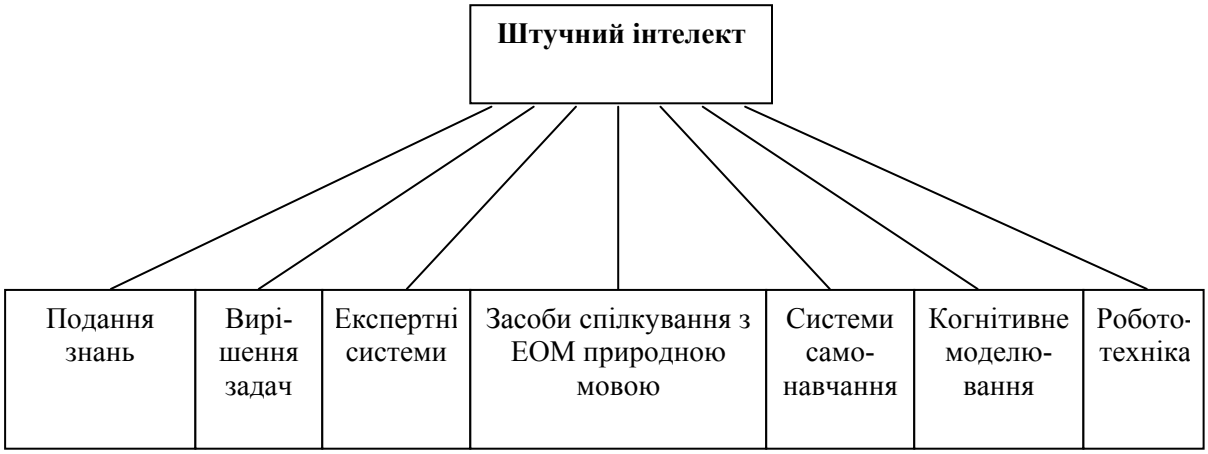

Рисунок 1.1 - Напрямки розвитку систем штучного інтелекту

Коротко розглянемо кожну із наведених дисциплін.

Подання знань

Подання знань є найважливішою областю досліджень з штучного інтелекту. Це основа решти дисциплін. Знання мають форму описів об'єктів, взаємозв'язків і процедур. Наявність адекватних знань і здатність їх ефективно використовувати означає "уміння". Мозок людини добре пристосований до образної обробки інформації, але при виконанні обчислень стає безпорадним навіть в порівнянні з невеликим калькулятором.

Чи можуть комп'ютерні системи (КС) відтворити образну обробку, яка здійснюється людським мозком і якщо так, то яким чином? Створення загальної теорії або методу подання знань є стратегічною проблемою. Така теорія здатна відкрити можливість накопичення знань, які потрібні щодня для вирішення все нових і нових задач.

Проте для досягнення поставленої мети необхідно, перш за все, знайти спосіб вираження загальних закономірностей нашого світу, в цьому і полягає суть проблеми подання знань.

Вирішення задач

В загальному поданні, вирішення задач зводиться до пошуку шляху з деякої початкової точки в цільову або кінцеву. Людина робить це досить ефективно за допомогою міркувань (дедуктивного логічного висновку), процедурного аналізу, аналогії і індукції. Комп'ютерні системи, принаймні в даний час, вирішують задачі тільки з використанням дедуктивного логічного висновку і процедурного аналізу. Характерним є те, що тип задачі визначає метод, найбільш відповідний для її вирішення.

Отже, задачі, які зводяться до процедурного аналізу, краще за все розв'язуються на комп'ютері. Облікові задачі, ведення рахунків, аналіз надходження готівки можуть служити прикладами процедурних задач, вирішуваних комп'ютером швидше і надійніше, ніж людиною.

Задачі, пов'язані з використанням аналогії або індукції, ефективніше розв'язуються людиною.

Задачі, що вимагають дедуктивних міркувань, ефективно розв'язуються за допомогою експертних систем (систем, основаних на знаннях), які і є предметом вивчення. Експертні системи є класом комп'ютерних програм, які дають поради, проводять аналіз, виконують класифікацію, дають консультації і ставлять діагноз. Вони орієнтовані на вирішення задач, що звичайно вимагають проведення експертизи людиною-фахівцем. На відміну від класичних комп’ютерних програм, що використовують процедурний аналіз, експертні системи вирішують задачі у вузькій конкретній області експертизи (предметній області) на основі дедуктивних міркувань.

Стратегії вирішення задач можуть бути досліджені шляхом розвитку методології ігрових ситуацій, які вимагають залучення творчих здібностей, виробляють уміння у гравця або відтворюють деякі аспекти вирішення проблеми людиною. При розробці гри повинні бути обов'язково розглянуті такі фактори:

Припущення. Основна парадигма гри повинна включати ряд ключових рішень:

1. Чи повинен бути в грі один переможець (як в шахах) чи можуть виграти всі гравці?

2. Яка кількість гравців?

3. Чи є гра детермінованою (не потрібно кидати жереб або відсутній генератор випадкових чисел), частково детермінованою або недетермінованою?

4. Чи є вся попередня інформація доступна (як в шахах) чи частина інформації невідома (як в картах)?

*Подання***.** У якому вигляді гра буде подана? У випадку шахів, наприклад, це означає, що необхідно визначити вигляд фігур, їх розташування на дошці і значення. Закони гри можуть бути визначені її розробником.

*Мета***.** Що вважати метою гри? У шахах, наприклад, метою є поставити мат супротивнику. Повинні бути також передбачені засоби, які дозволяли б розпізнати, досягнуто чи ні цільового стану. Мета може бути видима гравцям або прихована (як в пригодницьких іграх фірми „INFOCOM”).

Правила гри. Які з допустимих в грі ходів законні? Правила можуть бути відомі (як в шахах) або невідомі (як в пригодницьких іграх).

Стратегія управління. Якщо гра реалізована на комп'ютері, то повинні бути в наявності засоби для обдумування наступного (якнайкращого) ходу.

Засоби спілкування з ЕОМ природною мовою

Комп'ютерні системи (КС) мають свою власну мову для подання знань і вирішення задач.

Мова - це набір символів, що використовуються для подання знань (семантика) і правил, призначених для обробки символів (синтаксис) і вирішення задач. Людина працює найефективніше, якщо вона володіє спеціальними мовами, які розвиваються до рівня потреб конкретної предметної області (ПО).

Люди часто знаходять своє покликання в тому, щоб навчитися формулювати задачі на мові комп'ютера (тобто в зрозумілих комп'ютеру термінах). Точно так же комп'ютер видає одержане рішення своєю мовою, якщо тільки воно попередньо не переведене назад на мову користувача.

Якщо правила перекладу виражені у вигляді сукупності знань (символів, процедур), то логічно припустити, що можуть бути розроблені засоби, що дозволяють комп'ютеру розуміти постановку задачі природною мовою, а потім природною мовою видавати її вирішення. Це основна тема досліджень з розробки засобів спілкування з ЕОМ природною мовою.

Тобто, стратегічною метою досліджень з розробки засобів спілкування з ЕОМ природною мовою є встановлення принципів взаємодії між людьми і на їх основі створення комп'ютерних систем, з якими можна було б спілкуватися як з людьми. У цьому контексті можна виділити такі ключові проблеми:

1. Машинний переклад – використання КС для перекладу текстів з однієї мови на іншу; в даний час в цій області використовується складна модель, яка включає аналіз і синтез природно-мовних повідомлень, що складається з декількох блоків. Для аналізу це: морфологічний аналіз (аналіз слів в тексті), синтаксичний аналіз (аналіз речень, граматики і зв'язків між словами), семантичний (аналіз значення кожного речення на основі деякої природно-орієнтованої бази знань), прагматичний аналіз (аналіз значення речень в навколишньому контексті на основі власної бази знань). Синтез включає аналогічні етапи, але дещо в іншому порядку.
2. Інформаційний пошук – забезпечення за допомогою КС доступу до інформації з конкретної тематики, що зберігається у сховищах даних чи надвеликих базах даних (БД). В даний час можливий пошук тільки за ключовими словами. Сучасні КС не достатньо ефективно можуть знайти інформацію за контекстом або аналогією.
3. Генерація документів – застосування КС для перетворення документів, що мають певну форму або задані на спеціальній мові, в еквівалентний документ в іншій формі або іншою мовою.

4. Взаємодія з КС – організація діалогу між непідготовленим користувачем і ПК.

Важливість розробки засобів взаємодії з ЕОМ природною мовою обумовлена необхідністю їх поєднання з експертними системами, орієнтованими на непідготовленого користувача.

Системи самонавчання

На даному етапі розвитку комп'ютерної техніки вже є невід'ємною частиною, навіть нормою, здатність до навчання КС. Більше того, останнім часом задача ставиться таким чином, що здатність до навчання повинна мати кожна прикладна програма. Поява персональних комп'ютерів (ПК) змінила взаємовідносини між користувачем і обчислювальною технікою, а, отже, змінилася роль програміста. Замість того, щоб примушувати користувача долати складнощі програмування, простіше навчити комп'ютер складнощам виконання тієї або іншої задачі, що стоїть перед користувачем.

З часом в такі програмні продукти як електронні таблиці, системи управління базами даних, текстові процесори все більше та масштабніше будуть інтегруватися компоненти, що мають властивості штучного інтелекту. Певні вищевказані тенденції можна чітко спостерігати в сучасних програмних продуктах компаній Microsoft, Abobe, ABBYY Software LTD та ін. Загалом даний напрямок досліджень ШІ активно розвивається, і включає дослідження та розробку моделей, методів і алгоритмів, орієнтованих на автоматичне накопичення знань на основі аналізу і узагальнення даних.

Когнітивне моделювання

Когнітивне моделювання - це галузь науки, метою якої є розробка теорії і моделей людського мислення і його функцій. Когнітивне моделювання допомагає не тільки діагностувати і лікувати психічні захворювання, але і виявляти процеси, що протікають в свідомості людини при вирішенні задач. Це не означає, що кращими комп'ютерами є ті, які моделюють роботу людського мозку. Проте можна зробити висновок про те, якого типу комп'ютери нам потрібні, як спроектувати комп'ютер, який би розширив можливості мислення людини і дозволив би ефективно вирішувати задачі.

Обробка візуальної інформації і робототехніка

З появою перших автоматів (XIV ст.) люди були захоплені ідеєю побудови електричних і механічних пристроїв, які могли б діяти подібно людині. Найвідомішим з перших автоматів була штучна качка Вокансона (1738 рік). Вона могла хлопати крилами, пити воду, клювати зерно, імітувати здійснення природних потреб завдяки майстерно зробленій системі травлення.

На даному етапі розвитку вказаного напрямку ШІ, на підприємствах компанії IBM робототехнічні системи проектують і здійснюють комплектацію комп'ютерів практично без участі людини.

Дослідження у галузі робототехніки входять як складова частина в дослідження щодо ШІ, які ставлять за мету оснастити комп'ютери засобами візуальної обробки і маніпулювання об'єктами в деякому середовищі. В глобальному плані вказані дослідження ведуться за трьома основними напрямками:

1. Розробка елементів сприйняття (сенсорів) і розпізнавання інформації, що надходить від систем сприйняття;
2. Створення маніпуляторів і систем управління ними;
3. Виявлення евристик для вирішення задач переміщення в просторі і маніпулювання об'єктами (планування діяльності).

1.2 Вплив штучного інтелекту на ідеологію програмування

Приведемо одне із визначень ШІ, яке в даному контексті буде найбільш вдалим.

Штучний інтелект (ШІ) – це програмна система, що імітує процес мислення людини за допомогою ЕОМ. Для створення такої системи необхідно вивчити процес мислення людини, що вирішує певні задачі або приймає рішення в конкретній області, виділити основні кроки цього процесу та розробити програмні засоби, що відтворюватимуть вказаний процес на комп'ютері. Таким чином, методи ШІ припускають простий структурний підхід до розробки складних програмних систем прийняття рішень [4,7].

Традиційне програмування

Комп'ютерні програми, як правило, призначені для вирішення чітко визначених задач. Пристосувати програму до вирішення нових задач можна, але для цього необхідно внести в неї зміни, а для цього її всю потрібно уважно переглянути. Але докладний перегляд займає занадто багато часу, а при внесенні змін у програмі можуть виникнути додаткові помилки.

Штучний інтелект, як випливає із самої назви, надає комп'ютеру деякі риси інтелектуальності. Методи ШІ спрощують об'єднання програм і дають можливість закласти в системі штучного інтелекту здатність до самонавчання і накопичення нової, корисної в подальшому інформації. Людина може накопичувати знання, не змінюючи спосіб мислення і не забуваючи вже відомі факти. Система ШІ працює майже так само. Таким чином, модифікувати програми ШІ значно простіше, ніж традиційні програми.

Вплив ШІ на програмування

Методи ШІ припускають високий ступінь незалежності окремих частин програми, які реалізують певний етап вирішення однієї чи декількох задач. Розглянемо цей процес більш докладно. Незалежні частини програми можна порівняти з окремими блоками інформації в пам'яті людини. Вибираючи потрібну інформацію, мозок людини автоматично підключає лише необхідні для вирішення задачі факти, не перебираючи всі доступні йому знання.

Одну й ту саму задачу можна запрограмувати, використовуючи або традиційні методи, або методи штучного інтелекту. Застосування методів штучного інтелекту дозволяє істотно спростити і прискорити розробку програм. Окремі частини програм з використанням обох методів виконують точно визначені дії, однак програми ШІ мають особливу властивість, подібну до характерної властивості людського інтелекту: зміна будь-якої, навіть невеликої частини інформації не впливає на структуру всієї програми. Така гнучкість сприяє більш ефективному процесу програмування, а також дає можливість створювати програми, що вміють “розуміти”, тобто в яких присутні риси інтелектуальності.

Мислення людини

Штучний інтелект основується на знаннях про процес мислення людини. Звичайно, точно невідомо, як працює мозок людини, вчені лише починають розуміти складні механізми інтелекту. Однак для розробки прикладних програм штучного інтелекту наявних знань цілком достатньо.

Цілепокладання

В основі людської діяльності лежить процес мислення. Наприклад, у випадку, коли зранку дзвонить будильник, мозок людини дає команду руці відключити його. Це не автоматична реакція, адже вирішення конкретної задачі вимагає певної відповіді мозку. Поняття мети (цілі) можна визначити як кінцевий результат, на який спрямовані розумові процеси людини. Як тільки мета (виключити будильник) досягнута, перед мозком людини відразу постають нові цілі, наприклад: піти у ванну, почистити зуби, одягтися, поснідати, вийти на автобусну зупинку тощо. Здійснення усіх цих цілей призводить до досягнення головної мети – вчасно потрапити на роботу, в нашому прикладі. Думки, що приводять до кінцевого результату, не випадкові, а чітко обґрунтовані. Кожен крок на шляху до головної мети має свою локальну мету. Мозок завжди зосереджений на меті незалежно від того чи виконує людина просту фізичну роботу, чи вирішує складну інтелектуальну задачу. Мета змушує людину мислити.

Факти і правила

Мозок людини – це велике сховище знань. Людині властиво здобувати нові знання і застосовувати їх у відповідних ситуаціях. Інтелект в спрощеному випадку можна подати як сукупність фактів і способів їх

застосування для досягнення мети. У більшості випадках цілі досягаються за допомогою правил використання всіх відомих фактів.

Звичайно, для досягнення мети люди використовують та взаємозв'язують складні сукупності фактів і правил.

Спрощення

У випадку, коли мозок людини починає процес вирішення найпростішої задачі, для вибору потрібних дій у його розпорядженні є величезні обсяги інформації. Наприклад, прямуючи на роботу, людина виходить з будинку і йде на перехрестя. У той час коли вона обирає момент для переходу вулиці, у її мозок надходить велика кількість абсолютно різнорідної інформації. Перш ніж перетнути вулицю, людина аналізує швидкість і інтенсивність руху, відстань до протилежного тротуару, сигнали світлофора на перехресті. Одночасно мозок людини обробляє враження, які не мають прямого відношення до процесу переходу вулиці, наприклад погодні умови, колір і моделі машин, які проїжджають, вид і висоту дерев, що ростуть біля дороги, вигляд розташованих неподалік будинків. Безсумнівно, людина також думає про місце, куди вона йде, про те, як швидко їй необхідно прибути туди, кого вона може зустріти і т.і.

Якби людина, перш ніж стати на проїзну частину, аналізувала усі факти, що мають пряме, непряме або взагалі не мають ніякого відношення до її мети – перейти вулицю, вона простояла би на тротуарі кілька років. Яким же чином людський мозок з великої кількості фактів і правил швидко обирає підмножину, що стосується лише конкретної ситуації? Справа в тому, що в мозку існує складна система, що керує вибором правильної реакції на відповідну ситуацію. Такий вибір називається спрощенням. Механізм спрощення „блокує” думки, що не мають відношення до розв'язуваної в даний момент задачі. Так само, як видалення у дереві непотрібних гілок допомагає йому рости, механізм спрощення сприяє досягненню мети, ігноруючи всі неважливі для цього факти. Коли людина зіштовхується з деякою ситуацією, механізм спрощення змушує її мозок зосередитися лише на фактах і правилах, які необхідні для досягнення поставленої мети.

Враховуючи вищесказане, можна відзначити, що для того, щоб мати ознаки та властивості штучного інтелекту, програмна система повинна мати всі елементи, які складають процес ухвалення рішення людиною, цілі, факти, правила, механізми висновку і спрощення.

Головна відмінність систем ШІ від традиційних програмних систем полягає в тому, що різні компоненти її структури (рис. 1.2) визначаються окремо і модифікація будь-якої частини не потребує перебудови чи модифікації загальної структури. Завдяки такому підходу можна виділити окремі складові процесу мислення людини, що вирішує певну задачу, та включити їх у систему ШІ. Визначивши, як відбувається процес мислення

людини на кожній стадії процесу прийняття рішення, у програму легко можна включити блок, що реалізує дії, аналогічні мисленню людини на цій же стадії.

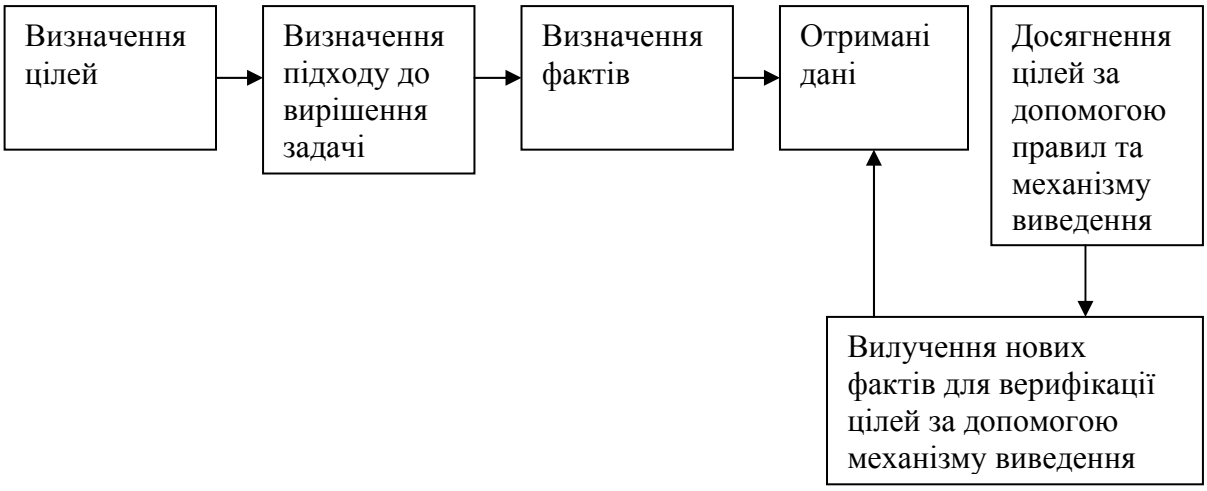

Рисунок 1.2 - Компоненти програмних систем ШІ

1.3 Експертні системи як різновид систем штучного інтелекту

Експертні системи є відносно молодим науковим напрямком, що виник в межах досліджень із штучного інтелекту в середині 70-х років XX ст. Свій розвиток він отримав завдяки суттєвим змінам, що відбулись в цей час в технології розробки та використання програмних засобів штучного інтелекту [8]. Найбільш важливими з них стали:

- відокремлення в програмі деякої універсальної частини (логічного виведення) від частини, яка залежить від предметної області (бази знань);
- підвищення рівня взаємодії користувача з комп'ютерною програмою.

Важливість цих змін стає більш зрозумілою, якщо звернути увагу на головну особливість процесу створення інтелектуальної програми, який полягає в виконанні послідовності таких дій:

- визначається необхідність модифікації;
- здобуваються нові знання, які дозволяють підвищити якість функціонування системи;
- нові знання перетворюються в форму, „зрозумілу" комп'ютерній системі;
- виконується модифікація знань системи.

При створенні перших інтелектуальних програм всі вищевказані дії виконував програміст. Але скоро виникла суперечність, яка полягала в тому, що для створення досконалих програм програмісту потрібно було мати рівень знань найкращих фахівців (експертів) у відповідній проблемній області, або експертам досконало опановувати програмування.

Відокремлення знань про предметну область і оформлення їх у вигляді достатньо простих інформаційних систем дозволило перекласти виконання першої та другої дії на експерта та інженера зі знань. При цьому за програмістом залишилась необхідність введення, отриманих від експерта знань, в комп'ютер. Створення розвинутих інтелектуальних інтерфейсів дозволило повністю перекласти цю задачу на комп'ютер, таким чином усунувши програміста від процесу модифікації системи.

Основні відмінності інтелектуальних програм від традиційних, які якісно були виділені на той час, можна відобразити у вигляді таблиці:

Таблиця 1.1 – Відмінності інтелектуальних програм від традиційних

Характеристика	**Інтелектуальні програми**	**Традиційні програми**
Тип обробки даних	Символьна обробка	Числова обробка
Метод обробки	Евристичний	Алгоритмічний
Визначення кроків розв'язання	Неявне	Точне
Шукані рішення	Задовільні	Оптимальні
Розподіл управління і даних	Роздільні	Змішані
Модифікація	Часта	Рідка

Традиційні програми мають фіксовану послідовність кроків, точно визначених програмістом і шляхом обробки числової інформації здійснюють пошук оптимальних рішень. В той же час інтелектуальні програми, подібно до людини, використовують для пошуку задовільного рішення метод „спроб та помилок".

Наприклад, для розв'язання задачі запобігання захвату літаків, згідно з алгоритмічним методом необхідно: провести особистий огляд кожної особи, яка проходить на посадку (в тому числі не тільки пасажирів, але і льотного складу, механіків) та обшукати весь багаж. Такий алгоритм забезпечить повну гарантію запобігання захвату літака, оскільки виключить можливість проникнення на борт літака зброї. Але він потребує занадто багато часу, має велику вартість і занадто непопулярний серед пасажирів та персоналу, щоб мати практичну цінність. Разом з тим, відповідний евристичний метод може базуватись на таких положеннях: пропустити всіх пасажирів через детектор металу; оглянути тільки тих з них, на яких реагує детектор, а також тих, чиї прикмети збігаються з характерними прикметами зловмисника (вік, одяг, поведінка і т.і.). Такий метод допоможе запобігти більшості випадків захвату літаків, але не гарантує, що вони взагалі не виникнуть. Використання евристичних методів робить пошук рішення набагато простішим і більш практичним.

Експерти, (а відповідно і ЕС), розв'язують складні задачі без використання систем рівнянь або інших складних математичних

обчислень. Замість цього, вони за допомогою символів подають поняття предметної області і застосовують різні стратегії і евристики при маніпулюванні цими поняттями. В ЕС знання, що відповідають поняттям та зв'язкам предметної області, також подаються в символьному вигляді. Неможна стверджувати, що ЕС взагалі не здійснює математичних обчислень, але в ній переважно використовується символьна обробка вмісту бази знань.

Наведена різниця між традиційними та інтелектуальними програмами не є абсолютною, а лише найбільш характерна для обох типів програм. Загалом, всі особливості інтелектуальних програм характерні, в цілому, і для експертних систем.

Щоб з'ясувати зміст експертного аналізу розглянемо такий простий приклад. Після інсталяції певної нової програми вона не працює, а видає повідомлення типу „Call to Undefined Link". Як і більшість інших повідомлень, це мало чим Вам допомагає. Ви видаляєте весь каталог і знов інсталюєте програму, але результат від цього не змінюється. Ви змінюєте настроювання в різних файлах ініціалізації, але й це не допомагає. І тоді Ви звертаєтеся до людини, яка працює у сервісному центрі і займається саме цими проблемами. Він радить викинути кілька застарілих DLL-модулів у системному каталозі і знову встановити програму. Виконавши його інструкції Ви за десять хвилин досягаєте успіху.

Отож, який би рівень експертного аналізу не був потрібним у даній області, спеціаліст із сервісного обслуговування здатний його зробити, а людина навіть з достатньо високими знаннями інформатики, яка програмує задачі лише у своїй предметній області – не здатна, оскільки не має досвіду у розв'язанні таких проблем. Отож, здібність виконувати експертний аналіз – це не тільки питання наявності певних знань і рівня кваліфікації. Для цього треба мати і дуже специфічні навички і уміння розібратися у конкретній ситуації в межах даної предметної області. Тобто, бути експертом і мати загальну освіту – це далеко не одне й те ж саме.

Виконання перерахованих нижче умов дозволяє віднести комп'ютерну програму до класу експертних:

- програма має оперувати знаннями. Просто виконувати деякий алгоритм, наприклад аналізувати об'єкти зі списку на наявність певної властивості, яка явно не відповідає таким вимогам. Наприклад, це те ж саме, що дати випадковій людини перелік симптомів та відповідних ліків і очікувати, що вона зможе успішно лікувати людей. Дуже швидко вона зіткнеться з ситуацією, не передбаченою у списку, який їй дали;
- знання, якими оперує система, мають бути сконцентровані на певну предметну область. Випадковий набір імен, дат, місць подій і т. ін. – не є знаннями, які є основою для експертного аналізу. Знання вимагають певної організації та інтеграції, тобто окремі відомості мають співвідноситися одні до одних і створювати щось подібне ланцюжку, в якому одна ланка безпосередньо пов'язана з іншою;

• з наявних у системі знань має безпосередньо випливати розв’язок проблеми. Просто продемонструвати деякі знання, наприклад правила технічного обслуговування комп’ютера, зовсім не те ж саме, що відремонтувати його. Також як і отримання доступу до технічної документації зовсім не те, що запрошення фахівця, здатного розв’язати проблему, яка виникла.

Таким чином, як загальне означення експертних систем можна прийняти означення, близьке до запропонованого вченим Фейгенбаумом: «Експертна система – це інтелектуальна комп’ютерна програма, яка використовує знання та процедури виведення для розв’язання проблем у певній проблемній області, настільки складних, що для їх розв’язання потрібно запрошувати експерта, а також виробляє рекомендації для розв’язання цих проблем».

Системи, які основані на знаннях, зберігають правила розв’язання проблем конкретної проблемної області у базах знань. Проблема ставиться перед системою у вигляді сукупності фактів, що описують деяку ситуацію. При цьому система за допомогою бази знань робить спроби виведення з цих фактів деякого висновку.

Предметом теорії ЕС є методи і прийоми конструювання людино-машинних систем, компетентних у вузькоспеціалізованій області. Така компетентність полягає в знанні предметної області, розумінні задач з цієї області і умінні розв’язувати деякі з цих задач. Знання, які використовуються у будь-якій області, існують як правило у двох видах: загальнодоступні (факти, означення, теорії, які викладені в підручниках та довідниках) і індивідуальні (які базуються на особистому досвіді спеціаліста і є чимось більшим ніж загальнодоступні знання). Такі індивідуальні знання значною мірою складаються з емпіричних правил, які називаються *евристиками*. Саме евристики і дозволяють експертам знаходити ефективні шляхи до розв’язання задач в умовах перекручених та неповних даних.

Відмінною рисою ЕС є здатність накопичувати знання і досвід найбільш досвідчених експертів в деякій вузькій предметній області. Внаслідок цього, користувачі ЕС, що мають середню кваліфікацію, можуть розв’язувати свої поточні задачі так само вдало, як це зробив би сам експерт. Такий ефект досягається завдяки тому, що ЕС в процесі своєї роботи використовує ту ж саму схему міркувань, яку використав би в даній ситуації експерт. Таким чином, вона дозволяє зберігати, акумулювати та розповсюджувати знання, дозволяючи зробити унікальний досвід кількох висококласних професіоналів доступним широким колам рядових фахівців.

Рівень користувачів ЕС може змінюватись в широких межах – від простого бухгалтера до консультанта уряду. Від роду діяльності користувача залежать і функції, які виконуватиме ЕС. Це може бути рутинна обробка великих масивів інформації з метою виявлення деяких

закономірностей або незвичайних відомостей, які свідчать про порушення звичайного порядку речей і потребують прийняття відповідного рішення. Інший клас ЕС орієнтовано на користувачів середньої кваліфікації, яким необхідно використовувати знання експертів для розв'язання щоденних проблем, наприклад, постановки діагнозу.

Загалом ЕС може бути розроблена для будь-якої предметної області, виконувати такі узагальнені функції:

- аналізувати та класифікувати;
- навчати та навчатись;
- ідентифікувати та інтерпретувати інформацію;
- здійснювати діагностику та тестування.

Варто відзначити, що остаточне рішення може безпосередньо формуватися ЕС, а може видаватися й через проміжну ланку – людину, яка працює з програмою. Така людина і сама може бути експертом, при цьому задачею ЕС буде просто підвищення ефективності роботи експерта. Спеціаліст же середнього рівня кваліфікації за допомогою ЕС може підвищити якість своєї роботи. Отже, правильний розподіл функцій між людиною і КС, є однією з ключових умов високої ефективності впровадження ЕС.

Велика зацікавленість в ЕС пояснюється як мінімум такими причинами:

• орієнтацією на розв'язання широкого кола задач в неформалізованих областях, які раніше були недоступні для обчислювальної техніки;

• надання спеціалістам, які не знайомі з програмуванням, можливості самостійно розробляти потрібні ним прикладні програми, що різко розширює коло користувачів КС;

• досягненням при розв'язанні практичних задач результатів, які не поступаються а інколи і перевищують можливості людини-експерта не забезпеченого комп'ютером.

Взагалі, характерною рисою ЕС є те, що необхідні знання і процедури виведення, які вона використовує, можуть розглядатися як модель проведення експертизи найкращим з експертів в своїй області.

Місце, яке займають ЕС як один із стилів програмування в цілому можна зобразити у вигляді такого рисунку:

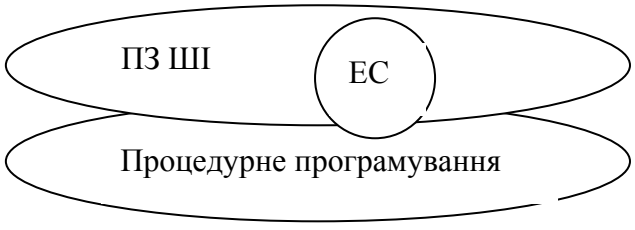

Рисунок 1.3 - Місце ЕС як одного із стилів програмування

Отже, чітко визначеної межі між програмами ШІ та ЕС не визначено. І ті, і інші програми в загальному випадку називають програмами, які основані на знаннях.

Проте, можна навести такі основні ознаки, що відрізняють ЕС від звичайних програм:

- ЕС моделює не стільки фізичну або якусь іншу природу певної проблемної області, скільки механізм мислення людини стосовно до розв'язання задач з цієї області. Це суттєво відрізняє ЕС від систем математичного моделювання або комп'ютерної анімації. Не можна однозначно стверджувати, що програма повністю відтворює психологічну модель спеціаліста, але головна увага все ж таки приділяється саме відтворенню комп'ютерними засобами методики розв'язання проблем, яку застосовує експерт;
- ЕС крім виконання певних обчислювальних операцій, формує певні міркування й висновки, на основі тих знань, якими вона оперує. Знання в ЕС подані, як правило, на певній спеціальній мові і зберігаються окремо від власно програмного коду, який і формує висновки та міркування. Цей компонент і називається базою знань;
- При розв'язанні задачі основними є евристичні та наближені методи, які, на відміну від алгоритмічних, не завжди гарантують успіх. Евристики створюються людиною завдяки набуттю нею практичного досвіду і є приблизними в тому сенсі, що не потребують вичерпної початкової інформації і забезпечують певну впевненість (або невпевненість) у тому, що рішення, яке пропонується, є правильним.

Від інших програм штучного інтелекту ЕС відрізняються в першу чергу тим, що:

- ЕС мають справу з предметами реального світу, операції з якими звичайно потребують наявності значного досвіду, накопиченого людиною. Більшість програм ШІ є винятково дослідницькими, і головна увага в них приділяється абстрактним математичним проблемам або спрощеним варіантам реальних проблем, а метою таких програм є підвищення рівня інтуїції або опрацювання методики. ЕС мають чітко виражену практичну спрямованість у науковій або комерційній галузі;
- Однією з найголовніших характеристик ЕС є її продуктивність, тобто швидкість отримання результату і рівень його достовірності (надійності). Програми ШІ можуть бути повільними і припускатись помилок в окремих ситуаціях, оскільки це швидше інструмент досліджень, ніж програмний продукт. А ЕС повинна за прийнятний час знайти розв'язок не гірший за запропонований у даній ситуації експертом;
- ЕС повинна вміти роз'яснювати, чому прийняте саме таке рішення і довести його обґрунтованість. Користувач повинен отримати всю інформацію необхідну для того, щоб повірити, що рішення прийняте коректно та професійно. На відміну від цього, програми ШІ спілкуються зазвичай тільки зі своїми творцями, які і так розуміють, на чому базується

результат. ЕС проектується у розрахунку на взаємодію з різними користувачами, для яких її робота має бути по можливості прозорою.

Отже, на відміну від звичайних інтелектуальних програм, можна сказати, що ЕС є не просто системою, що основана на знаннях, а не на алгоритмічних або статистичних методах. Вона є програмою, яка може провести повноцінний експертний аналіз в даній предметній області.

Для визначення розбіжностей можна навести ряд відомих програм ШІ, що не відносяться до ЕС:

1) програма, яка здатна переглянути текст розповіді дитини, переказати його, дати відповіді на деякі запитання. Така програма повинна розуміти природну мову і причинно-наслідкові зв'язки, що існують в щоденному житті;
2) програма друку з голосу;
3) програма, яка розв'язує задачі на знаходження аналогій в геометричних фігурах (такі задачі включають до тестів на IQ);
4) програми, які самостійно доводять і навіть відкривають нові теореми.

Тобто, ЕС не є програмами, що виконують дії, для здійснення яких не потрібні знання експерта (1,3), розв'язують задачі розпізнавання, оскільки в загальному випадку вони розв'язуються чисельними методами (2), або задачі, які вирішуються, як правило, шляхом формальних перетворень і процедурного аналізу (4).

Серед характеристик, які дозволяють віднести інтелектуальну програму до ЕС слід відзначити такі:

• відображення внутрішнім функціонуванням програми підходу до проблеми з боку людини;

• можливість пояснення програмою своїх дій засобами та способом, який може зрозуміти людина;

• можливість взаємодії програми з оператором шляхом гнучкого діалогу, подібного до діалогу природною мовою.

Отже, ЕС містить знання у певній предметній області, що накопичені внаслідок практичної діяльності людини і використовує їх для розв'язання проблем специфічних для даної області. Цим вони відрізняються від традиційних систем, у яких використовуються більш загальні і менш пов'язані з конкретною проблемною областю теоретичні методи, частіше за все – математичні. Тому процес створення ЕС навіть називають інженерією знань, а не програмуванням.

Враховуючи все вищевказане, можна визначити основні структурні елементи класичної ЕС, відповідно до вказаних функціональних призначень та поставлених системних вимог [9].

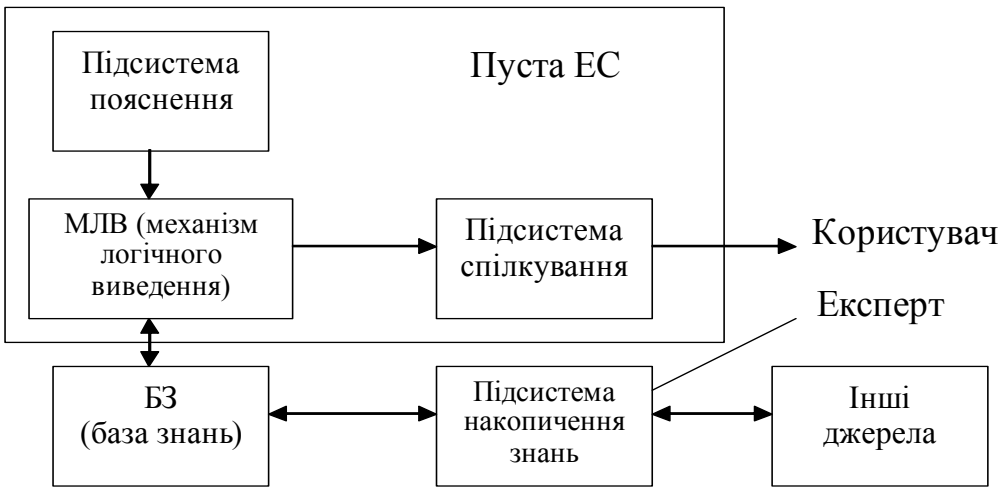

Рисунок 1.4 - Класична структура ЕС

Визначимо основні терміни приведеної структури.

Користувач – фахівець в певній ПО, для якого призначена система. Його кваліфікація недостатньо висока, тому він потребує допомоги і підтримки з боку ЕС.

Експерта доцільно представити в подвійній ролі – ролі фахівця зі ШІ (як проміжна ланка між БЗ і експертом – *підсистема накопичення знань*) і ролі експерта.

Інтерфейс користувача – *підсистема спілкування*: комплекс програм, що реалізують діалог користувача з ЕС як на стадії введення інформації, так і отримання інформації.

Підсистема пояснень - програма, що дозволяє користувачу одержати відповіді на питання "Як була одержана та або інша рекомендація?" і "Чому система ухвалила таке рішення?" Відповідь на питання „Як?” – трасування всього процесу отримання рішення з вказанням використаних фрагментів БЗ. Відповідь на питання „Чому?” - посилання на висновок, який безпосередньо передує одержаному рішенню.

Підсистема накопичення знань (інтелектуальний редактор) – програма, що представляє інженеру зі знань можливість створювати БЗ в діалоговому режимі.

Механізм логічного виведення (МЛВ) - програма, що моделює хід міркувань експерта на підставі знань, що є в БЗ.

База знань (БЗ) – ядро ЕС, сукупність знань предметної області, записана на машинний носій у формі, зрозумілій експерту і користувачу. Паралельно такому "людському" уявленню, існує БЗ в машинному уявленні.

1.4 Класифікація експертних систем

Клас ЕС сьогодні об'єднує декілька тисяч різних програмних комплексів, які можна класифікувати за різними критеріями. Розглянемо лише класифікацію за деякими глобальними критеріями [8].

Класифікація за типом задачі, що вирішується

Інтерпретація даних. Під інтерпретацією розуміється визначення значення даних, результати якого повинні бути узгодженими і коректними. Звичайно передбачається багатоваріантний аналіз даних. Це одна з традиційних задач для ЕС.

Діагностика. Під діагностикою розуміється виявлення несправності в деякій системі. Несправність – це відхилення від норми. Таке трактування дозволяє з єдиних теоретичних позицій розглядати і несправність устаткування в технічних системах, і захворювання живих організмів і всілякі природні аномалії. Важливою специфікою є необхідність розуміння функціональної структури системи діагностування.

Моніторинг. Основна задача моніторингу – безперервна інтерпретація даних в реальному масштабі часу і сигналізація про вихід тих чи інших параметрів за допустимі межі. Головні проблеми – пропуск тривожної ситуації і інверсна задача помилкового спрацьовування. Складність цих проблем – в розмитості симптомів „тривожних" сигналів.

Проектування. Підготовка специфікацій на створення об'єктів з наперед визначеними властивостями. Під специфікацією розуміється весь набір необхідних документів: креслення, записка пояснення і т.д. Основні проблеми тут – отримання чіткого структурного опису знань про об'єкт. Для організації ефективного проектування необхідно формувати не тільки самі проектні рішення, але і мотиви їх ухвалення. Таким чином, в задачах проектування тісно пов'язуються два основні процеси: процес виведення рішення і процес пояснення.

Прогнозування. Системи прогнозування логічно виводять вірогідні наслідки із заданих ситуацій. У системі прогнозування звичайно використовується параметрична динамічна модель, в якій значення параметрів підганяються під задану ситуацію. Наслідки, що виводяться з цієї моделі, складають основу для прогнозів з оцінками вірогідності.

Планування. Під плануванням розуміється знаходження планів дій, що відносяться до об'єктів, здатних виконувати деякі функції. У таких ЕС використовуються моделі поведінки реальних об'єктів з тим, щоб логічно вивести наслідки планованої діяльності.

Навчання. Системи навчання діагностують помилки при вивченні будь-якої дисципліни за допомогою ЕОМ і підказують правильні рішення. Вони акумулюють знання про гіпотетичного учня і його характерні помилки, потім в роботі здатні діагностувати слабкості в знаннях тих, кого навчають, і знаходити відповідні способи для їх ліквідації. Такі системи

планують процес спілкування з учнем залежно від успіхів учня, з метою передачі знань.

У загальному випадку, всі системи, основані на знаннях, можна розділити на:

- системи, які вирішують задачі аналізу,
- системи, які вирішують задачі синтезу.

Основна відмінність задач аналізу від задач синтезу – в задачах аналізу безліч рішень може бути перераховано і включено в систему, а в задачах синтезу безліч рішень потенційно будується з рішень компонентів або підпроблем. Задачі аналізу – інтерпретація даних, діагностика, задачі синтезу – проектування, планування. Комбіновані задачі – навчання, моніторинг, прогнозування.

Класифікація за зв'язком з реальним часом

Статичні ЕС – розробляються в предметних областях, в яких БЗ і дані, що інтерпретуються, не змінюються в часі, вони стабільні.

Квазідинамічні ЕС – інтерпретують ситуацію, яка змінюється з деяким фіксованим інтервалом часу.

Динамічні ЕС – працюють в поєднанні з датчиками об'єктів в режимі реального часу з безперервною інтерпретацією даних, що надходять.

Класифікація за типом ЕОМ

На сьогоднішній день можна виділити:

- ЕС для унікальних стратегічно важливих задач на СУПЕР-ЕВМ (CRAY, CONVEX);
- ЕС на ЕОМ середньої продуктивності (mainframe);
- ЕС на символьних процесорах і робочих станціях (SUN, APOLLO);
- ЕС на міні- та суперміні ЕОМ (VAX, micro-VAX);
- ЕС на ПК (IBM PC, MAC).

Класифікація за ступенем інтеграції з іншими програмами

Автономні ЕС – працюють безпосередньо в режимі консультацій з користувачем для специфічних експертних задач при рішенні яких не вимагається залучати традиційні методи обробки даних.

Гібридні ЕС – програмний комплекс, що агрегує стандартні прикладні програми (наприклад, математичну статистику, лінійне програмування, СУБД) і засоби маніпулювання знаннями. Це може бути інтелектуальна надбудова над прикладними програмами або інтегроване середовище для вирішення складної задачі з елементами експертних знань. Не дивлячись на зовнішню привабливість гібридного підходу, розробка таких систем є надзвичайно складною задачею. Компонування не просто різних програм, а різних методологій породжує цілий комплекс і теоретичних, і практичних труднощів.

1.5 Контрольні питання

1. Поняття системи штучного інтелекту.
2. Нейрокібернетика та кібернетика “чорного ящика”.
3. Взаємозв’язок ЕС та СШІ. Місце ЕС в структурі ШІ.
4. Вплив штучного інтелекту на ідеологію програмування.
5. Основні відмінності інтелектуальних програм від традиційних.
6. Поняття експертної системи та її прикладне значення.
7. Сутність експертного аналізу. Основні ознаки класу експертних програм.
8. Взаємозв’язок та відмінності ЕС, програмних засобів ШІ і традиційних програм.
9. Класична структура ЕС.
10. Класифікація систем штучного інтелекту.

2 СФЕРИ КОМПЕТЕНЦІЇ ЕКСПЕРТНИХ СИСТЕМ

2.1 Порівняння людської і штучної компетенції

Експерт володіє величезними знаннями, уміє швидко знайти головну інформацію для розпізнавання ситуацій. Завдяки навчанню й досвіду він може робити те, що звичайні люди виконувати не здатні. Експерт здійснює пошук у просторі альтернатив вибірково, зводячи до мінімуму зайву роботу, ухиляючись від малоуспішних зусиль, якнайшвидше відсікаючи безперспективні шляхи дослідження. На ранніх стадіях експерт досягає високої продуктивності, оскільки найефективніше використовує свій час.

Висококваліфіковані експерти дуже цінуються в усьому світі, одержують високу зарплату. Однак, людина-експерт може втрачати свої навики, виявляти небажані суб'єктивні схильності, а при відсутності помічників і не мати достатньої інформаційної бази для прийняття рішень.

За цих умов, незважаючи на дорожнечу розробки комп'ютерної експертної системи (роки роботи інженерів і експертів), її початкова вартість компенсується потім якістю прийнятих рішень і надійністю в роботі, а при достатньому тиражуванні розробки – часто й доступною вартістю при експлуатації. Тож перед початком розробки необхідно врахувати всі переваги й недоліки ручної і машинної праці, тобто людської і штучної компетенції [10].

Таблиця 2.1 – Переваги штучної компетенції над людською

Людська компетенція	**Штучна компетенція**
1. Нестійка, залежить від тренування	Стійка, не зменшується. Залежить тільки від стану техніки
2. Важко передається і тиражується	Передається як програмний продукт. Особливо ефективна як тиражування досвіду, що потрібний у декількох місцях одночасно
3. Важко документується	Документується як комп'ютерна програма
4. Іноді непередбачувана	Точно відповідає алгоритму
5. Дорога за вартістю	Може бути припустимою за вартістю
6. Неможливість роботи в умовах небезпечного середовища	Може використовуватися в умовах ядерних реакторів, хімічних виробництв і т. п.

Проте людська компетенція має й свої переваги. Вони виявляються при розв'язуванні задач, що вимагають інтуїції, творчого підходу. Існує умовна класифікація типів експертів:

- **ініціатор** – швидко відчуває перспективну проблему, один із перших її формулює;
- **діагност** – здатний до конструктивного оцінювання проблеми, її сильних і слабких сторін;
- **ерудит** – має виняткову пам'ять, схильний до визначення упорядкувань і класифікацій, виявлення уваги до деталей;
- **методолог** – виявляє здібності в теоретичних моделях, гіпотезах, іноді не маючи прямих обов'язків до застосування;
- **ремісник** – втілює ідеї інших. Іноді робить це краще, ніж самі ініціатори;
- **естет** – працює привабливо, але іноді не схильний до подробиць, до «нудної» технології;
- **незалежний** – схильний до суто індивідуального шляху дослідження. Іноді знаходить оригінальні розв'язки;
- **фанат** – самоствердкуюча позиція, яка потребує натхнення від себе та інших. Іноді занадто засліплений одним напрямком дослідження;

Найкращий варіант – поєднання різних типів експертів, що дає змогу забезпечити глибину, різнобічність аналізу, достатні рівні теоретичного і практичного реалізування розробки.

Таблиця 2.2 – Переваги людської компетенції перед штучною

Людська компетенція	**Штучна компетенція**
1. Творча	Запрограмована
2. Адаптується до зміни умов	Вимагає підказування при найменших змінах умов
3. Широка за охопленням предмета	Вузько спрямована
4. Відсікає явно помилковий пошук (здоровий глузд)	Діє завжди як педант (наприклад, чи є В.Шекспір у списку працівників?)
5. Спроможність реагування на різні види інформації (іноді не передбачені інструкцією)	Найчастіше як вхідна використовується символьна інформація

2.2 Критерії вибору задач, що реалізуються методами і засобами експертних систем

Критерій – це свого роду правило, за допомогою якого можна що-небудь оцінити.

До основних критеріїв вибору задач, що реалізуються методами і засобами експертних систем, можна віднести такі:

1. Дані і знання повинні бути надійні, достовірні, не змінюватись в часі, тобто бути стабільними в процесі вирішення задачі: не повинні коректуватися і не повинні містити помилок і суперечностей.
2. Простір або область можливих рішень відносно невелика. Простір пошуку повинен бути невеликим, оскільки необхідно зосередитися на вузькій предметній області, для якої характерний невеликий об'єм знань, у тому числі і основаних на здоровому глузді.
3. В процесі вирішення задачі повинні використовуватися формальні міркування, а задача повинна бути не дуже проста і не дуже важка для експерта (з розрахунку приблизно 30 хв. для експерта).
4. Задача повинна бути поставлена чітко, тобто визначені цілі (або мета), які ставляться перед експертною системою в процесі консультації, необхідний набір евристик, які використовуються в процесі вирішення задачі людиною.
5. Повинен бути, принаймні, один експерт, що вміє чітко виражати свої думки: явно формулювати свої знання і пояснювати методи застосування цих знань для вирішення задачі (експлікувати знання).

Також можна виділити додаткові (неявні) критерії вибору задач, що реалізуються методами і засобами експертних систем:

1. Існування формалізованого апарату даної предметної області. Однією з найпривабливіших властивостей систем ШІ є здатність автоматизувати процес міркувань фахівців предметної області навіть в тих додатках, де відсутня для цього формалізована теоретична база. Це області, в яких застосовуються емпіричні асоціації для отримання шуканого результату.
2. Відбір такої предметної області, для якої підходять вербальні міркування. Області, в якій знання відносяться або до області відчуттів, або не можуть бути висловлені і носять характер практичного досвіду, мало придатні для використання в системах штучного інтелекту.

В цілому, ЕС не рекомендується застосовувати для таких типів задач:

- математичних, які розв'язуються звичним шляхом формальних перетворень або процедурного аналізу;
- задач розпізнавання, які в загальному випадку розв'язуються чисельними методами (хоча в деяких ситуаціях це твердження спірне);

- задач, знання про методи вирішення яких відсутні.

Істотно утруднене створення ЕС для предметної області, в якій має місце одна або декілька перерахованих нижче особливостей:

- думки експертів часто не збігаються;
- використовуються складні стратегії міркувань;
- знання включають просторові і тимчасові співвідношення;
- є дуже велике число об'єктів розгляду і дуже багато залежить від понять, що опираються на "здоровий глузд";
- експертам потрібен значний час для вирішення задачі.

Відповідно до вищевказаного, тепер можна виділити клас задач, для яких розробка ЕС є найбільш придатною:

1. Задачі прогнозу і класифікації, які припускають велике число можливих відповідей (декілька десятків).
2. Багатопараметричні задачі, для яких важко, а іноді і неможливо визначити відповідні строгі аналітичні залежності.
3. Задачі, для вирішення яких потрібна тривала професійна підготовка.
4. Задачі обробки недостовірної інформації.

Приступаючи до створення ЕС необхідно спочатку вирішити, чи є створення і застосування ЕС для задачі, що стоїть перед кінцевим користувачем, **доцільним (доречним), виправданим і можливим.**

Критерії доцільності розробки ЕС:

- вирішення задачі опирається на використання операцій з символами, тобто задача не стільки пов'язана з розрахунками за формулами, скільки з логічним аналізом і перебором варіантів;
- задача не має чіткого алгоритмічного уявлення і опирається на використання евристик, які ведуть до результату, економлять набір переборів, але не дають гарантію успіху;
- задача не тривіальна;
- задача має кінцеву, прийнятну розмірність, тобто не є дуже громіздкою для вирішення на ЕОМ;
- задача викликає великий інтерес для практики і дозволяє знаходити рішення, які за допомогою класичних методів не можуть бути одержані.

Критерії виправданості розробки ЕС (хоча б одна позиція виконується):

- вирішення задачі має високу значущість або обіцяє принести високий дохід (не обов'язково в грошовому виразі);
- досвід вирішення задач в даній предметній області поступово втрачається;
- рівень підготовки фахівців-експертів в даній предметній області надзвичайно низький;

- накопичений досвід потрібен в багатьох областях (тиражування досвіду і знань)
- експертизи проводяться в середовищі, небезпечному для людини.

Критерії можливості розробки ЕС (обов'язкові всі вимоги):

- задача потребує лише інтелектуальних навиків;
- експерти можуть чітко висловити і описати використані ними методи роботи, які застосовуються у вирішенні даної задачі;
- є фахівці, чиї знання і досвід можна закласти в ЕС,
- експерти одностайні в рішенні задачі;
- задача не дуже важка;
- задача достатньо зрозуміла, якщо для вирішення задачі необхідно розробляти спеціальні методи, то ЕС не застосовується;
- задача вимагає знань і здорового глузду, які є результатом накопичення практичного досвіду, і цей досвід з якихось причин не вдається виділити і формалізувати.

Відповідно тепер можна сформулювати основні властивості експертної системи:

- наявність доступної бази знань;
- накопичення й зберігання високоякісного досвіду (інституціональна пам'ять);
- прогностичні можливості за рахунок включення відповідних моделей і алгоритмів обробки даних і знань;
- можливість навчання й тренування;
- здатність пояснення процесу прийняття рішень;
- наявність засобів редагування, що допомагають експертам модифікувати знання;
- наявність інтерфейсу з можливостями графічного виведення інформації.

Зокрема, видатний український вчений В.М. Глушков визначив клас динамічних систем управління [10] з використанням людино-машинних методів за участю багатьох експертів (можуть бути сотні й тисячі) як ідеальну розподілену експертну систему. Кожний експерт оцінює свій параметр або їх сукупність за багатьма критеріями з урахуванням багатьох обмежень, що можуть бути суперечливими (так звані «невласні» моделі). Допускаються конфліктні моделі (ситуації з протилежними інтересами), недостовірні (імовірнісні) дані й оцінки, моделі сценаріїв. Тут передбачається застосування алгебраїчного апарату, методів диференціального та інтегрального числення, лінійного та нелінійного програмування.

2.3 Рівні реалізації експертної системи

За аналогією з попереднім матеріалом, визначимо критерії оцінювання рівня реалізації експертної системи (ЕС):

- компетентність – здатність досягати експертного рівня прийняття рішень (умілі дії, адекватні обставинам);
- робастність – система лише поступово зменшує якість прийняття рішень по мірі наближення до меж компетенції (відсутність «провалів» на краях області можливого пошуку рішень);
- глибина – система здатна вирішувати задачі реальної складності, а не на рівні лише ідей або виготовлення «красивих іграшок». Рішення, знайдене на умовних прикладах, іноді не може бути застосоване в масштабах, що визначають реальну проблему;
- здатність до самоаналізу – здатність пояснювати дії у процесі роботи системи. Стосується сфери метазнань – знань про те, як система організовує самі знання;
- ефективність – окупність системи.

Важливе значення має захист інформації, проблема забезпечення таємниці (якщо це необхідно). Річ у тому, що автоматизовані системи, по суті, є відкритими системами. Поряд із керівником проекту, виконавцем, споживачем до інформації допущені оператор даних, адміністратор, організатор виконання робіт тощо.

Для захисту електронних документів існують спеціальні засоби санкціонованого доступу [10] з можливістю криптографічної експертизи, шифрування відправлень.

Формула оцінювання економічної ефективності застосування ЕС:

$$U = \frac{P}{A_1}(A_2 - A_1) + A_2(C_1 - C_2), \tag{2.1}$$

де A_1 – обсяг виробництва до впровадження системи;

A_2 – очікуваний обсяг виробництва після впровадження системи;

P – прибуток до впровадження системи;

C_1 – собівартість виробництва до впровадження системи;

C_2 – собівартість виробництва після впровадження системи.

Витрати на розробку та впровадження системи:

$$K_1 = N + E \times K \tag{2.2}$$

де N – витрати на функціонування системи;

K – витрати на розробку системи;

E – коефіцієнт зведення капітальних і поточних витрат. Коефіцієнт ефективності вкладень:

$$\mu = \frac{U}{K_1}. \tag{2.3}$$

Приклади ефективної реалізації елементів експертних систем давало вітчизняне виробництво. Так, внаслідок комп'ютеризації в одній із шахт Донбасу вдалося замінити 40 табельників, 7 обліковців і 7 працівників планово-економічних служб. Диспетчер, головний економіст, директор на своїх дисплеях безупинно одержували інформацію про стан підземного виробітку, транспортних магістралей, матеріальних і енергетичних потоків, вентиляції тощо. Розвиток подібних систем передбачено в державній «Програмі створення дистанційно-керованих агрегатів, комплексів устаткування, машин і маніпуляторів для виконання робіт в екстремальних умовах без присутності людей».

Рекомендації щодо розробки експертної системи

При виборі експертів, крім вимоги достатньої компетентності, необхідні: уміння чітко формулювати думки, заінтересованість у співпраці, розуміння корисності розробки. Інакше експерт обмежиться рекомендаціями, тоді як при розробці експертної системи необхідна напружена взаємодія.

Щодо розробників, то вони, зазвичай, вибирають інструментальні засоби ті, які знають, а не ті, які підходять до справи. Цю обставину необхідно враховувати разом з орієнтацією на вибір перевірених засобів програмування. В іншому випадку виникає необхідність боротися з помилками самої системи, які важко виявляються.

Користувач не прийме систему, яка працює погано, але він також не прийме систему, яка хоч і дає високі результати, але його роздратовує, заганяє у сліпий кут незрозумілими питаннями або вимагає трудомісткого введення інформації.

Може виникнути протиріччя у зв'язку із способом фінансування: якщо фінансує роботу одна організація, а користувачем є інша, то розроблювачі орієнтуватимуться на тих, хто їх фінансує, тоді як запити дійсних споживачів не будуть враховані.

Варто враховувати також чинник спадної граничної ефективності, коли кожен наступний крок підвищення якості системи дається з дедалі більшим зусиллям, Так, у [11] описується медична експертна система «PUFF», початкова продуктивність якої відповідала наборові 100 правил-продукцій. Треба було збільшити кількість правил до 400, щоб поліпшити якість роботи системи на 10 %.

Чинник спадної граничної ефективності в теорії споживання був сформульований у вигляді принципів Госсена (1810-1858): у міру задоволення потреб ступінь задоволення падає; при неможливості задовольнити всі потреби їх задоволення необхідно обмежити на тому рівні, на якому відчувається однаковий приріст задоволення від кожного виду потреб.

Основний принцип, який забезпечує успішний результат у створенні експертної системи, – постійна готовність перегляду будь-якої її складової:

для того, щоб побачити, як рухатися далі, необхідно виробити вміння повертатися до початку розробки й перевіряти відповідність знайденого рішення початковому задуму. Гарантованому успіху передує, звичайно, створення діючого прототипу по всіх складових, від входу до виходу, реалізації системи.

2.4 Аналіз практично-прикладних експертних залежностей

Кращими прикладами оптимізації функцій і навіть справжніми кібернетичними системами є живі істоти в здійсненні процесів регулювання температури тіла, кров'яного тиску й інших параметрів свого стану. Саме вони демонструють властивість гомеостазу системи - властивість зберігати значення істотних параметрів у деяких заданих межах.

Відома гіперболічна залежність $y = a/x^b$ між шириною і довжиною тулуба тварини, де a,b - кількісні параметри, що характеризують вид тварини.

Приведемо ряд експертних закономірностей, виявлених у різних галузях людської діяльності [12]:

Закон Лоткі:

Альфред Лоткі проаналізував кількість наукових статей, опублікованих авторами окремих галузей знань. Виявилося, що кількість авторів дуже точно описується залежністю (2.4).

$$n(x) = n_1/x^2. \qquad (2.4)$$

де n_1 - кількість авторів, котрі опублікували хоча б одну статтю;

x - кількість статей, опублікованих одним автором.

Якщо вибрати 100 авторів, котрі опублікували наукову статтю, то з них тільки 25 опублікувало ще одну статтю, три статті опублікувало 11 авторів, чотири – 7 авторів,..., десять – тільки один автор.

Графік залежності (2.4) поданий на рис. 2.1.

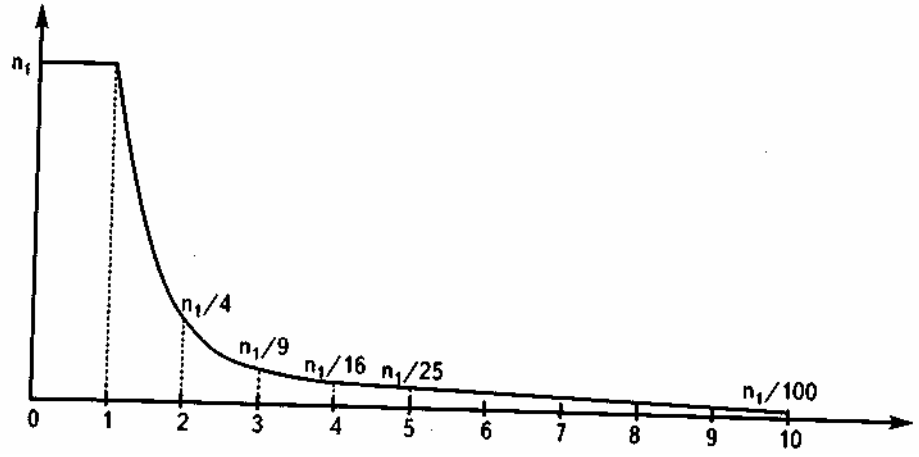

Рисунок 2.1 - Графік залежності за законом А. Лоткі

Закон Лоткі перевірений за інформацією різних наукових журналів. Виявлено повну відповідність даним статистики, що свідчить про цікаву закономірність зі сфери інтенсифікації інтелектуальної діяльності.

Закон Ципфа:

$$N(x) = N / x^{\alpha} . \tag{2.5}$$

Закон Ципфа визначає розподіл імовірності, якому підпорядкована залежність між кількістю міст *N(x)* і кількістю мешканців *x*; *N*, α - параметри розселення, властиві певній країні, регіону.

Закон Парето:

$$g(x) = g / x^{\beta} . \tag{2.6}$$

де ***g***, β - параметри економічного стану, властиві певній країні, регіону.

Це залежність між кількістю населення ***x*,** що володіє певними прибутками, і розміром цих прибутків ***g(x)****;*

Закон Бредфорда:

$$x(s) = A \ln s \tag{2.7}$$

Це залежність між кількістю населення, що користується даним словниковим запасом і відповідним обсягом словника ***s****;* ***A*** - параметр країни, регіону. Залежність може бути характеристикою розвитку культури, інтелекту.

Гравітаційна модель:

$$M_{ij} = \frac{P_i P_j}{d_{ij}^2} \tag{2.8}$$

Це показник взаємодії між регіонами *i* та *j;*

P_i - кількість населення і-го регіону;

d_{ij} - відстань між регіонами *i* та *j*.

Модель використовується для прогнозу соціальних і економічних взаємодій між регіонами.

Наведені залежності показують нелінійний (гіперболічний, логарифмічний) характер закономірностей, із якими зустрічаються дослідники в галузях науки та практики.

Закон Пуассона:

$$P_n = e^{-\delta} \frac{\delta^n}{n}, \qquad (2.9)$$

має різні застосування. У системі масового обслуговування може інтерпретуватися як імовірність зайнятості ***n*** приладів паралельного обслуговування.

$\delta = \alpha\lambda$ - показник завантаження системи,

де λ - інтенсивність вхідного (пуассонівського) потоку споживачів;

а - середній час обслуговування одного споживача.

Наприклад, при інтенсивності потоку в середньому два клієнти за хвилину $\lambda = 2, \alpha = 10 хв.$, показник завантаження системи становитиме: $\delta = 20$.

Формула (2.9) застосовувалася при визначенні залежності між пропускною спроможністю обслуговуючої системи «роздача» і кількістю місць у підприємстві громадського харчування. Вона забезпечує достатню точність наближення [12].

Таблиця відповідності необхідної кількості місць і завантаження системи має вигляд (при ймовірності недостачі місць < 0,001):

Показник завантаження, δ	0,1	1	2	3	5	10	15	20	25	30	40	50	60	70	80	90	100	110	120
Необхідна кількість місць, *n*	3	6	9	11	14	22	29	35	42	48	61	73	85	97	109	121	133	145	156

2.5 Контрольні питання

1. Визначте переваги штучної компетенції над людською.
2. Визначте переваги людської компетенції перед штучною.
3. Визначте для вирішення яких типів задач рекомендується застосовувати ЕС.
4. Наведіть критерії доцільності розробки ЕС.
5. Наведіть критерії виправданості розробки ЕС.
6. Наведіть критерії можливості розробки ЕС.
7. Основні властивості експертної системи.
8. Вкажіть критерії оцінювання рівня реалізації експертної системи.
9. Рекомендації щодо розробки експертної системи.
10. Приведіть приклади експертних закономірностей, що застосовуються у різних галузях людської діяльності.

3 ПОНЯТТЯ ЗНАНЬ. ВИКОРИСТАННЯ ЗНАНЬ В ЕКСПЕРТНИХ СИСТЕМАХ

3.1 Поняття знань

Головною відмінністю ЕС від інших програмних засобів є наявність бази знань, в якій зберігаються знання експертів про відповідну проблемну область. Знання зберігаються у вигляді сукупності записів деякою мовою подання знань, яка дозволяє легко змінювати та поповнювати базу знань в формі, яку розуміють спеціалісти. В традиційних же програмах знання жорстко „зашиті" в алгоритм і скорегувати їх може лише сам автор програми, тобто лише програміст (за умови, що він пам'ятає за яким принципом працює одна із множини створених ним програм).

До останнього часу саме різні мови подання знань (МПЗ) були центральною проблемою при розробці ЕС. Зараз існують десятки мов і моделей знань, серед яких найбільше поширення отримали продукції, семантичні сітки, фрейми, числення предикатів першого порядку і об'єктно-орієнтовані мови програмування. Вибір конкретної моделі визначається структурою знань в конкретній предметній області. Перед тим як обрати МПЗ і створити базу знань (БЗ) необхідно виявити цю структуру та виконати структурування знань.

Що ж таке знання? Загальне означення знань: «Перевірений практикою результат пізнання дійсності, правильне її відображення в мисленні людини». Однак таке означення не пояснює чим саме відрізняються програми, які основані на знаннях від традиційних прикладних програм.

Складність поняття знань полягає в множинності його матеріальних носіїв. За цією ознакою можна виділити п'ять основних форм знань:

Z1 - знання в пам'яті людини;

Z2 - матеріальні знання (підручники, довідники, статті і т. і.);

Z3 - поле знань (напівформалізований опис Z1 і Z2);

Z4 - знання мовою подання знань (формалізація Z3);

Z5 - база знань в комп'ютері (на машинних носіях інформації).

Наприклад, людина вирішує важкі і цікаві проблеми використовуючи не чіткі алгоритми, а виключно свій досвід в формі накопичених знань (Z1). Частину свого досвіду людство зберігає у вигляді книг чи заміток (Z2). Сукупність Z1 і Z2 створює знання про предметну область, які не пов'язані з машинною обробкою.

В ЕС використовують «екстракт» Z3 того, що вдалось отримати з Z1 і Z2. При цьому співвідношення емпіричного досвіду Z1 і його формалізованої теоретичної основи з книжок Z2, є різним для різних предметних областей. Вважають, що чим більша вага Z1 в процесі прийняття рішень, тим більш придатна ЕС для впровадження в дану область.

При створенні ЕС принциповим є етап розробки поля знань Z3. Поле знань – це деякий напівформалізований опис понять предметної області і зв'язків, що існують між ними. Наприклад у вигляді рисунка, таблиці, схеми, діаграми, мережі і т.д. В подальшому поле знань переписується деякою мовою подання знань, при цьому створюється модель знань Z4. Реалізація моделі за допомогою програмних засобів веде до виникнення п'ятої форми подання знань Z5 – бази знань.

Важливо пам'ятати, що не у будь-якій предметній області доцільно виділяти знання. Знання важливі там, де присутні розмиті означення, зміни понять, залежність ситуацій від множини контекстів, де присутня велика доля невизначеної та нечіткої інформації, тобто в областях де переважають емпіричні знання, де накопичення фактів випереджає розвиток теорії (медицина, геологія, юриспруденція, фінанси тощо). Такі ж добре структуровані області знань як математика, фізика, теоретична механіка мають добре розвинутий формальний апарат для опису своїх закономірностей, що дозволяє успішно проводити машинне моделювання з використанням традиційного алгоритмічного програмування (без виділення рівня знань).

Таким чином, приймемо в якості робочого, таке означення: **знання** – це основні закономірності предметної області, які дозволяють людині розв'язувати конкретні виробничі, наукові та інші задачі, тобто – факти, поняття, взаємозв'язки, оцінки, правила, евристики (або фактичні знання), а також стратегії прийняття рішень в цій області (стратегічні знання).

Чим же саме знання відрізняються від даних? Спробуємо виділити за аналогією п'ять форм даних:

D1 - результат спостереження об'єкта (наприклад, фіксація температури повітря на термометрі) або дані в пам'яті (наприклад, дата народження);

D2 - фіксація даних на матеріальному носії - таблиці, графіку, і т.і. (наприклад, таблиці Брадіса або дані про температуру протягом деякого інтервалу часу);

D3 - модель даних, деяка схема опису, яка пов'язує декілька об'єктів;

D4 - дані на мові опису даних;

D5 - база даних на машинному носії інформації.

Традиційно виділяють лише три рівні: зовнішній D1, логічний D3 та фізичний D5, які відповідають рівням знань Z1, Z3 і Z5.

Існує ще один аспект, який відображує відмінність даних і знань. Будь яке поняття, яке використовує людина, має два боки – екстенсіонал та інтенсіонал.

Екстенсіонал – це визначення поняття через перелік понять нижчого рівня ієрархії або фактів, що відносяться до означення, тобто це набір конкретних фактів, що відповідають даному поняттю.

Інтенсіонал – це визначення через поняття вищого рівня абстракції з вказуванням специфічних властивостей, тобто означення, або опис поняття через його властивості.

Наприклад, для поняття «поліклініка» екстенсіоналом буде перелік типу: «поліклініка №2», «дитяча поліклініка», «обласна поліклініка» і т. і. Інтенсіонал же в цьому випадку можна визначити так: «медична установа для надання амбулаторної допомоги за місцем проживання і роботи». Наприклад, для реляційної бази даних екстенсіональними поданнями є конкретні факти про предметну область (рядок таблиці, або його стовпець). Тобто екстенсіональні відношення створюють просторову структуру такої бази даних, а інтенсіональні - відповідають схемі бази даних, причому аргументами інтенсіонального відношення виступають атрибути, домени яких використовувались для створення відповідного екстенсіонального відношення.

Інтенсіонал відокремлює знання від даних, які завжди задаються екстенсіонально. Особливість інтенсіоналу полягає в його точності, виразності і правильно виділених властивостях. Наприклад, коли давньогрецький вчений Платон визначив інтенсіонал людини, як «тварина на двох ногах, яка не має пір'я», інший давньогрецький вчений Діоген спростував його означення, принесши обскубаного півня [13].

3.2 Властивості знань

Розібравшись в понятті знань, можна відзначити, що знання і дані мають багато спільного, але знання мають більш загальну структуру. Інколи знання називають добре структурованими даними або метаданими (тобто даними про дані). Можна сказати, що знання, це зазначення того, як використовувати дані. Крім того, знання поєднують в собі багато рис процедурної і декларативної інформації. Російський вчений в галузі штучного інтелекту Поспєлов визначив п'ять основних властивостей, які відрізняють знання від даних:

1. Внутрішня інтерпретовність.

Кожна інформаційна одиниця повинна мати унікальне ім'я, згідно з яким ЕС знаходить її, а також відповідає на запитання, які надсилаються на це ім'я. Коли дані в пам'яті ЕОМ були позбавлені імен, то була відсутня і можливість їх ідентифікації системою. При цьому система не мала інформації про те, що криється за тими або іншими двійковими кодами машинного слова і не мала змоги без участі програміста відповісти на запитання типу: «Що ти знаєш про механіка Іванова?». При використанні таких інформаційних одиниць, як знання, в пам'ять ЕОМ заноситься спеціальне машинне слово, в якому вказано в яких комірках зберігаються відомості про прізвища, роки народження, спеціальності тощо. В пам'ять заносяться перелік всіх відповідних об'єктів, які можуть відігравати роль імен для тих машинних слів, які відповідають рядкам таблиці.

2. Структурованість.

Знання мають гнучку структуру типу «матрьошки», забезпечуючи рекурсивну вкладеність одних інформаційних одиниць в інші. Тобто для знань характерна можливість встановлення відношень типу «частина - ціле», «рід - вид», «елемент - клас».

3. Зв’язність.

Можливість встановлення між інформаційними одиницями зв’язків різного типу, які можуть мати як декларативний («одночасно», «причина-наслідок»), так і процедурний характер («аргумент - функція»).

Перераховані три особливості знань дозволили створити найбільш загальну модель подання знань, яку називають семантичною мережею.

4. Семантична метрика.

Дане відношення характеризує ситуаційну відстань або силу асоціативного зв’язку між інформаційними одиницями. Його називають також відношенням релевантності для інформаційних одиниць. Таке відношення дозволяє виділяти в інформаційній базі деякі типові ситуації (наприклад, «закупка»). Відношення релевантності дозволяє знаходити знання близькі до тих, які вже знайдені.

5. Активність.

В традиційних системах всі інформаційні одиниці, які використовує комп’ютер, підрозділяють на команди і дані. При цьому всі процеси ініціалізуються командами, дані ж є пасивними і використовуються командами лише в разі необхідності. Інтелектуальні системи, як і людина, потребують для актуалізації своїх дій знання, які вміщені в них. Виконання будь-якої програми в ЕС викликається поточним станом інформаційної бази. При цьому джерелом активності може стати з’явлення в базі знань нових фактів, описів подій або встановлення нових зв’язків.

Таким чином, знання дозволяють співвіднести деякі елементи мови і їх значення в реальній дійсності. Властивість інтерпретування знань досягається за рахунок того, що змістовні аспекти даних, тобто їх семантика, зберігаються в пам'яті системи разом з іншими характеристиками даних [14]. Для зберігання даних використовуються бази даних (для них характерні великий об'єм і відносно невелика питома вартість інформації), для зберігання знань – бази знань (для них характерні невеликий об'єм і дуже дорогі інформаційні масиви).

3.3 Класифікація знань

Існує багато видів класифікації знань [15]. Розглянемо деякі з них.

За поняттям глибини знання розділяють на *глибинні* та *поверхневі* знання.

Глибинні знання вміщують абстракції, образи, аналогії, в яких відображається розуміння структури предметної області, а також призначення та взаємозв’язок окремих понять.

Поверхневі знання стосуються сукупності емпіричних асоціацій і причинно-наслідкових відношень між поняттями предметної області.

В більшості ЕС використовуються поверхневі знання, які дозволяють отримати достатньо якісні результати. Це ще пов'язано з тим, що на даний момент немає адекватних моделей, які дозволяють працювати з глибинними знаннями. Однак, введення глибинних знань дозволяє створювати більш потужні бази знань. Наприклад, некваліфікований лікар діє за принципом «якщо кашель - таблетки від кашлю, якщо ангіна - еритроміцин і т.і.». Досвідчений лікар обирає різні варіанти лікування хвороби в залежності, наприклад від стану хворого, його віку, наявності ліків в аптеці і т.і.

Глибинні знання створюються як результат узагальнення первинних понять про предметну область в деякі більш абстрактні структури, які часто не мають вербального опису. Наприклад, в медицині існує поняття синдрому, який використовують для узагальненого і поглибленого опису хвороби.

Глибинні знання пов’язані з типом мислення: емпіричним або теоретичним. При емпіричному мисленні пошук рішення задачі здійснюється лише на основі тих зв’язків, які вже є в ситуації задачі. При теоретичному мисленні пошук здійснюється шляхом конструювання нових зв’язків між елементами задачі. Тобто емпіричне мислення відображає об’єкт з боку його зовнішніх проявів та зв’язків (тобто на рівні поверхневих знань), а теоретичне – включає здібність до узагальнення і відображує внутрішні зв’язки об’єктів і закони їх розвитку (тобто глибинні знання).

За поняттям жорсткості знання розподіляють на *жорсткі* та *м’які*.

Жорсткі знання дозволяють отримувати однозначні рішення при заданих початкових умовах.

М’які знання дозволяють множинні, розпливчаті рішення і різні варіанти рекомендацій. Аналогічно підрозділяють і предметні області.

Загальною тенденцією інженерії знань є перехід від жорстких поверхневих знань до м’яких і глибинних.

За формою опису знання можна розділити на *процедурні* та *декларативні*. Історично первинними були процедурні знання, тобто знання, розподілені в алгоритмах. Вони керували даними. Для їх зміни вимагалося змінити програми. Проте, з розвитком систем ШІ пріоритет даних поступово змінювався і все більша частина знань зосереджувалась в структурах даних (таблиці, списки), тобто збільшувалася роль декларативних знань.

На теперішній час знання мають декларативну форму – це речення, записані мовами подання знань, близькими до природних і зрозумілими нефахівцям.

Також у будь-який момент часу в системі містяться три типи знань:

- структуровані знання - статичні знання про предметну область. Після того, як ці знання визначені, вони вже не змінюються;
- структуровані динамічні знання – змінні знання про предметну область. Обновляються у міру визначення нової інформації;
- робочі знання – знання, що використовуються для вирішення конкретної задачі або проведення консультації.

3.4 Контрольні питання

1. Поняття знань.
2. Визначте та проаналізуйте форми знань.
3. Визначте та проаналізуйте форми даних.
4. Порівняльна характеристика знань і даних.
5. Чи доцільно виділяти знання у будь-якій предметній області ?
6. Поняття інтенсіоналу та екстенсіоналу.
7. Основні властивості знань.
8. Класифікація знань за поняттям глибини.
9. Класифікація знань за поняттям жорсткості.
10. Класифікація знань за формою опису.

4 МЕТОДОЛОГІЯ ПРОЕКТУВАННЯ ЕКСПЕРТНИХ СИСТЕМ

4.1 Метод «швидкого прототипування»

Принципи проектування ЕС істотно відрізняються від принципів проектування традиційних програм. Такий стан справ обумовлюється неформалізованістю задач, відсутністю остаточно сформованої теорії і методології проектування ЕС. Наслідком цього стає необхідність модифікації самих принципів і засобів побудови ЕС безпосередньо в процесі їх проектування, по мірі того, як збільшуються знання розробників про предметну область (ПО). В зв'язку з цим найбільш популярним на даному етапі є метод «швидкого прототипування» ЕС, згідно з яким процес розробки ЕС складається з послідовної розробки декількох її прототипів.

Перший прототип є обмеженою версією ЕС, яка призначена продемонструвати життєздатність обраного підходу і забезпечити перевірку правильності кодування фактів, зв'язків і стратегій міркувань експерта. Вона дозволяє інженеру зі знань залучити експерта до активної участі в розробці ЕС. Обсяг прототипу як правило складає декілька десятків правил, фреймів або прикладів. Час розробки складає від 4-х до дванадцяти тижнів в залежності від вибраних інструментальних засобів (ІЗ) і досвіду розробника. В разі успіху першого прототипу експерт через інженера зі знань починає нарощувати знання прототипу про ПО. Таке нарощування може привести до того, що прототип розпочне успішно вирішувати всі потрібні задачі з даної ПО. Однак для перетворення його в кінцевий продукт як правило виконують перепрограмування ЕС на мові більш низького рівня, що дозволяє підвищити продуктивність і зменшити обсяги необхідної пам'яті [16].

4.2 Етапи проектування експертної системи

Процес проектування ЕС, в якому беруть участь: експерт, інженер зі знань та програміст, - можна відобразити за допомогою такого рисунку:

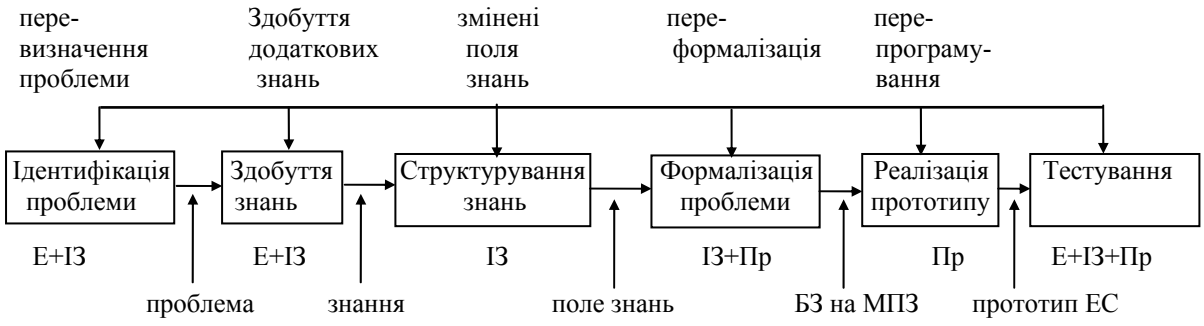

Рисунок 4.1 - Процес та етапи проектування ЕС

Ідентифікація проблеми

На цьому етапі експерт і інженер зі знань визначають найсуттєвіші особливості задачі. До них відносять уточнення самої задачі (її тип, широту постановки), визначення учасників розробки (припустимо, додаткових експертів), потрібні ресурси, цілі і задачі створення ЕС (наприклад, підвищити компетентність, розтиражувати здібності і навики, які рідко зустрічаються; підвищити якості прийманих рішень; автоматизувати рутинні аспекти роботи експерта і т.і.). Тут же уточнюються підпроблеми і ключові поняття ПО, вхідні дані, бажаний вигляд рішення, визначаються джерела знань. Найбільш складним є визначення задачі, широти її постановки, придатності для її вирішення методів ЕС. Тут можливі такі основні ситуації:

а) задачу доцільно вирішувати методами ЕС, але вона занадто важка і не може бути вирішена з використанням доступних ресурсів;

б) задачу можна вирішити методами ЕС, але це не надасть користувачу суттєвих переваг;

с) задача може бути вирішена, але ЕС буде працювати занадто повільно.

Критеріями, які підтверджують необхідність створення ЕС для вирішення конкретної задачі, можуть бути такі:

- недостатня кількість спеціалістів, які витрачають значний час на надання допомоги іншим робітникам;
- необхідність застосування для вирішення задачі багаточисельного колективу спеціалістів, оскільки жоден з них сам по собі не має достатніх знань;
- низька продуктивність в зв’язку з необхідністю постійного виконання повного аналізу складного набору умов, які звичайний спеціаліст не в змозі розглянути за виділений час;
- великі розбіжності між рішеннями, які приймають провідні і низькокваліфіковані виконавці;
- вузька спеціалізація задач;
- задача не є для експерта ні занадто важкою, ні занадто простою і потребує для свого вирішення відносно невеликого проміжку часу, наприклад, від трьох годин до трьох тижнів.

Припустимо, нам необхідно спроектувати медичну експертну систему. Перш за все, потрібно визначити цільове призначення системи і основні задачі, які вона буде розв’язувати. Цілі необхідно сформулювати точно, повно і несуперечно. Наприклад, для діагностичної системи це може бути відповідь на такі питання:

1. Чи здоровий пацієнт? Якщо ні, то яке саме у нього захворювання? Якщо захворювань декілька, то яке з них найбільш небезпечне?

2. Які зміни в дієті слід рекомендувати пацієнту і які з них найбільш важливі?

3. Які лабораторні дослідження необхідно виконати і які з них є першочерговими?

4. Як треба змінити образ життя пацієнта або кліматичні умови в яких він знаходиться?

5. Чи необхідно направити пацієнта на обстеження до вузькоспеціалізованих спеціалістів і якщо так, то до яких саме?

6. ... і т.д.

Після того, як цілі і задачі сформульовані, інженер зі знань повинен сформулювати підцілі, щоб побудувати ієрархічну структуру системи і розбити її на модулі. Для формулювання підцілей необхідно розбити задачу на підзадачі, які, в разі необхідності, можуть дробитись і далі. В нашому прикладі цільову задачу можна розбити на декілька підзадач, вирішення яких можна звести до таких питань:

1. Чи є порушення в системі кровообігу?

2. Чи є порушення в системі травлення?

3. Чи є порушення нервової системи?

4. ... і т.і.

Відзначимо, що всі ці підзадачі відносяться до першого питання верхнього рівня ієрархії.

Процес деталізації продовжується доти, доки не будуть виявлені факти, що містять безпосередні результати вимірювань або лабораторних аналізів, тобто початкові дані поставленої задачі [17]. Наприклад,

- результати хімічного аналізу крові;
- наявні ознаки і симптоми захворювань;
- кліматичні умови і спадкові фактори і т.і.

Важливими завданнями етапу ідентифікації є також кадрові питання і чітке визначення потрібних і наявних ресурсів. Тривалість етапу складає в середньому один-два тижні.

Здобуття знань

Зміст етапу полягає в перенесенні компетентності експерта на інженера зі знань. Це найбільш трудомісткий і складний етап, протягом якого інженер зі знань шляхом аналізу текстів, діалогу, експертних ігор, дискусій, інтерв'ю, спостережень та інших засобів отримує необхідні знання від експерта. Тривалість етапу складає від одного до трьох місяців і більше. Після завершення етапу інженер зі знань повинен мати достатньо повне уявлення про ПО і способи прийняття рішень в ній. На даному етапі бажано працювати лише з одним експертом, оскільки спілкуючись з декількома експертами, інженер зі знань ризикує втратити можливість глибоко дослідити дану область [18]. Крім того, спостерігаючи за процесом рішення експертом конкретних задач, не слід створювати занадто специфічні правила. Необхідно виявляти найбільш загальні принципи, якими користується експерт, вводити поняття високого рівня, які експерт фактично використовує, хоча і не називає їх, оскільки вони

можуть бути не вербалізованими (відсутніми в літературі). Введення таких понять може дозволити замінити декілька специфічних правил одним більш загальним.

Структурування знань

Визначається структура отриманих знань, тобто визначається перелік основних об'єктів, понять і атрибутів, відношень між поняттями, структура вхідної і вихідної інформації, стратегія прийняття рішень, обмеження стратегії і т.і.

Головним на стадії концептуалізації є визначення рівня деталізації системи. Тут необхідно створити ієрархічну систему знань і уявлень експерта у вигляді чіткої «піраміди знань». Першим кроком концептуалізації є виявлення основних об'єктів предметної області. Наприклад, «захворювання», «аналіз», «технологічний процес», «пристрій» і т.і. Далі для кожного об'єкта визначається набір характерних ознак або атрибутів. Наприклад, «колір», «швидкість» і т.і. Після цього для кожної ознаки визначається область її значень, яку може бути задано:

а) алфавітом значень. Наприклад, ознака - «колір світлофора», задається алфавітом «зелений, червоний, жовтий»;

б) інтервалом значень. Наприклад, ознака «опромінення» може мати значення від 0,5 до 400 мР/год;

в) шкалою впевненості. Наприклад, ознака «висока температура» із значеннями «36.6/0; 37.0/0,2; 37.6/0,4; 38.0//0,6; 39.0/ 0.8; 40.0/1,0»;

г) логічною функцією істинності. Наприклад, ознака «горло червоне» – значення «так/ні».

Після виявлення областей означення для всіх ознак робиться спроба агрегування та дезагрегування, тобто групи близьких об'єктів об'єднуються в об'єкти більш високого рівня абстракції, а ознаки, які можна розбити на більш дрібні, – уточнюються і деталізуються.

Сукупність виділених понять повинна бути точною, повною і несуперечливою. Наприклад, які саме аналізи потрібно провести? Чи треба звернутись до історії захворювання пацієнта і якщо так, то які дані будуть найбільш важливими? Які ще відомості про пацієнта можуть бути корисними (наприклад, чи спостерігались спадкові захворювання у родичів хворого)? Чи треба враховувати ліки, які хворий приймав раніше, а також попередні призначення лікарів? Чи мають значення образ життя та кліматичні умови і режим харчування? Які симптоми наявні у хворого (головний біль, висока температура і т.і.).

Після цього необхідно визначити основні відношення між об'єктами та поняттями ПО. Для цього необхідно виявити кінцеві концепти або висновки. Їх кількість як правило обмежена (від 3-5 до 100-200) в ЕС діагностики і до 500 в ЕС інтерпретації даних. При цьому в кінцевих рекомендаціях вони можуть по різному поєднуватися, породжуючи велику кількість можливостей. Об'єкти, ознаки і їх значення поєднуються

системою залежностей для виявлення правильних висновків. Необхідно намагатись виявити найбільшу кількість зв'язків, а в ідеалі всі зв'язки, що існують в ПО, оскільки від цього залежить якість роботи механізму виведення.

Крім того, дуже важливо чітко уявити специфічні особливості ПО, які створюють труднощі при вирішенні задач. В нашому прикладі вони можуть полягати в тому, що:

- деякі з ознак (результати або симптоми) перекриваються іншими;
- окремі ознаки змінюються стрибкоподібно, їх значення залежать від періоду спостерігання або зовнішніх умов;
- можливі неправильні показання лабораторного обладнання;
- деякі ознаки недоступні або потребують великих витрат;
- можуть бути враховані не всі взаємозв'язки між ознаками.

Дуже важливою складовою етапу структурування знань є вибір способу організації виділених об'єктів ПО і визначення зв'язків між ними в термінах деякої моделі знань. Окремі ПО, наприклад біологія, самі по собі мають класифікаційну структуру, оскільки всі об'єкти пов'язані родовидовими відношеннями. Однак таких областей не багато, тому в більшості випадків всю множину продукцій розбивають на деякі споріднені класи. В приведеному прикладі, можна виділити групу правил, які пов'язані з першою підціллю, тобто дозволяють дати відповідь на питання про виявлення порушень в системі кровообігу. З підціллю наступного рівня буде пов'язана інша група правил меншого розміру.

Взагалі в ПО можна виділити ієрархії трьох типів:

- структурну,
- причинно-наслідкову;
- функціональну.

Наприклад, для структурної ієрархії комп'ютера, її вершина може бути зображена безпосередньо самим комп'ютером. Наступний рівень може бути зображено набором скомпонованих блоків. Третій рівень створюється платами, з яких складається кожен блок. Четвертий рівень описується такими складовими, як мікросхеми. Використання такої ієрархії при пошуку несправності не потребує від ремонтника глибоких знань про будову комп'ютера та функції блоків. Знання, на основі яких може бути здійснено такий пошук, відносяться до поверхневих.

Згідно з причинно-наслідковою ієрархєю функціонування комп'ютера описується в термінах «причина-наслідок». Наприклад, на найвищому рівні ієрархії натиснення клавіші на клавіатурі має привести до відображення символу на екрані дисплея або його виведення на друк. Якщо цього не сталось, необхідно спуститись на більш низький рівень і перевірити працездатність принтера. Якщо він не працює, можна подивитись, чи світиться лампочка на його робочій панелі. Якщо ні, можна припустити, що відсутня напруга, пошкоджений кабель або вийшов з ладу запобіжник. Ясна річ, що знання про поведінку системи є більш

змістовними, ніж знання про її структуру, але все ж і вони достатньо поверхневі відносно такої складної системи, якою є комп'ютер, а тим більше – людина.

На найвищому рівні абстракції комп'ютер розглядається як ієрархія модулів або підсистем, які виконують певні функції. Наприклад, функція комп'ютера в цілому, яка розміщується на вершині такої ієрархії, полягає в перетворенні вхідних даних в потрібні вихідні за допомогою заданих алгоритмів та евристик (якщо комп'ютер має систему, основану на знаннях). На наступному рівні такої ієрархії буде розташована ціла множина таких функцій, як введення даних, обробка, запис в оперативну пам'ять, виведення. Якщо не виконується основна функція комп'ютера (не виводяться результати обчислень або виведені дані є помилковими), необхідно спуститись на наступний рівень ієрархії, спробувати з'ясувати, яка з його функцій не спрацювала. Припустимо, що не вдається записати дані у зовнішню пам'ять, тобто не працює дисковод. Тоді треба перейти на той рівень ієрархії, на якому знаходиться опис дисковода. Наприклад, можливо не працює двигун, який обертає диск.

Спробуємо побудувати таку ж ієрархію в області медицини. Структурна ієрархія може бути введена при розгляді фізичної будови тіла людини. До неї можна включити такі рівні: рівень організму в цілому; окремі органи; тканини, клітини; молекули і гени. В функціональній ієрархії можна виділити рівень організму в цілому, далі – рівень підсистем організму (м'язова і нервова системи, система кровообігу і т.і.). Дроблення здійснюється до отримання опису потрібного ступеня детальності. Таким чином, зміст бази знань визначається тим, які об'єкти, взаємозв'язки між ними і види ієрархій виділені в ПО. Використаємо в нашому прикладі причинно-наслідкову ієрархію, оскільки в більшості присутніх в неї пар атрибут-значення містяться назви хімічних елементів і симптомів захворювань.

Треба підкреслити, що якісна база знань повинна містити не лише поверхневі, а й глибинні знання, які відображають найбільш загальні принципи, у відповідності до яких йде розвиток всіх процесів в ПО, а також властивості цих процесів. Дійсно, припустимо, що Ви працюєте разом з лікарем над створенням експертної системи. Ви були присутні на бесідах лікаря з хворими, виявили зв'язок між результатами лабораторних аналізів, скаргами пацієнтів і різними видами захворювань. Якщо вас не цікавить шлях, яким лікар прийшов до того або іншого висновку, Ви лише механістично включите в продукційні правила вихідні дані і висновки. Це дасть змогу системі приймати однакові рішення в однакових ситуаціях. Однак застосовувати таку систему в медицині було б небезпечно, оскільки в ній відсутні будь-які загальнотеоретичні знання і не відображено практичний досвід лікаря. Абсолютно ж ідентичні ситуації в реальному житті зустрічаються вкрай рідко (а в найбільш загальному сенсі – не зустрічаються ніколи). Можна сказати, що такий самий ефект Ви

отримуєте при створенні ЕС для гри в шахи, якщо спостерігаючи дії досвідченого гравця будете запам'ятовувати лише ті рішення, які він приймає при наявності тієї або іншої комбінації фігур, щоб в подальшому відтворити їх в комп'ютері. Такі знання також будуть поверхневими і неважко уявити, чим закінчиться використання такої ЕС. Тобто і в цьому випадку необхідно виявити глибинні знання, які дозволяють гравцю приймати правильні рішення. Обмежуватись лише поверхневими знаннями можна лише в тих ПО, де глибинні знання до теперішнього часу не виявлені. Але при цьому треба дуже ретельно продумати, чи надасть використання такої ЕС практичну користь.

Середня тривалість етапу складає 2-4 тижні.

Формалізація знань

На основі отриманої структури поля знань інженер зі знань обирає найбільш адекватну з існуючих моделей формалізації (або моделей подання знань), враховуючи при цьому наявність відповідних інструментальних засобів (ІЗ). В процесі формалізації треба враховувати три важливих фактори: простір гіпотез (модель бази); модель процесу виведення (прийняття рішень); характеристику даних.

Модель бази або простір гіпотез подають частіше у вигляді однієї з таких форм:

- логічні моделі (наприклад, числення предикатів першого порядку);
- продукційні правила;
- семантичні мережі;
- фрейми (об'єктно-орієнтоване подання).

Перші два типи моделей дозволяють створювати модульні бази знань, які доцільно використовувати, коли між фрагментами знань відсутні явні зв'язки. Це гнучкі моделі для слабоструктурованих областей, вони просто змінюються.

Семантичні мережі і фрейми більш жорсткі, негнучкі, але більш адекватно відображають глибинні знання і поняття зі складною структурою.

В процесі формалізації слід враховувати такі фактори:

1. Якщо поле знань характеризує глибинні знання, слід використовувати мережні моделі, якщо поверхневі – модульні.
2. Логіка добре працює лише на добре структурованих ПО (математика, аналітична хімія і т.і.).
3. Якщо ієрархічна структура поля знань містить моделі різних рівнів абстракції, слід використовувати багаторівневі моделі формалізації. Не слід за будь-яку ціну прагнути створити однорідну базу.
4. Якщо структура поля знань суперечить наявним ІЗ, переходьте до самостійного створення оболонки на мовах штучного інтелекту, але не відмовляйтесь від розробленого поля знань.
5. Оскільки прототип частіше за все відкидається на стадії тестування не слід безмірно нарощувати обсяг бази знань.

Відносно моделі знань треба впевнитись чи буде вона мати поведінковий характер, чи математичний. В першому випадку постає проблема спрощення механізму виведення, в другому ж важливо здобути всю додаткову інформацію з універсальної математичної моделі і ввести її безпосередньо до ЕС.

В процесі формалізації, при виборі моделі подання знань, важливо врахувати також і характер даних (достовірність, невизначеність, форму подання, постійність у часі і т.і.).

В разі, коли окремі підзадачі суттєво відрізняються за формами простору гіпотез і моделями виведення, слід обрати такий формалізм, який привнесе мінімальний негативний ефект.

Слід також бути готовим до того, що, оскільки всі існуючі моделі подання знань занадто жорсткі для відображення реального світу, мови введення і виведення інформації занадто стилізовані і стандартний набір механізмів виведення є досить обмеженим, то для кожної ПО фактично потрібно будувати свою власну модель подання.

Якщо прийнято рішення про використання для формального подання знань існуючих ІЗ, то при їх виборі треба врахувати такі зауваження:

а) щоб уникнути ситуації, коли інженер зі знань обирає ті ІЗ які він краще знає, а не ті які більш підходять для реалізації ЕС, треба провести опитування з питання вибору ІЗ з кількох незалежних ІЗ;

б) намагатись уникнути вибору в якості ІЗ мов програмування низького рівня (Паскаль, Сі, Фортран), оскільки це значно підвищить термін виконання робіт;

в) в разі виявлення невідповідності ІЗ задачі, відкидати його і обирати інший, або створювати власну оболонку, але ні в якому випадку не пристосовувати надбанні знання до можливостей ІЗ.

Етап формалізації є достатньо складним і відповідальним і потребує не менше як один - два місяці.

Реалізація прототипу

Задачею етапу є розробка програмного комплексу, який би довів життєздатність обраного підходу. Важливим принципом виконання цього етапу є неприпустимість відкладення процесу здобуття знань до завершення програмування, оскільки це найбільш трудомісткий процес, по мірі виконання якого, до того ж, приходиться вносити зміни у всі компоненти системи. Тобто, необхідно розпочинати роботу з найпростішими знаннями одразу, як це стане можливим, і при першій же можливості випробувати створені управляючі структури.

Необхідно також уважно стежити за тим, щоб в процесі створення ЕС знання про область експертизи не змішувались із загальними знаннями, що в подальшому значно ускладнює або робить взагалі неможливим модифікацію і розвиток системи. При створенні першого прототипу ЕС (ЕС-1) не звертають уваги на підсистеми розуміння і синтезу природної мови, врахування складних часових та причинно-наслідкових відношень,

моделювання міркувань з нечіткими поняттями і т.і. Програмування ЕС-1 треба проводити швидко, оскільки вона лише повинна підтвердити правильність попередніх проектних рішень і в подальшому скоріше за все програма буде перероблена.

Задачі аналізу функціонування ЕС при розширені бази знань і кола вирішуваних задач, а також урахування вимог користувача і вдосконалення систем введення-виведення здійснюється при розробці прототипів ЕС-2 і ЕС-3.

Загальний термін реалізації прототипів може складати від одного-двох місяців і більше.

Тестування

Полягає в оцінюванні якості роботи і корисності програми-прототипу, а також у висновках про необхідність її переробки. За результатами етапу ЕС повинна бути приведена у відповідність до вимог користувача. Тут перевіряються зручність та адекватність інтерфейсу введення-виведення, ефективність стратегій управління, ефективність бази знань. Можна виділити такі особливості тестування різних прототипів ЕС:

- оцінювання ЕС-1 здійснює експерт, якого цікавить повнота і безпомилковість правил виведення;
- оцінювання ЕС-2 здійснює інженер зі знань з точки зору ефективності системи;
- оцінювання ЕС-3 надає користувач, якого цікавлять зручність роботи і отримання практичної вигоди.

Основні проблеми, які виникають на цьому етапі, пов’язані перш за все із незручністю вхідної мови; вибором неправильних питань або недостатньою кількістю інформації в базі знань; недоврахуванням взаємозалежностей, які існують між правилами; помилковістю, суперечністю та неповнотою правил. Термін виконання етапу складає один-два тижні.

4.4 Контрольні питання

1. Основні положення методу «швидкого прототипування».
2. Процес та етапи проектування ЕС: загальна характеристика.
3. Ідентифікація проблеми як етап проектування ЕС.
4. Здобуття знань як етап проектування ЕС.
5. Структурування знань як етап проектування ЕС.
6. Процес агрегування та дезагрегування.
7. Види структурних ієрархій предметної області.
8. Формалізація проблеми як етап проектування ЕС.
9. Реалізація прототипу як етап проектування ЕС.
10. Тестування як етап проектування ЕС.

5 ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ ЕКСПЕРТНИХ СИСТЕМ

5.1 Загальна характеристика інструментальних засобів

При розробленні експертних систем використовується спеціальний інструментарій, який дозволяє значно скоротити час розробки. Такий інструментарій включає як програмні, так і апаратні засоби. До апаратних засобів відносяться: ПЕОМ, інтелектуальні робочі станції, послідовні символьні ЕОМ (ЛІСП і ПРОЛОГ машини), ЕОМ загального призначення та паралельні символьні ЕОМ. Крім того, для розширення можливостей числових процесорів всіх типів випускаються спеціальні символьні співпроцесори.

Загальну класифікацію програмного інструментарію можна показати так:

1) процедурні мови, орієнтовані на обробку символьної інформації (наприклад, ЛІСП і т.і.);

2) мови інженерії знань, тобто мови високого рівня, які орієнтовані на розробку ЕС (наприклад, PROLOG, OPS-5);

3) засоби автоматизації процесів конструювання, використання та модифікації ЕС (наприклад, RLL, HEARSAY-4 і т.і);

4) пусті (базові) ЕС або „оболонки”, які не містять знань ні про яку ПО (наприклад, CODEX, EMYCIN, DECTOOLS і т.і).

В наведеній вище класифікації інструментальні засоби (ІЗ) розташовані в порядку зменшення трудовитрат, необхідних при створенні за їх допомогою конкретних ЕС. При використанні інструментарію першого типу програміст вимушений самостійно програмувати всі компоненти ЕС на мові достатньо низького рівня. На другому рівні продуктивність різко зростає, однак за рахунок деякого падіння ефективності. ІЗ третього рівня дозволяють розробнику ЕС не розробляти всі або деякі компоненти ЕС, а обирати їх із заздалегідь сформованого набору. При використанні ІЗ четвертого рівня розробник повністю звільняється від роботи зі створення програм, оскільки він має в своєму розпорядженні пусту систему, яку необхідно наповнити знаннями з відповідної ПО [19].

Разом з тим, при використанні ІЗ третього та четвертого типів виникає і ряд проблем:

- реалізовані в них стратегії управління виведенням можуть не відповідати методам вирішення, які використовує експерт, наслідком чого можуть стати неефективні або взагалі неправильні рішення;
- мова подання знань може не підійти для даної ПО.

В наш час дуже широке розповсюдження отримали ІЗ, які називають оболонками, що налагоджуються. Вони дозволяють використовувати оболонку не в деякому «застиглому» вигляді, а генерувати її з деякої

множини механізмів, які передбачені в ЕС (наприклад, ЕКО, ЕКСПЕРТИЗА, НЕКС). В останній час розмежовують оболонки, які отримали назву "оболонок застосування" від тих оболонок, які призначені для здобуття знань. Кім того стали виділяти універсальні та спеціалізовані оболонки.

Таким чином, ми розглядатимемо інструментальні засоби побудови ЕС в якості систем програмування, які спрощують проектування ЕС. Відповідно, їх склад і структура визначаються особливостями вирішуваних експертними системами задач і технологій проектування ЕС. Необхідно відзначити, що вибір технології і інструментальних засобів реалізації ЕС – це ключове питання створення ЕС.

Відповідно до наведеної вище загальної класифікації, розглянемо більш детально особливості програмного інструментарію, що використовується для розробки ЕС.

Традиційні мови програмування

До цієї групи інструментальних засобів входять традиційні мови програмування (C, C++, Pascal, Basic, та ін.), орієнтовані в основному на чисельні алгоритми, і які не дуже ефективні для роботи з символьними і логічними даними.

Побудова ЕС на основі цих мов вимагає від програмістів великої і віртуозної роботи. Проте великою перевагою цих мов є висока ефективність, обумовлена близькістю до традиційної машинної архітектури. Крім того, використання традиційних мов програмування дозволяє включати інтегровані ЕС у великі програмні комплекси загального призначення. Найзручнішими в цьому плані є об'єктно-орієнтовані мови, зокрема, C++. Це пов'язано з тим, що парадигма об'єктно-орієнтованого програмування тісно пов'язана з фреймовою моделлю подання знань [20].

Мови штучного інтелекту

Один з напрямів еволюції систем програмування пов'язаний з функціональними мовами. Перша широко поширена мова цього типу розроблена в 1959 році і одержала назву ЛІСП (LISP – List Processing Language). Вона орієнтована на вирішення задач обробки символьної інформації.

Розвиток функціональних мов йшов по шляху вбудовування в них механізмів роботи з базами знань. Прикладом такої мови може служити ПРОЛОГ (PROLOG – Programming Language Based on Logic), розроблений в 1971 році і призначений для роботи з логічними базами знань. Універсальність цих мов невелика, проте, її втрату вони компенсують багатими можливостями під час роботи з символьною і логічною інформацією.

Спеціалізований програмний інструментарій

До цієї групи програмних засобів ШІ входять бібліотеки і надбудови над мовами ШІ (наприклад, ЛІСП), які дозволяють користувачам

працювати із заготовками ЕС на вищому рівні, ніж це можливо в звичних мовах ШІ.

"Оболонки"

Під "оболонками" (Shells) розуміють "порожні версії" існуючих ЕС, тобто готові ЕС без розробленої бази знань. Перевага оболонок у тому, що вони взагалі не вимагають роботи програмістів для створення готової експертної системи. Потрібні тільки фахівці в предметній області для заповнення БЗ. Проте, якщо деяка предметна область погано вкладається в модель, яка використовується в деякій оболонці, заповнити БЗ в цьому випадку досить не просто або майже неможливо.

Деякі вчені в галузі ШІ приводять ще одну класифікацію ІЗ. На їх думку, інструментальні засоби умовно можна розділити на такі групи:

1. Засоби організації даних і знань.
2. Засоби підтримки.
3. Сервісні засоби.
4. Комплексні засоби.
5. Інтегровані засоби.

Засоби організації даних і знань – це символьні мови програмування, не процедурні мови програмування, мови інженерії знань (моделі подання знань), інформаційно-пошукові системи, системи управління базами даних (СУБД).

Засоби підтримки – це асемблери і макроасемблери, компілятори, інтерпретатори, налагоджувачі і т.д.

Сервісні засоби – це редактори, ділові ігри і навчальні системи. Вони мають обслуговуючий, допоміжний характер.

Комплексні засоби – СУБД з вбудованою мовою програмування інтерпретуючого і/або компілюючого типу, системи обробки даних з оточенням, спеціалізовані засоби обробки даних.

Інтегровані засоби – інтегровані прикладні системи, "порожні" або інструментальні ЕС (оболонки ЕС), середовища підтримки. Це найскладніший і найефективніший інструмент для реалізації практично будь-яких додатків ЕС. Основними компонентами цих систем є текстова і таблична обробка, засоби управління базами даних, розвинені графічні засоби. Такі системи надають користувачу власне операційне середовище, що дозволяє вирішувати задачі, не використовуючи безпосередньо операційну систему.

Оболонки ЕС, що підтримують відомі моделі подання знань – продукції, фрейми, семантичні мережі, дерева, таблиці і т.д. ефективно використовуються для створення прикладних ЕС.

Нарешті, наймогутнішим інструментальним засобом є так звані середовища підтримки, що об'єднують засоби автоматизації всіх основних етапів створення ЕС [21]. Найбільша кількість середовищ створюється на основі поняття циклу життя програмного забезпечення. Під циклом життя

розуміється послідовність технологічних етапів створення, експлуатації і розвитку програмного забезпечення.

5.2 Стадії розробки експертних систем і інструментарію

Після завершення розробки першого прототипу, експерт і інженер зі знань мають нагоду оцінити, що саме буде включено в розробку остаточного варіанта системи. Для цього іноді необхідно виділити додаткові етапи (стадії існування) при переході ЕС від прототипу до промислового зразка:

- демонстраційний прототип,
- дослідницький прототип,
- діючий прототип,
- промислова система,
- комерційна система.

Найчастіше реалізується плавний перехід від демонстраційного прототипу до промислової системи.

Характеристики прототипів

Демонстраційний прототип: система вирішує частину задачі, демонструючи життєздатність підходу (декілька десятків правил або понять).

Дослідницький прототип: система вирішує більшість задач, але нестійка в роботі і не повністю перевірена (декілька сотень правил або понять).

Діючий прототип: система надійно вирішує всі задачі на реальних прикладах, але для складної задачі вимагає багато часу і обсягів пам'яті.

Промислова система: система забезпечує високу якість рішень при мінімізації необхідного часу і пам'яті; переписується з використанням ефективніших засобів подання знань.

Комерційна система: промислова система, придатна до продажу, добре документована і забезпечена сервісом.

Таким чином, на третьому етапі роботи створюється "демонстраційний" прототип. Доцільно об'єднати його з дослідницьким, щоб відпрацювати більшу кількість правил. Інженер зі знань спільно з експертом заповнює БЗ системи значною кількістю правил.

Звичайно вважається, що для "демонстраційного прототипу" необхідний обсяг БЗ порядку сотні правил. При заповненні такої бази проводиться постійний контроль знань, що неперервно заносяться в систему, на несуперечність з уже існуючими знаннями за допомогою засобів, які є в робочій БЗ, або вручну. Отже, основне навантаження на третьому етапі несе інженер зі знань. Також на цьому етапі перевіряється успішність вибору експерта.

Про ступінь кваліфікації експерта можна судити за відсотком банальних висновків системи в загальному обсязі одержаних рішень.

В процесі роботи можуть виникнути складнощі з формалізацією одержаних знань відповідно до вибраної структури БЗ. Всі ці моменти фіксуються, але робота повинна продовжуватися з використанням вибраних засобів розробки ЕС. Тільки у разі виникнення конфлікту між можливостями вибраних засобів з фундаментальними властивостями знань, який неможливо ліквідовувати без істотних порушень точності роботи ЕС, необхідно повернутися до вибору нового ІЗ.

В процесі заповнення БЗ на третьому етапі необхідно постійно проводити перевірку працездатності системи на тестових прикладах.

Основний критерій цих перевірок – не погіршення якості вирішення тестових задач з введенням нового фрагмента знань. До важливої складової частини тестових прикладів відноситься постійний контроль "дружності" інтерфейсу системи. Система повинна забезпечувати користувачу можливість легким і природним способом запитувати про незрозуміле, припиняти роботу.

Паралель з процесом заповнення бази, а часто і залежно від цього, відбувається формування підсистеми пояснень. Постійний контроль інтерфейсу дозволить розвивати цю підсистему в потрібному напрямі. Фіксація складнощів, які виникають при реалізації підсистеми пояснень, дозволить усунути обмеження, продиктовані можливостями інструментальних засобів.

Основною метою створення "демонстраційного прототипу" служить показ можливості створення ЕС для роботи в даній предметній області на множині конкретних прикладів.

Важливою задачею, також вирішуваною в рамках демонстраційного прототипу, є відлагодження інтерфейсу системи з користувачем. Визначені в ході роботи складнощі з реалізацією тієї або іншої частини ЕС, зафіксовані в протоколах, служать корисною інформацією для наступного етапу роботи над системою.

Якщо при реалізації "демонстраційного прототипу" з'ясувалася невисока ефективність системи, пов'язана з обмеженнями інструментальних засобів, то відбувається повернення до другого етапу і потім створення нового "демонстраційного прототипу" за допомогою інструментальних засобів нижчого рівня, як правило, на універсальній мові програмування.

На третьому етапі основне навантаження здійснюється на інженерів-програмістів, що здійснюють трансляцію готової системи з мови одного інструментального засобу на іншу мову. Після відлагодження нового "демонстраційного прототипу" відбувається перехід до наступного етапу.

На четвертому етапі здійснюється розробка "діючого прототипу". Розробка проводиться аналогічно третьому етапу наповнення БЗ і підсистеми пояснень. Важлива особливість етапу – пробна експлуатація

системи на реальних задачах протягом півроку. Результати апробації постійно аналізуються з участю експерта. В результаті зміст БЗ повинен дозволяти вирішувати будь-яку задачу з досліджуваної предметної області.

На цьому етапі більшість експертів бере ініціативу в свої руки і здійснює контроль за системою для уточнення, детальної розробки і обслуговування.

На завершальному етапі створюється "промислова система". "Діючий прототип" доводиться до деякого рівня, який і експерти, і користувачі вважають достатнім. При цьому здійснюється оцінювання системи: проводиться її тестування відносно критеріїв ефективності. ЕС оцінюється головним чином для того, щоб перевірити точність роботи програми і її корисність.

Критерії можуть бути згруповані таким чином:

- критерії користувачів (зрозумілість і прозорість роботи системи, зручність інтерфейсу);
- критерії запрошених експертів (оцінка порад-рішень, що пропонуються системою, порівняння її з власними рішеннями, оцінка підсистеми пояснень);
- критерії колективу розробників (ефективність реалізації; продуктивність; час відгуку; дизайн; широта обхвату предметної області; несуперечність БЗ; кількість тупикових ситуацій, коли система не може ухвалити рішення; аналіз чутливості програми до незначних змін в поданні знань і т.д.).

Якщо за всіма критеріями одержані задовільні відповіді, на промислову систему оформляється вся необхідна документація. В нашій країні поняття "комерційної системи" входить в поняття "промисловий програмний продукт або промислова ЕС".

"Комерційна" версія системи захищається від несанкціонованого копіювання і доступу. Встановлюється ціна програмного продукту і проводяться заходи щодо реклами.

При такому підході до розробки ЕС, колектив розробників повинен складатися з 4-6 чоловік, до складу яких входять:

- керівник проекту, що здійснює наукове і адміністративне керівництво проектом;
- один-два інженери–програмісти, що володіють мовами програмування і орієнтуються в області інструментальних засобів і "оболонок" ЕС;
- один-два інженери зі знань, що також мають уявлення про роботу з інструментальними засобами і деякі знання в області психології спілкування;
- експерт-фахівець в конкретній предметній області, здатний чітко сформулювати свої знання і описати процес прийняття рішення.

Подібно до класифікації стадій існування ЕС, програмні інструментальні засоби для їх створення також поділяють на три великі групи:

1. Експериментальні системи. Створюються для розв'язання вузькоспеціальних задач і не призначені для використання в інших задачах. Працюють дуже повільно і неефективно, але й потребують відносно невеликих зусиль на своє створення.

2. Дослідницькі системи. Такі системи вже прискіпливо перевірені та супроводжуються розробниками. Як правило вони теж працюють досить повільно і не дуже ефективно. Їх використовують при розробці ЕС на стадіях демонстраційного, дослідницького та іноді діючого прототипів.

3. Комерційні системи. Вища стадія існування інструментальних систем. Такі системи всебічно і досконало перевірені, добре підтримуються розробниками, мають високу продуктивність і зручний інтерфейс. За їх допомогою можна створювати промислові та комерційні ЕС.

Слід пам'ятати, що разом із правильним вибором широти охоплення задачі, правильний вибір ІЗ є найбільш відповідальним рішенням, яке необхідно приймати при розробці ЕС. Проблема тут полягає в тому, що більшість ІЗ розроблялись під конкретні задачі, а деякі з них взагалі з'явились шляхом усунення з відповідних ЕС предметних знань. Багато сучасних комерційних ІЗ містять останні досягнення в галузі штучного інтелекту, але ніхто й досі не може чітко й однозначно відповісти, які характеристики ІЗ потрібні для створення ЕС, спрямованих на вирішення конкретних класів задач. Тобто, для кожного ІЗ є задача, яку вони можуть успішно вирішити, але, на жаль, зворотна вимога не має сили. Як показує досвід, для будь-якої задачі можна знайти декілька однаково придатних ІЗ, але жоден з них не буде повністю задовольняти всі вимоги задачі.

5.3 Контрольні питання

1. Поняття інструментальних засобів побудови експертних систем.
2. Апаратні та програмні інструментальні засоби.
3. Загальна класифікація програмного інструментарію.
4. Характеристика основних видів програмного інструментарію.
5. Інтегровані засоби програмного інструментарію.
6. Стадії розробки експертних систем.
7. Взаємозв'язок та послідовність етапів розробки експертних систем.
8. Процес та основні критерії оцінювання промислової ЕС.
9. Стадії розробки інструментальних засобів.
10. Універсальність інструментальних засобів для розробки ЕС.

6 ТЕОРЕТИЧНИЙ АНАЛІЗ ПРОЦЕСУ ЗДОБУТТЯ ЗНАНЬ

6.1 Поняття та аналіз процесу здобуття знань

Наведемо означення терміна здобуття знань (knowledge acquisition), дане відомим фахівцем в галузі штучного інтелекту Бучананом:

"*здобуття знань* – це передача потенційного досвіду вирішення проблем, отриманого від деякого джерела знань, і перетворення його до вигляду, який надасть можливість використовувати його у програмі" [22].

Необхідно відзначити, що термін „здобуття знань" має узагальнений характер і зовсім нейтральний до способу передачі знань. Наприклад, передача може здійснюватися за допомогою спеціальної програми, що у процесі обробки великого масиву історій хвороби встановлює зв'язок між симптомами і захворюваннями. А от термін виявлення знань (knowledge elicitation) відноситься саме до одного зі способів передачі знань – опитування експертів у певній проблемній області, що виконується аналітиком чи інженером зі знань. Останній потім створює комп'ютерну програму, що подає такі знання (чи доручає це кому-небудь іншому, забезпечуючи його всією необхідною інформацією) [23].

Цей же термін застосовується і для позначення процесу взаємодії експерта зі спеціальною програмою, метою якого є:

- отримати будь-яким систематичним способом знання, якими володіє експерт, наприклад, пропонуючи експерту репрезентативні задачі і фіксуючи пропоновані способи їх вирішення;
- зберегти отримані в такий спосіб знання в деякому проміжному вигляді;
- перетворити знання з проміжного подання у вигляд, придатний для практичного використання в програмі, наприклад у набір породжувальних правил. Перевага використання такої програми – зниження трудомісткості процесу, оскільки перенесення знань від експерта до системи здійснюється в один прийом.

Варто відзначити, що на практиці при виявленні знань у ході опитування експертів за робочий день вдається сформулювати від двох до п'яти "еквівалентів породжувальних правил". Причин такої низької продуктивності декілька:

- перш ніж приступити до опитування експертів, інженер зі знань, що не є фахівцем у даній предметній області, повинен витратити досить багато часу на ознайомлення з її специфікою і термінологією; тільки після цього процес опитування може стати продуктивним;
- експерти схильні думати про знайому їм область не стільки в термінах загальних принципів, скільки в термінах окремих типових об'єктів, подій і їхніх властивостей;

- для подання специфічних знань про предметну область потрібно підібрати придатну систему позначень і структурну оболонку, що само по собі є непростою задачею.

Як відомо, будь-яку складну задачу найкраще розбити на підзадачі, і саме так ми зробимо із задачею здобуття знань.

У роботі [22] пропонується виконати аналіз процесу здобуття знань у термінах моделі процесу проектування експертної системи (рис. 6.1):

1. *Ідентифікація*. Аналізується клас проблем, що передбачається вирішувати за допомогою проектованої системи, включаючи дані, якими потрібно оперувати, і критерії оцінювання якості рішень. Визначаються ресурси, доступні при розробці проекту, – джерела експертних знань; трудомісткість; обмеження за часом, вартістю та обчислювальними ресурсами.
2. *Концептуалізація*. Формулюються базові концепції і відношення між ними. Сюди ж входять і характеристика різних видів використовуваних даних, аналіз інформаційних потоків та на їх основі структур в предметній області, в термінах причинно-наслідкових зв'язків, відношень частина-ціле, постійне-тимчасове і т.п.
3. *Формалізація*. Починається спроба подати структуру простору станів і характер методів пошуку в ньому. Виконується оцінювання повноти і ступеня вірогідності (невизначеності) інформації й інших обмежень, що накладаються на логічну інтерпретацію даних, таких як залежність від часу, надійність і повнота різних джерел інформації.
4. *Реалізація*. Перетворення формалізованих знань у діючу програму, причому на перший план виходить специфікація методів організації керування процесом і уточнення деталей організації інформаційних потоків. Правила перетворюються у форму, придатну для виконання програмою в обраному режимі керування. Приймаються рішення про використовувані структури даних і розбиття програми на ряд більш-менш незалежних модулів.
5. *Тестування*. Перевірка роботи створеного варіанта системи на великому числі репрезентативних задач. У процесі тестування аналізуються можливі джерела помилок у поводженні системи. Найчастіше таким джерелом є наявний у системі набір правил. Виявляється, що в ньому не вистачає певних правил, одні – абсолютно коректні, а між деякими виявляються протиріччя.

Як видно з рис. 6.1, проектування експертної системи починається з аналізу класу проблем, що передбачається вирішувати за допомогою цієї системи. Було б похибкою приступати до проектування системи, заздалегідь задавшись певною концепцією і певною структурною організацією знань. Дуже сумнівно, щоб той варіант концепції чи той спосіб організації ідей, який ми задали апріорі, взявши за основу попередні розробки, був застосовний і до нової предметної області.

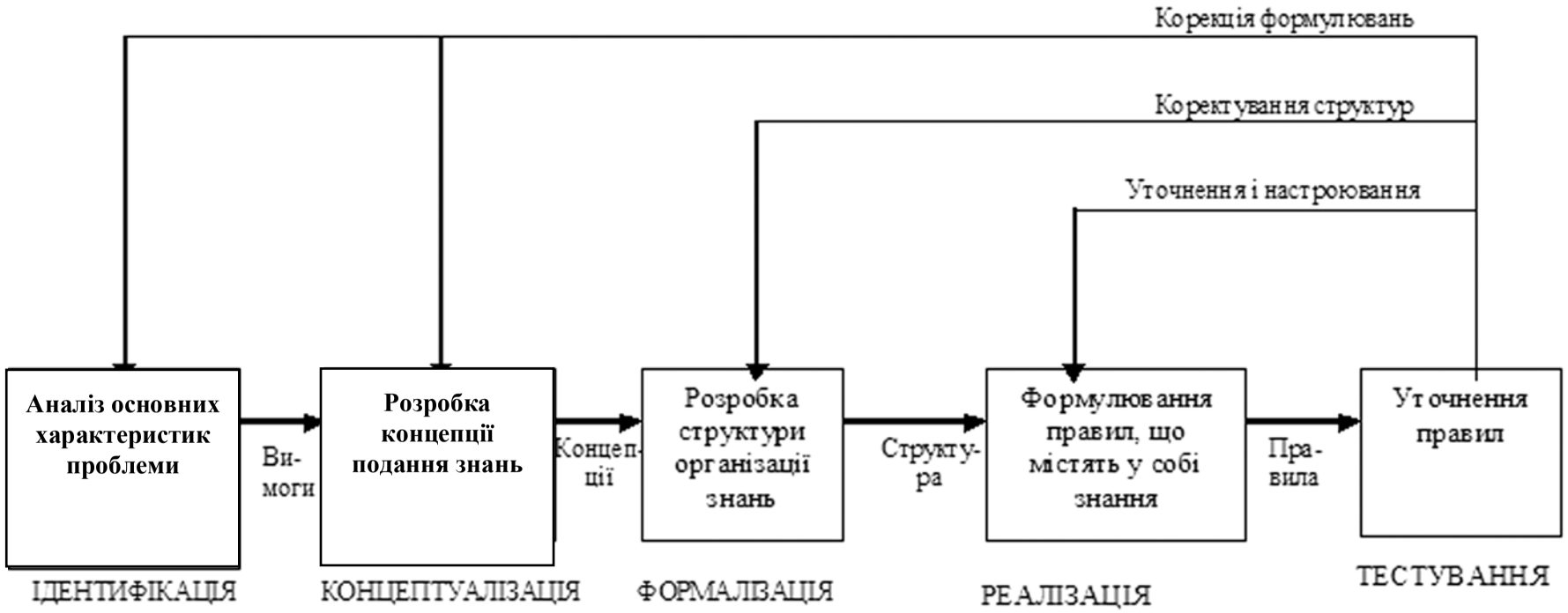

Рисунок 6.1 – Стадії здобуття знань

Рівні аналізу знань

Наведений вище поділ на етапи зустрічається також і в роботі вченого Уілінгі, що розробив моделюючий підхід до інженерії знань у рамках створеного ним середовища KADS [24]. В основі цього підходу лежить ідея про те, що експертна система є не контейнером, наповненим поданими експертом знаннями, а "операційною моделлю", що демонструє деяку потрібну нам поведінку у взаємодії з явищами реального світу. Здобуття знань, таким чином, містить у собі не лише отримання специфічних знань про предметну область, але й інтерпретацію отриманих даних стосовно до деякої концептуальної оболонки і їх формалізацію таким способом, щоб програма могла дійсно використовувати їх у процесі роботи.

В основу оболонки KADS покладено п'ять базових принципів.

1. Використання множини моделей, що дозволяє здолати складність процесів інженерії знань.
2. 4-рівнева структура для моделювання необхідної *експертності* – набору якостей, що лежать в основі вищого рівня роботи фахівців.
3. Повторне використання родових компонентів моделі як шаблонів, що підтримують спадну стратегію здобуття знань.
4. Процес диференціації простих моделей у складні.
5. Важливість перетворення моделей експертності зі збереженням структури в процесі розробки і впровадження.

Нижче розглянемо докладно два перших принципи.

Головним мотивом створення оболонки KADS було подолання складності знань. На сьогоднішній день в інженерів зі знань є можливість використовувати при побудові експертних систем безліч найрізноманітніших методів і технологій. Однак при цьому залишаються три основних проблемних питання:

- визначення проблеми, яку необхідно вирішити за допомогою експертної системи;
- визначення функцій, що покладаються на експертну систему стосовно до цієї проблеми;
- визначення задач, які необхідно вирішити для виконання покладеної функції.

Перший із принципів, покладених в основу KADS, полягає в тому, що оболонка повинна містити множину найрізноманітніших моделей, які допомагають знайти відповідь на ці питання. Прикладами таких моделей можуть служити:

- організаційна модель "соціально-економічного середовища", у якій повинна функціонувати система, наприклад фінансові послуги, охорона здоров'я і т.п.;
- прикладна модель вирішуваної проблеми і виконуваної функції, наприклад діагностика, планування розкладу робіт і т.д.;

- модель задач, що демонструє як повинна виконуватися специфікована функція, для чого здійснюється її розбиття на окремі задачі, наприклад збір даних про доходи, формування гіпотез про захворювання.

Між цією термінологією і тією, якою користався вчений Бучанан, немає прямої відповідності, але можна сказати, що організаційна і прикладна моделі аналогічні стадії ідентифікації в запропонованій вченим Бучананом структурі.

У підході, що реалізований при створенні KADS, стадія "концептуалізації" розбивається на дві частини: модель кооперації, чи комунікації, і модель експертності. Перша відповідає за декомпозицію процесу рішення проблеми, формування набору найпростіших задач і розподіл їх між виконавцями, у якості яких можуть виступати і люди, і машини. Друга модель представляє процес, що звичайно називається отриманням знань, тобто аналіз різних видів знань, які експерт використовує в ході вирішення проблеми. Крім зазначених, до складу оболонки KADS входить ще й модель проектування, що включає технології обчислень і механізми подання знань, що можуть бути використані для реалізації специфікацій, сформульованих попередніми моделями.

На перший погляд здається, що поданий вище аналіз у певній мірі розмиває відмінності між стадіями концептуалізації і формалізації. Можна, звичайно, заперечити, що стадія формалізації являє собою просто більш детальне пророблення концепцій і взаємовідношень, виявлених на ранніх стадіях. Модель проектування частково включає те, що в колишній схемі було віднесено до стадії реалізації, але вона не припускає створення діючої програми.

В своїх ранніх роботах вчений Уілінга трохи по-іншому проводив розмежування між рівнями аналізу [24]. Він розглядав чотири рівні аналізу.

- *Концептуалізація знань*. На цьому рівні передбачався формальний опис знань у термінах принципових концепцій і відношень між концепціями.
- *Рівень епістемологічного аналізу*. Метою такого аналізу було виявлення структурних властивостей концептуальних знань, зокрема таксономічних відношень.
- *Рівень логічного аналізу*. Основна увага приділялася тому, як будувати логічний висновок у даній предметній області на основі наявних знань.
- *Рівень аналізу впровадження*. Досліджувались механізми програмної реалізації системи.

У більш пізніх розробках три перших рівні включені до складу моделі експертності, а рівень аналізу впровадження – у модель проектування. 4-рівнева структура KADS погоджується із запропонованою

вченим Кленсі схемою поділу знань різного виду відповідно до їх ролі в процесі вирішення проблем [25]. Зокрема, знання, що стосуються конкретної предметної області, розділені на знання більш високого рівня (знання, що відносяться до побудови логічного висновку в цій предметній області), знання вибору розв'язуваних задач і знання стратегії вирішення задач.

Ці рівні знань показані в табл. 6.1. Стратегічний рівень керує процесом виконання задач, що використовують при вирішенні проблем методи логічного висновку, які підходять для конкретної предметної області, і знання з цієї області.

Таблиця 6.1 – 4-рівнева схема диференціації знань у системі KADS

Категорія знань	**Організація**	**Види знань**
Стратегічна	Стратегії	Плани, метаправила
Задача	Задачі	Цілі, керуючі терми, структури задач
Логічний висновок	Структура логічного висновку	Джерела знань, метакласи, схема предметної області
Предметна область	Теорія предметної області	Концепції, властивості, відношення

Необхідно відзначити, що описана схема диференціації знань приводить нас до досить простої архітектури експертної системи. Зокрема, виявляється, що навіть у рамках традиційної архітектури, яка припускає наявність бази знань і машини логічного висновку, можна неявним образом включити задачі і стратегії в структуру знань про предметну область, і в механізм побудови логічних висновків. Ми ще побачимо далі, що явне виділення цих задач і стратегій є головним моментом як у процесі здобуття знань, так і в процесі проектування структури експертної системи.

Онтологічний аналіз

Вчений Александер і його колеги запропонували ще один рівень аналізу знань, що одержав назву онтологічного аналізу [25]. В основі цього підходу лежить опис системи в термінах сутностей, відношень між ними і перетворення сутностей, що комплексно виконується в процесі вирішення деякої задачі. Автори зазначеної роботи використовують для структурування знань про предметну область три основні категорії:

- статична онтологія – до неї входять сутності предметної області, їх властивості і відношення;
- динамічна онтологія – визначає стани, що виникають у процесі вирішення проблеми, і спосіб перетворення одних станів в інші;
- епістемічна онтологія – описує знання, що керують процесом переходу з одного стану в інший.

У цій схемі спостерігається зовсім очевидна відповідність з рівнями концептуалізації знань і епістемологічного аналізу в структурі, запропонованій у роботі вченого Вікінга [24]. Але на нижніх рівнях – логічного аналізу й аналізу впровадження – така відповідність уже не спостерігається. Онтологічний аналіз припускає, що вирішувана проблема може бути зведена до проблеми пошуку, але при цьому не розглядається, яким саме способом потрібно виконувати пошук. Прикладом практичного застосування такого підходу є система OPAL.

Розглянута схема онтологічного аналізу виглядає досить абстрактною, але її цінність у тому, що вона спрощує аналіз слабоструктурованих задач. Кожен, хто мав справу з виявленням знань у процесі опитування людини-експерта, знає, як важко знайти придатну схему організації таких знань. Найчастіше в таких випадках говорять: "Давайте скористаємося фреймами чи системою правил", відкладаючи в такий спосіб вибір придатного методу реалізації на майбутнє, коли природа знань експерта стане більш зрозуміла.

6.2 Методи здобуття знань

Розглянувши в попередньому підрозділі поняття здобуття знань ще раз відзначимо, що здобуття знань – деяка взаємодія інженера зі знань і експерта у формі живого спілкування. Проте це не єдина, хоча і найпоширеніша форма здобуття знань.

Існує класифікація методів здобуття знань (рис. 6.2) і це дозволяє інженерам зі знань залежно від конкретної задачі і ситуації обрати конкретний метод [14,23].

Із запропонованої схеми класифікації видно, що основний принцип розподілу пов'язаний з джерелом знань.

Комунікативні методи охоплюють всі види контактів з живим джерелом знань – експертом.

Текстології – методи здобуття знань з документів і спеціальної літератури.

Розділення цих груп методів на верхньому рівні класифікації не означає їх антагоністичність, звичайно інженер зі знань їх комбінує.

Комунікативні методи в свою чергу також діляться на дві групи: активні і пасивні.

Пасивні – припускають провідну роль за експертом, а інженер зі знань лише фіксує (протоколює, записує на диктофон) міркування експерта під час його реальної роботи з прийняття рішень або записує те, що експерт вважає потрібним розповісти у вигляді лекції.

При активних методах, навпаки, ініціатива повністю в руках інженера зі знань, який активно контактує з експертами різними способами.

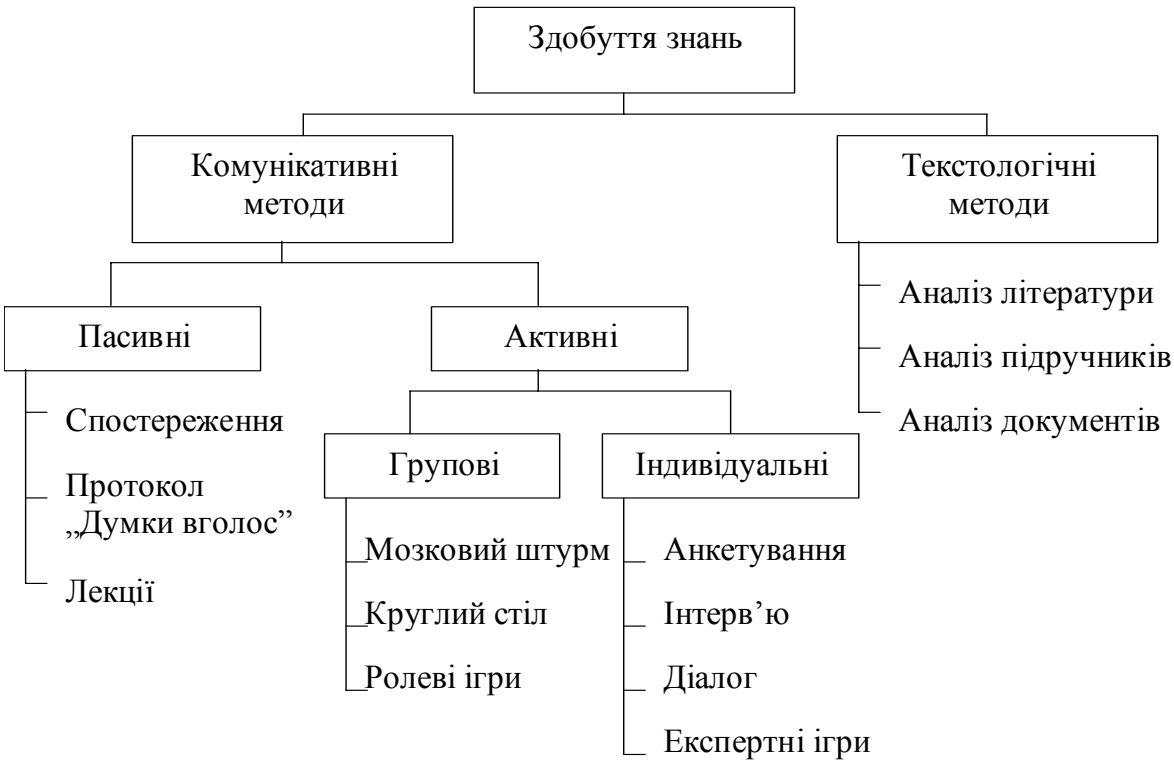

Рисунок 6.2 - Класифікація методів здобуття знань

Пасивні методи достатньо прості, але вимагають від інженера зі знань чітко аналізувати "потік свідомості" експерта і виявляти в ньому значущі фрагменти знань. Тут відсутній зворотний зв'язок, що ослаблює ефективність цих методів і визначає їх допоміжну роль.

Активні методи залежно від числа експертів, що передають свої знання, також діляться на дві групи. Якщо число експертів більше одного, то доцільно крім індивідуальних зустрічей застосовувати методи групових обговорень предметної області. Ці методи активізують мислення і дозволяють виявити нетривіальні аспекти знань. Проте індивідуальні методи і сьогодні є провідними, оскільки така делікатна процедура як отримання знань не передбачає зайвих свідків.

Ігрові методи широко використовуються в соціології, економіці, менеджменті, педагогіці для підготовки керівників, вчителів, лікарів та інших фахівців. Гра – це особлива форма діяльності і творчості, де людина стає більш розкутою, відчуває себе набагато вільнішою, ніж в звичній трудовій діяльності.

6.3 Структуризація знань предметної області

Концептуальна структура предметної області

Одна з найбільш творчих процедур при побудові ЕС – процедура концептуального аналізу одержаних знань або структуризація.

Структуризація – процес створення напівформалізованого опису предметної області. Такий напівформалізований опис називається полем

знань. Звичайно він створюється в графічній формі. Поле знань P_{zn} можна описати таким чином:

$$P_{zn}=<S_k, S_f>,$$

де S_k - концептуальна структура предметної області;

S_f - функціональна структура предметної області.

Концептуальна структура S_k, або модель предметної області, служить для опису її об'єктів і відношень між ними, тобто можна сказати, що концептуальна модель є S_t = <A,R>, де A – множина об'єктів предметної області; R – множина відношень, що пов'язують об'єкти. Множина відношень представляє собою зв'язки між об'єктами. Цими відношеннями аналітик фіксує концепцію предметної області, ієрархію понять, властивості і структуру об'єктів. Розробка концептуальної структури має самостійне значення, не залежне від кінцевої мети – розробки ЕС. Ця структура може служити для навчання, підвищення кваліфікації, для прогнозування, пояснення, реструктурування і т.п.

Основні об'єкти концептуальної моделі:

• АКО (A-Kind-OF) – "це є", наприклад, [Мас]-[АКО]-[ПК]. АКО відображає родовидові відношення - ієрархію понять предметної області. Цей об’єкт обов’язково присутній в будь-якій концептуальній структурі;

• A-part-of – "частина від", наприклад, [процесор]-[A-part-of]-[комп'ютер]. Це відношення служить для віддзеркалення фізичної структури і декомпозиції складних об'єктів на складові.

• Has-attribute – "має властивість", наприклад, [пам'ять]-[Has-attribute]- [обсяг пам'яті].

• Value – "значення", наприклад, [обсяг пам'яті]-[Vaiue]-[16Мбайт].

Поле значень може нагадувати семантичну мережу, але воно менш формалізоване. Якщо в мережі жорстко обумовлені різні види зв'язків, то в полі знань вони довільні.

Короткий алгоритм формування концептуальної структури:

1. Визначити всі результуючі поняття чи виходи системи. Це може бути набір діагнозів, рекомендацій, порад системи.

2. Визначити всі вхідні поняття або фактори, від яких залежить результат роботи системи.

3. Встановити проміжні поняття, які застосовуються в міркуваннях експертів, якщо вони є.

4. Для всіх понять знайти уточнюючі і узагальнюючі формулювання, тобто встановити ієрархію об'єктів.

5. Для об'єктів, які застосовуються в міркуваннях, визначити властивості і їх значення.

6. Визначити інші зв'язки і спробувати все в цілому зобразити графічно.

Відкинути зайві зв'язки, об'єкти, обговорити структуру з експертом, доповнити, якщо треба, повторюючи підпункти алгоритму 1-6.

Концептуальна структура предметної області зображена на рисунку.

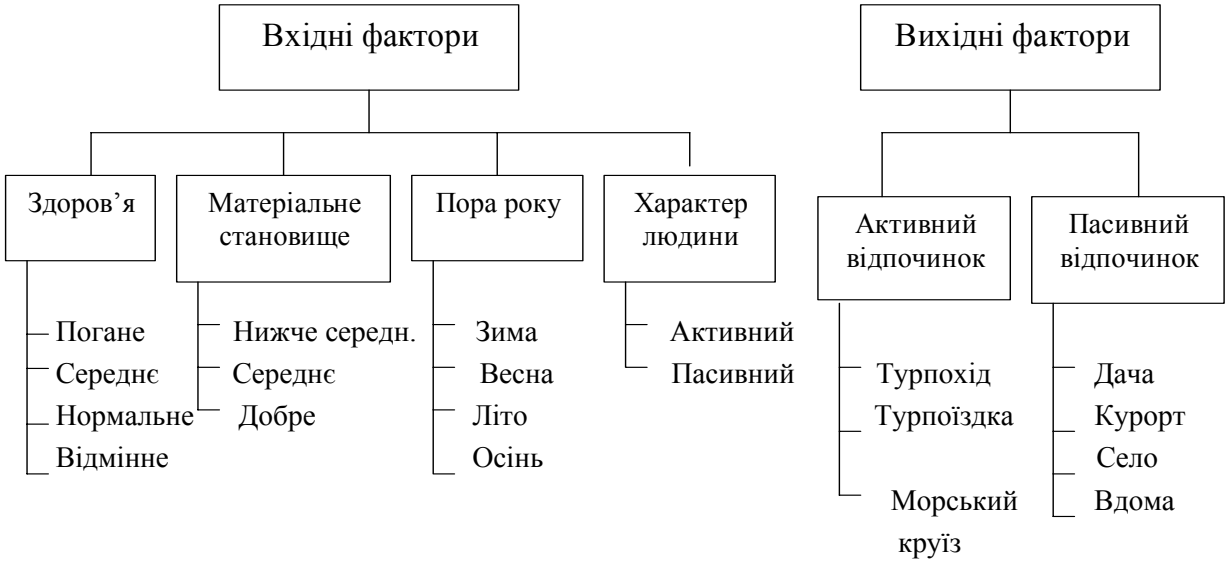

Рисунок 6.3 - Концептуальна структура предметної області

Функціональна структура предметної області

Функціональна структура відображає модель міркувань і прийняття рішення, якою користується експерт при вирішенні задач. Звичайно функціональна структура зображається у вигляді каузальних (від англ. cause – причина) відносин і може бути формалізована пізніше у вигляді коротких правил "ЯКЩО-ТО" (продукційна модель) або у вигляді семантичних мереж.

Зобразити функціональну структуру можна у вигляді таблиці, графа, або речень на природній мові. Бажано використовувати наочні форми.

У моделях міркувань можуть бути присутні нечіткі поняття: "часто", "багато", "дуже", "високий", "великий" і т.д.. Для подання їх в базі знань використовується так звана нечітка логіка, автор якої Л.Заде, запропонував простий формалізм для таких понять. Цей формалізм використовує поняття нечіткої функції приналежності, яка чисельно відображає на шкалі 0...10 або 0...1 ступінь упевненості експерта у тому, що конкретне значення можна віднести до даного нечіткого поняття. Ступені впевненості використовуються при множинних рекомендаціях. Наприклад, експерт радить "купувати акції компанії X із ступенем упевненості 9, а компанії У із ступенем упевненості 6".

Нижчевказану таблицю можна і далі продовжити для всіх комбінацій об'єктів з концептуальної моделі (табл. 6.2).

Формалізація і програмна реалізація бази знань

Сформувавши поле знань у вигляді концептуальної моделі і функціональної структури, інженер зі знань разом з програмістом

підбирають відповідну мову подання знань, яка, з одного боку, дозволить виразити всі особливості знань предметної області без спотворення структури поля знань, а з іншого, матиме ефективну програмну реалізацію у вигляді транслятора або оболонки. Якщо тепер ввести структуру з таблиці в базу знань будь-якої продукційної оболонки, то можна одержати ЕС, яка, ставлячи питання про ваше здоров'я, матеріальне положення, час відпустки і характер, допоможе вам вибрати відповідний вид відпочинку.

Таблиця 6.2 – Функціональна структура предметної області

Здоров'я	Матеріальне становище	Пора року	Характер	Відпочинок (1)	Ступінь упевненості (1)	Відпочинок (2)	Ступінь упевненості (2)
Погане	Нижче середнього	Весна	Пасивний	Дача	9	Дома	6
	Середнє	Літо	Пасивний	Село	8	Курорт	7
Середнє	Середнє	Осінь	Активний	Турпоїздка	8	Дача	3
	Добре	Зима	Пасивний	Курорт	7	Турпоїздка	5
Нормальне	Добре	Літо чи осінь	Активний	Турпоїздка	9	Похід	7

ПРИКЛАД
Побудуємо правило на підставі даних функціональної структури:

ЯКЩО здоров'я погане
І матеріальне становище нижче середнього
І характер пасивний
І пора року весна
ТО дача (ступінь упевненості =9)
АБО удома (ступінь упевненості =6)
і так далі.

6.4 Контрольні питання

1. Поняття здобуття знань.
2. Аналіз процесу здобуття знань.
3. Стадії здобуття знань.
4. Рівні аналізу знань.
5. Схема диференціації знань у системі KADS.
6. Онтологічний аналіз і здобуття знань.
7. Класифікація та характеристика методів здобуття знань.
8. Концептуальна структура предметної області.
9. Функціональна структура предметної області.
10. Формалізація і програмна реалізація бази знань.

7 ПОДАННЯ ЗНАНЬ: ПРИНЦИПИ ТА МЕТОДИ. МОДЕЛІ ПОДАННЯ ЗНАНЬ

7.1 Поняття та принципи подання знань

В галузі експертних систем подання знань означає не що інше, як систематизовану методику опису на машинному рівні того, що знає людина-експерт, яка спеціалізується в конкретній предметній області. Але помилково вважати, начебто подання знань зводиться до кодування в сенсі, аналогічному шифруванню. Якщо закодувати повідомлення, підставивши деяким регулярним образом замість одних символів інші, то отриманий результат не має нічого спільного з поданням змісту повідомлення в тому сенсі, як це припускається в теорії штучного інтелекту, навіть якщо отриманий код легко сприймається на машинному рівні і його можна зберігати в пам'яті комп'ютера.

Звернемо увагу хоча б на те, що в такому коді зберігається та лексична чи структурна неоднозначність, що притаманна природній людській мові. Наприклад, повідомлення "Відвідування тіточки може бути докучливим" буде настільки ж неоднозначним у закодованому виді, як і на "людській" мові. Переклад цього тексту в машинний код не позбавить нас від того, що це повідомлення можна трактувати і як твердження, що "набридає наносити візити тіточці", і як твердження, що "набридає, коли тіточка наносить візит".

Таким чином, один з парадоксів штучного інтелекту полягає в тому, що багато задач пошуку значеннєвого змісту, що легко вирішуються людиною, дуже важко реалізувати на ЕОМ і навпаки. Розглянемо таку фразу: "Молоток ударив графин, і він розбився".

До чого відноситься "він" у цій фразі? Для нас відповідь очевидна, і ми навіть не помічаємо неоднозначності в цій фразі. Але як у загальному значенні машина буде інтерпретувати цю фразу? Припущення, що "він" відноситься до останнього за порядком проходження в реченні іменника, не завжди спрацьовує. Наприклад:

"Графин вдарився об камінь, і він розбився."

Для нас зовсім очевидно, що потерпілим в обох випадках повинен бути графин. Ми володіємо тим, що називається "попереднім знанням", але незрозуміло, як воно повинно бути подане в ЕОМ. Також далеко не очевидно, як зібрати такого роду знання і як організувати їх отримання у конкретній ситуації. Єдине, що в цьому випадку можна запропонувати – сформувати величезну таблицю, яка складається з усіх можливих пар об'єктів у всесвіті, і вказати в ній, який із двох предметів більш крихкий.

Тепер розглянемо задачу із зовсім іншої області. Потрібно вирішити, чи є деяка логічна формула теоремою числення висловлювань. Наприклад, чи є теоремою формула

$$(p \,\&\, (q \supset r)) \supset ((s \vee p) \,\&\, \neg r \supset \neg q).$$

Виявляється, що не є, оскільки існує варіант, коли істинне значення присвоюється послідовно змінним p, q, r, s, і весь вираз стає помилковим. Написати програму, що допоможе комп'ютеру прийти до такого висновку– задача досить тривіальна, а зробити те ж саме звичайній людині досить складно.

За великим рахунком, різниця між цими двома задачами полягає в тому, що знання, необхідне для вирішення задач з області числення висловлювань, можна виразити в компактній формі у вигляді правил, а знання, що вимагаються для правильної інтерпретації будь-якої фрази у формі

"X ударив Y, і він розбився",

здаються на перший погляд нескінченними за обсягом і припускають безліч винятків начебто того, що існує і пластиковий молоток, і виточена з каменю ваза, паперова стіна і т.д. і т.п. Здається, що для вирішення подібних проблем програма повинна мати щось на кшталт "здорового глузду", у той час як для вирішення формальних логічних задач ніякого здорового глузду не потрібно.

Таким чином, будь-яке спілкування людини зі світом техніки припускає наявність деякого попереднього знання. Якщо, наприклад, хтось береться за пошук несправності в цифровій схемі, то можна припустити, що він володіє певними базовими знаннями з області електротехніки. Немає необхідності підкреслювати, що комп'ютер у чистому вигляді ніяких попередніх знань не має, а тому технічна експертність – набір якостей, що лежать в основі високого рівня роботи людей-фахівців при вирішенні проблем у певній вузькій області, – повинна включати і ці попередні знання [27].

І нарешті, подання припускає певну організованість знань. Подання знань повинне дозволити отримувати їх у потрібній ситуації за допомогою нескладного і більш-менш природного механізму. Простого переведення інформації (знань) у форму, придатну для збереження на машинних носіях, тут явно недостатньо. Для того щоб можна було досить швидко отримувати ті елементи знань, що найбільш придатні в конкретній ситуації, база знань повинна мати досить розвинуті засоби індексування і контекстної адресації. Тоді програма, що використовує знання, зможе керувати послідовністю застосування певних "елементів" знань навіть не маючи точної інформації про те, як вони зберігаються.

Звичайно, програмний код, виконуваний комп'ютером, повинен відповідати застосовуваній системі позначень, але це не можна вважати занадто серйозним обмеженням. Багато схем подання, на перший погляд, надзвичайно сильно відрізняються, проте виявляються насправді формально еквівалентними, тобто все, що може бути виражене в одній системі подання, може бути виражене й в іншій.

Перш ніж перейти до розгляду конкретних прикладів, давайте уточнимо термінологію, узявши за основу цитати з "класичних" робіт зі штучного інтелекту.

Подання (representation) у роботі вченого Уїнстона [28] визначається як "безліч синтаксичних і семантичних угод, що уможливлює опис предмета". У штучному інтелекті під "*предметом*" розуміється стан деякої проблемної області, наприклад об'єкти в цій області, їх властивості, відношення, які існують між об'єктами. *Опис* (description) "дозволяє використовувати угоди з подання для опису певних предметів". Синтаксис подання специфікує набір правил, що регламентують об'єднання символів для формування виразів мовою подання. Можна говорити про те, що вираз гарно чи погано сформований, тобто про те, наскільки він відповідає цим правилам. Зміст повинні мати лише добре сформовані вирази [28].

Загальноприйнятим в галузі штучного інтелекту є синтаксис у вигляді конструкції предикат-аргумент, що має форму

<вираз> ::= <предикат> (<аргумент>,..., <аргумент>)

У цій конструкції за *k*-значним предикатом повинні випливати *k* аргументів. Так, предикат „*at*” може бути двомісним відношенням, у якому перший аргумент виступає як ім'я деякого об'єкта, а в якості другого – його місцезнаходження (наприклад, кімната):

at(робот, кімната А)

Семантика подання специфікує, як повинен інтерпретуватися вираз, побудований відповідно до синтаксичних правил, тобто як з його форми можна отримати певний зміст. Специфікація звичайно виконується присвоєнням змісту окремим символам, а потім індукуванням присвоєння в більш складних виразах. Так, привласнюючи зміст символам „*at*”, робот, кімната А, ми можемо сказати, що вираз

at(робот, кімната А)

означає: робот знаходиться в кімнаті А (але не навпаки – кімната А знаходиться в роботі).

Процес вирішення проблеми, як правило, містить у собі поряд із поданням предметів навколишнього світу і судження про деякі дії. Як уже було відзначено деякі проблеми формулюються в термінах вихідного і цільового станів і множини операцій, які можна використовувати при спробах перетворити початковий стан у цільовий. Але тут залишається нез'ясованим питання про те, як можна подавати операції.

7.2 Логічні моделі подання знань. Логічне програмування

Логічне програмування є порівняно новим напрямком в програмуванні та інформатиці, який оснований на ідеях та методах запозичених із математичної логіки. Саме слово логіка тлумачиться як “наука про закони мислення та його форми” або як “хід міркувань, умовиводів”. Це слово походить від грецького слова “логос”, що з одного боку означає “слово” або “мова”, а з іншого – те, що виражає мова, тобто – мислення. Таким чином логіка має безпосереднє відношення до мови і стикується з граматикою та лінгвістикою.

Логіку створив видатний вчений стародавньої Греції – Арістотель в IV столітті до нашої ери. Він зробив це так вдало, що протягом багатьох століть вона залишалась незмінною. І в наш час розвиток логіки йде шляхом поширення та уточнення її основних положень, але не шляхом їх перегляду, як це сталося з більшістю інших наук.

Лише в XVIII столітті німецький вчений Лейбніц вирішив створити нову логіку, яка стала би “мистецтвом числення”. Він запропонував поставити у відповідність кожному поняттю деякий символ, і зобразити міркування у вигляді деяких обчислень. Однак його ідеї не були сприйняті вченими того часу і не отримали розвитку.

Ідею Лейбніца частково втілив у життя лише ірландський математик Дж.Буль, який в XIX столітті створив алгебру логіки, де діяли закони подібні до законів звичайної алгебри, але символами позначалися не числа, а речення. Мовою булевої алгебри можна було описувати міркування і “обчислювати” їх результати. Але вона охоплювала лише найпростіший тип міркувань. Алгебра логіки Буля на відміну від формальної логіки Арістотеля, отримала назву математичної логіки, оскільки вона використовувала мову та методи математики і була викликана саме потребами математики.

На початку XX ст. ряд вчених показали можливість застосування у техніці логіки висловлювань, яка є одним з розділів математичної логіки. В середині XX ст. було виявлено тісний зв’язок математичної логіки з кібернетикою. З того часу розпочалось широке впровадження математичної логіки в медицині, біології, економіці, техніці і т. ін.

Наприкінці 50-х років розпочались роботи зі створення методу автоматизованого доведення теорем, в результаті яких французькому вченому Робінсону вдалося створити правило виведення, придатне для виконання машиною. Це правило Робінсон назвав резолюцією. Але пройшло ще більш ніж 10 років напруженої роботи багатьох вчених, доки французькі вчені Кольморое і Русель у 1972 р. створили в Марсельскому університеті мову програмування, яка була основана на правилі резолюцій. Цю мову вони назвали ПРОЛОГ - мова ПРОграмування за допомогою ЛОГіки.

Перша версія ПРОЛОГу була створена на ФОРТРАНі, працювала дуже повільно і не знайшла широкого розповсюдження. Але ПРОЛОГ наочно продемонстрував великі потенційні можливості щодо реалізації ідей та методів штучного інтелекту, які були високо оцінені вченими різних країн. У 1981 році Міністерство зовнішньої торгівлі і розвитку Японії оголосило, що японський проект створення комп'ютерів п'ятого покоління буде засновуватись саме на методах логічного програмування. Метою проекту було створення систем обробки інформації, що базуються на знаннях, мають можливість спілкування з користувачем за допомогою природної мови або графічних засобів, і можуть надавати експертні консультації при розв'язанні задач високого рівня. На відміну від традиційних, продуктивність Пролог-програм вимірюється за допомогою нового параметра – кількості логічних виведень за секунду. Сьогодні вже створені експериментальні зразки Пролог-машин, продуктивність яких складає більш за 1 Млв./сек.

Формальні мови

Математична логіка є *формальною мовою* в тому сенсі, що у відношенні будь-якої послідовності символів вона дозволяє сказати, чи задовольняє ця послідовність правила конструювання виразів у цій мові (*формули*). Звичайно формальним мовам протиставляються природні, такі як французька і англійська, у яких граматичні правила не є жорсткими. Твердження, що логіка є *численням* з певними синтаксичними правилами логічного висновку, означає, що вплив одних членів виразу на інші залежить лише від *форми виразу* в даною мовою і ні в якому разі не залежить від будь-яких сторонніх ідей чи інтуїтивних припущень. Під *автоматичним формуванням суджень (automated reasoning)* розуміється поведінка деякої комп'ютерної програми, що будує логічний висновок на підставі певних законів. Так, не можна віднести до класу програм автоматичного формування суджень програму, що моделює підкидання монетки, щоб визначити, чи випливає одна формула з набору інших. В літературі також часто зустрічається термін *автоматична дедукція (automated deduction)*, рівнозначний за змістом терміну *автоматичне формування суджень*.

При реалізації автоматичного формування суджень, як правило, прагнуть до максимально можливої однаковості і стандартизації в поданні формул, але в той же час у літературі часто приходиться зіштовхуватися з найрізноманітнішими системами позначень, що відносяться до логіки. Основними синтаксичними схемами подання виразів є *кон'юнктивна нормальна форма (conjunctive normal form* – CNF), *повна фразова форма (full clausal form)* і *фраза Хорна (Horn clause)*, остання є підмножиною повної фразової форми [29]. Далі ми побачимо, що ці форми подання значно спрощують процедуру логічного висновку, але спочатку розглянемо деякі особливості числення висловлювань і предикатів.

Числення висловлювань

Числення висловлювань являє собою логіку неаналізованих припущень, у якій *пропозиціональні константи* можуть розглядатися як такі, що зображають певні прості вирази типу "Сократ- чоловік" і "Сократ смертний". Малі літери p, q, r, ... надалі будуть використовуватися для позначення пропозиціональних констант, що іноді називають атомарними формулами чи атомами.

Нижче наведені всі синтаксичні правила, що використовуються для конструювання правильно побудованих формул (ППФ) у численні висловлювань.

($S.\Psi$) Якщо Ψ є атомом, то Ψ є ППФ.

($S.\neg$) Якщо Ψ є ППФ, то $\neg\Psi$ також є ППФ.

($S.\vee$) Якщо Ψ і Φ є ППФ, то $(\Psi \vee \Phi)$ також є ППФ.

У цих правилах малі літери грецького алфавіту (наприклад, Ψ і Φ) зображають пропозиціональні змінні, тобто не атомарні формули, а будь-яке просте чи складне висловлення. Пропозиціональні константи є частиною мови висловлювань, що використовуються для застосування числення пропозиціональних змінних до конкретної проблеми.

Вираз $\neg\Psi$ читається як “не Ψ”, а $(\Psi \vee \Phi)$ читається як диз'юнкція “Ψ чи Φ (чи обидвох)”. Можна ввести інші логічні константи “$\wedge$” (кон’юнкція), “$\supset$” (імплікація чи обумовленість), “$\equiv$” (еквівалентність або рівнозначність), що, власне кажучи, є скороченнями комбінації трьох наведених вище констант.

У кон’юнктивній нормальній формі числення висловлювань константи “імплікація” і “еквівалентність” заміняються константами “заперечення” і “диз'юнкція”, а потім заперечення складного виразу розкривається за допомогою формул де Моргана:

$\neg(\Psi \wedge \Phi)$ перетворюється в $(\neg\Psi \vee \neg\Phi)$,
$\neg(\Psi \vee \Phi)$ перетворюється в $(\neg\Psi \wedge \neg\Phi)$,
$\neg\neg\Psi$ перетворюється в Ψ.

Останній етап перетворень – внесення диз'юнкцій усередину дужок:

$(\zeta \vee (\Psi \wedge \Phi))$ заміняється $((\zeta \vee \Psi) \wedge (\zeta \vee \Phi))$.

Прийнято скорочувати вкладеність дужкових форм, відкидаючи в нормальній кон’юнктивній формі знаки операцій $\vee$ і $\wedge$. Нижче зображений приклад перетворення виразу, що містить імплікацію двох дужкових форм, у нормальну кон’юнктивну форму.

$\neg(p \vee q) \supset (\neg p \wedge \neg q)$	Вихідний вираз.
$\neg\neg(p \vee q) \vee (\neg p \wedge \neg q)$	Виключення $\supset$.
$(p \vee q) \vee (\neg p \wedge \neg q)$	Введення $\neg$ усередину дужок.
$(\neg p \vee (p \vee q)) \wedge (\neg q \vee (p \vee q))$	Занесення $\vee$ усередину дужок.

$\{\{\neg p, p, q\}, \{\neg q, p, q\}\}$	Відкидання $\vee$ і $\wedge$ в кон'юктивній нормальній формі.

Вираз у внутрішніх дужках – це або атомарні формули, або негативні атомарні формули. Вирази такого типу називаються *літералами*, причому з погляду формальної логіки порядок літералів не має значення. Отже, для зображення безлічі літералів – фраз – можна запозичити з теорії множин фігурні дужки. Літерали в одній і тій же фразі неявно поєднуються диз'юнкцією, а фрази, розташовані у фігурних дужках, неявно поєднуються кон'юнкцією.

Фразова форма дуже схожа на кон'юнктивну нормальну форму, за винятком того, що додатні і від'ємні літерали в кожній диз'юнкції групуються разом по різні сторони від символу стрілки, а потім символ заперечення відкидається. Наприклад, приведений вище вираз

$$\{\{\neg p, p, q\}, \{\neg q, p, q\}\}$$

перетворюється у дві фрази:

$$p, q \leftarrow p,$$
$$p, q \leftarrow q,$$

у яких додатні літерали згруповані ліворуч від знаку стрілки, а від'ємні – праворуч.

Більш строго, фраза являє собою вираз вигляду

$$p_1, ..., p_m \leftarrow q_1, ..., q_n,$$

у якому $p_1, ..., p_m$, $q_1, ..., q_n$ є атомарними формулами, причому $m \geq 0$ і $n \geq 0$. Атоми в множині $p_1, ..., p_m$ зображають *висновки*, об'єднані операторами диз'юнкції, а атоми в множині $q_1, ..., q_n$ – умови, об'єднані операторами кон'юнкції.

Числення предикатів

Числення висловлювань має певні обмеження. Воно не дозволяє оперувати з узагальненими твердженнями вигляду "Всі люди смертні". Звичайно, можна позначити таке твердження деякою пропозиційною константою p, а іншою константою q позначити твердження "Сократ – людина". Але з $(p \wedge q)$ не можна вивести твердження "Сократ смертний".

Для цього потрібно аналізувати пропозиціональні символи у формі предикатів і аргументів, кванторів і квантифікованих змінних. Логіка предикатів надає нам набір синтаксичних правил, що дозволяють виконати такий аналіз, набір семантичних правил, за допомогою яких інтерпретуються ці вирази, і теорію доведень, що дозволяє вивести правильні формули, використовуючи синтаксичні правила дедукції.

Предикатами позначаються властивості, такі як “бути людиною”, і відношення, такі як бути “вище, ніж”.

Аргументи можуть бути окремими константами чи складним виразом “функція-аргумент”, що позначає сутності деякого середовища необхідних нам об'єктів, чи окремими квантифікованими змінними, котрі визначені в цьому просторі об'єктів. Спеціальні оператори – *квантори* – використовуються для зв'язування змінних і обмеження області їхньої інтерпретації. Стандартними є квантори спільності $(\forall)$ і існування $(\exists)$. Перший інтерпретується як “всі", а другий – “дехто” (чи “дещо”).

Нижче приведені синтаксичні правила числення предикатів першого порядку.

$(S.\alpha)$ Будь-який символ (константа чи змінна) є термом.
Якщо Γ_k є символом *k*-значної функції,
і $\alpha_1,...,\alpha_k$ є термами,
то $\Gamma_k(\alpha_1,...,\alpha_k)$ є термом.

$(S.\Psi)$ Якщо Ψ_k є символом k-значного предиката
$\alpha_1,...,\alpha_k$ є термами,
то $\Psi_k(\alpha_1,...,\alpha_k)$ є правильно побудованою формулою (ППФ).

$(S.\neg)$ та $(S.\vee)$ Правила запозичаються з числення висловлювань.

$(S.\forall)$ Якщо Ψ є ППФ і χ є змінною, то $((\forall\chi)\Psi)$ є ППФ.

Для позначення використовуються такі символи:

- Ψ – довільний предикат;
- Γ – довільна функція;
- α – довільний терм;
- χ – довільна змінна.

Дійсні імена, символи функцій і предикатів є елементами *мови першого порядку*. Використання квантора існування дозволяє перетворити терми з квантором спільності відповідно до означення

$(\exists\chi)\Psi$ визначається як $\neg(\forall\chi)\neg\Psi$.

Вираз $(\exists X)$(ФІЛОСОФ(X)) читається як “Дехто є філософом”, а вираз $(\forall X)$(ФІЛОСОФ(X)) читається як “Будь хто є філософом”. Вираз ФІЛОСОФ(X) являє собою правильно побудовану формулу, але це не речення, оскільки область інтерпретації для змінної X не визначена будь-

яким квантором. Формули, у яких усі згадані змінні мають певні області інтерпретації, називаються *замкнутими формулами*.

Як і в численні висловлювань, у численні предикатів існує нормальна форма подання виразів, але для побудови такої нормальної форми використовується розширений набір правил синтаксичних перетворень. Нижче наведена послідовність застосування таких правил. Для приведення будь-якого виразу до нормальної форми належить виконати такі операції.

1. Виключити оператори еквівалентності, а потім імплікації.

2. Використовуючи правила де Моргана і правила заміщення $(\exists\chi)\Psi$ на $\neg(\forall\chi)\neg\Psi$ (а отже, і $((\forall\chi)\Psi$ на $\neg(\exists\chi)\neg\Psi$), виконати приведення заперечення.

3. Виконати приведення змінних. При цьому варто враховувати особливості визначення області інтерпретації змінних кванторами. Наприклад, у виразі $(\exists X)$ (ФІЛОСОФ(X))& $(\exists X)$(АТЛЕТ(X)) змінні можуть мати різні інтерпретації в одній і тій же області. Тому винесення квантора за дужки – $(\exists X)$(ФіЛОСОФ(X))&(АТЛЕТ(X)) – дасть вираз, що не випливає з вихідної формули.

4. Виключити квантори існування. Квантори існування, що з'являються поза областю інтерпретації будь-якого квантора спільності, можна замінити довільним ім'ям (його називають *константою Сколема*), у той час як екзистенціальні змінні, які можуть існувати всередині області інтерпретації одного чи більше кванторів спільності, можуть бути замінені функціями Сколема. *Функція Сколема* – це функція з довільним ім'ям, що має такий зміст: “значенням даної змінної є деяка функція від значень, присвоєних універсальним змінним, в області інтерпретації яких вона знаходиться”.

5. Перетворити в префіксну форму. На цьому кроці всі квантори, що залишилися, (залишаться лише квантори спільності) переносяться “у голову” виразу й у такий спосіб виявляються ліворуч у списку квантифікованих змінних. За ними йде матриця, у якій відсутні квантори.

6. Рознести оператори диз'юнкції і кон’юнкції.

7. Відкинути квантори спільності. Тепер усі вільні змінні є неявно універсально квантифікованими змінними. Екзистенціальні змінні стануть або константами, або функціями універсальних змінних.

8. Як і раніше, відкинути оператори кон’юнкцій, залишивши множину фраз.

9. Знову перейменувати змінні, щоб ті самі імена не зустрічалися в різних фразах.

Мова PROLOG

Фрази Хорна (Horn clause) являють собою підмножину фраз, що містять тільки один додатний літерал. У загальному вигляді фраза Хорна зображається виразами

$$p \leftarrow q_1,...,q_n$$

У мові PROLOG ця ж фраза записується в такому вигляді (зверніть увагу на символ крапки наприкінці):

p :- q1,…,qn.

Така фраза інтерпретується в такий спосіб:

“Для всіх значень змінних у фразі *p* істинно, якщо істинні q_1 і...і q_n”,

тобто пари символів “:-” читається як “якщо”, а коми читаються як “і”.

PROLOG – це не зовсім звичайна мова програмування, у якій програма складається в основному з логічних формул, а процес виконання програми являє собою доведення теореми певного вигляду.

Фраза у формі

p :- ql,...,qn.

може розглядатися як процедура. Така процедура припускає такий порядок виконання операцій.

(1) Літерал мети зіставляється з літералом p (*уніфікується* з p), що називається *головою* фрази.

(2) Хвіст фрази q1,...,qn конкретизується підстановкою значень змінних (чи уніфікаторів), сформованих у результаті цього зіставлення.

(3) Конкретизовані терми хвостової частини утворюють потім множину підцілей, які можуть бути використані іншими процедурами.

Таким чином, зіставлення (чи *уніфікація*) відіграє ту ж роль, що і передача параметрів функції в інших, більш звичних мовах програмування.

Наприклад, розглянемо набір фраз мови PROLOG, поданих у лістингу 7.1. Припустимо, що *a, b* і *c* – певні блоки у множині блоків. Дві перші фрази стверджують, що *a* знаходиться на (on) *b*, а *b* знаходиться на (on) *c*. Третя фраза стверджує, що *X* знаходиться вище (above) *Y*, якщо *X* знаходиться на (on) *Y*. Четверта фраза стверджує, що *X* знаходиться вище (above) *Y*, якщо існує якийсь інший блок *Z*, розміщений на (on) *Y*, і *X* знаходиться вище (above) *Y*.

Лістинг 7.1. Проста програма мовою PROLOG, що визначає відношення on (на)

```
on(a, b).
on(b, c).
above(X, Y)  :-  on(X, Y).
above(X, Y)  :-  on(Z, Y), above(X, Z).
```

Очевидно, що від програми вимагається вивести мету above (a, c) з цієї множини фраз. Тут необхідно відзначити, що процес формулювання виразу мети включає обробку двох процедур *above* і використання двох фраз *on*.

Спростування резолюцій

У мові PROLOG використовується "інтерпретація фраз Хорна для вирішення проблем" [30]. Фундаментальний метод доведення теорем, на якому базується PROLOG, називається спростуванням резолюцій *(resolution refutation)*. Повний опис цього методу читач знайде в книзі Робінсона [31], а ми спробуємо коротко викласти тільки основні ідеї.

Принцип резолюцій

Нагадаємо, що ми намагаємося спростити синтаксис числення таким чином, щоб зменшити кількість правил впливу, необхідну для доведення теорем. Замість дюжини або більше правил, що використовуються при доведенні теорем вручну, системи автоматичного доведення для фразових форм використовують єдине правило доведення – принцип резолюції, – вперше описане Робінсоном [31].

Розглянемо такий приклад з числення висловлень. Надалі великими буквами *P*, *Q*, *R*, ... будуть позначатися окремі фрази, а малими грецькими Ψ, Φ і ζ - пропозиціональні змінні, як і раніше.

Якщо Ψ і Φ зображують дві довільні фрази, які можна зобразити в кон'юнктивній нормальній формі, і

$\Psi = \{\Psi_1,...,\Psi_i,...,\Psi_m\}$, і

$\Phi = \{\Phi_1,...,\Phi_i,...,\Phi_n\}$, і

$\Psi_i = \neg\Phi_j$, при 1[i[m,1[j[n

то нову фразу ζ можна вивести із з'єднання Ψ' і Φ', де

$\Psi' = \Psi - \{\Psi_i\}$ і

$\Phi' = \Phi - \{\Phi_i\}$

Фраза $\zeta = \Psi' \cup \Phi'$ називається резольвентою кроку резолюції, а Ψ і Φ є батьківськими фразами. Іноді говорять, що Ψ і Φ "зіштовхуються" на парі доповнювальних літералів Ψ_i і Φ_i.

Потужність резолюції забезпечується тим, що в ній підсумовується множина інших правил. Це стане очевидно після того, як звичайні правила будуть зображені в кон'юнктивній нормальній формі.

У лівій колонці табл. 7.1 перераховані найменування правил виведення, у середній показано, як вони виглядають у звичайних позначеннях, а в правій колонці – у фразовій формі. У кожному записі вирази у верхній частині зображають схему передумов, а вирази в нижній частині – схему висновків. З цієї таблиці видно, що кожне з наведених вище п'яти правил є одним з екземплярів резолюції.

Таблиця 7.1 – Узагальнення резолюції

Правило виведення	**Звичайна форма**	**Кон'юктивна нормальна форма**
Modus ponens	$\dfrac{\Psi \supset \Phi, \Psi}{\Phi}$	$\dfrac{\{\neg\Psi, \Phi\}, \{\Psi\}}{\{\Phi\}}$
Modus fallens	$\dfrac{\Psi \supset \Phi, \neg\Phi}{\neg\Psi}$	$\dfrac{\{\neg\Psi, \Phi\}, \{\Phi\}}{\neg\Psi}$
Зчеплення	$\dfrac{\Psi \supset \Phi, \Phi \supset \zeta}{\Psi \supset \zeta}$	$\dfrac{\{\neg\Psi, \Phi\}\{\neg\Phi, \zeta\}}{\{\neg\Psi, \zeta\}}$
Злиття	$\dfrac{\Psi \supset \Phi, \neg\Psi \supset \Phi}{\Phi}$	$\dfrac{\{\Psi, \Phi\}, \{\neg\Psi, \Phi\}}{\{\Phi\}}$
Reductio	$\dfrac{\Psi, \neg\Psi}{\perp}$	$\dfrac{\{\neg\Psi\}, \{\Psi\}}{\{\}}$

Зверніть увагу на те, що протиріччя в правилі, яке звичайно позначається значком „$\perp$”, дає в результаті порожню фразу – „{}”. Це означає, що передумови несумісні. Якщо вважати, що передумови описують деякий стан предметної області, то такий набір передумов не може бути реально забезпечений у ній, тобто такий стан неможливий. Головне, що потрібно відзначити з усього сказаного вище, що компонент автоматичного доведення теорем, який є основним компонентом більшості систем штучного інтелекту і, зокрема, мов програмування штучного інтелекту, таких як PROLOG, є *системою спростування резолюцій*. Для того щоб довести, що p випливає з деякого опису стану (чи теорії) T, потрібно застосувати $\neg p$ і спробувати довести, що з цього припущення випливає твердження, що суперечить T. Якщо це вдасться зробити, то тим самим підтверджується твердження p, а в протилежному випадку воно спростовується.

У численні предикатів використання резолюцій вимагає додаткових зусиль, оскільки в цьому численні присутні змінні. Основна операція зіставлення в доведенні теорем за допомогою резолюцій називається *уніфікацією* (докладний опис використовуваного при цьому алгоритму читач знайде, наприклад, у роботі Нільсона [32]). При зіставленні доповнювальних літералів відшукується така підстановка змінних, котра перетворює обидва вирази в ідентичні.

Наприклад, вираз БІЖИТЬ_ШВИДШЕ_НІЖ (*X, равлик*) і БІЖИТЬ_ШВИДШЕ_НІЖ (*черепаха, Y*) перетворюються в ідентичні при підстановленні {*X/черепаха, Y/равлик*}. Така підстановка називається *уніфікатором*. Основна мета програміста – відшукати найбільш загальну підстановку такого роду.

7.3 Продукційні моделі подання знань. Проектування продукційних експертних систем

В традиційному програмуванні команди слідують в жорстко фіксованій послідовності. Всі місця розгалуження задаються в явному вигляді. Такий спосіб програмування є зручним, коли послідовність обробки мало залежить від даних, які обробляються, тобто коли розгалуження є винятком, а не нормою. В іншому випадку програму зручніше розглядати як сукупність незалежних модулів, які управляються зразками. Така програма на кожному кроці аналізує поточну ситуацію і визначає шляхом аналізу зразків, який саме модуль більше за інших підходить для її обробки.

Кожний з модулів, які управляються зразками, складається з механізмів дослідження і модифікації однієї або кількох структур даних. Системи, що створюються на основі модулів, які управляються зразками, називають системами виведення, які управляються зразками. Функції управління в таких системах виконує інтерпретатор. З точки зору подання знань такий підхід характеризується рядом особливостей:

1. Розподіл постійних знань, які зберігаються в базі знань і тимчасових знань, які зберігаються в базі даних;
2. Структурна незалежність модулів, яка спрощує модифікацію та вдосконалення системи, що дуже важливо для ЕС;
3. Відокремлення системи управління від модулів, які містять знання про ПО. Така властивість дозволяє застосування різних систем управління до тієї ж самої бази знань.

Як правило, в системах що розглядаються, модуль розділяють на дві частини: передумову і дію. Модулі, які мають таку структуру називають *правилами*, а відповідні системи, *системами, що основані на правилах*.. Системи, що створюються з правил, в яких зіставлення і управління є явними функціями системи, зафіксованими в інтерпретаторі, називають *продукційними системами*.

В найбільш загальному вигляді *продукцією* називають такий вираз:

$$(r)\ Q\ ;\ P\ ;\ A \rightarrow B\ ;\ N,$$

де r – ім’я продукції, за допомогою якого дана продукція виділяється серед інших. Ім’я може бути зображене порядковим номером продукції або деякою лексемою, що відображає її зміст, наприклад «придбання книги».

Q – характеризує сферу застосування продукції, яка легко визначається людиною. Знання людини неначе розкладені по полицях: на одній – як готувати їжу, на іншій – як добратись до роботи і т.і. Розділення знань на окремі сфери дозволяє економити час при пошуку потрібних знань.

P – умова можливості застосування ядра продукції. Як правило, це логічний вираз, частіше за все предикат. Якщо P істинно, то ядро продукції активізується. Розглянемо продукцію "Наявність грошей: якщо бажаєш купити X, то сплати в касу вартість X і віддай чек продавцю". Якщо гроші відсутні, то умова застосування ядра продукції хибна, тобто застосувати продукцію неможливо;

N – післяумова продукції. Активізуються лише в тому випадку, якщо буде виконане ядро продукції. Описують дії і процедури, які слід виконати після виконання B. Наприклад, після покупки товару в магазині слід в описі товарів зменшити кількість даного товару на одиницю.

A→B – ядро продукції. Ядра продукцій класифікують за різними ознаками. Перш за все виділяють два великих класи: детерміновані і недетерміновані ядра. В детермінованих ядрах в разі виконання умови A обов'язково виконується дія B, а в недетермінованих дія B може і не виконуватися. Тобто, в даному випадку продукція може мати такий вигляд: ЯКЩО A, ТО можливо B, ЯКЩО A, ТО з ймовірністю p B, ЯКЩО A, ТО з великим ступенем впевненості B.

Детерміновані продукції можуть бути однозначними або альтернативними, наприклад ЯКЩО A, ТО частіше за все B, інколи C. Особливим типом є продукції прогнозу вигляду:

ЯКЩО A, ТО з ймовірністю P можна очікувати B.

Одним з основних недоліків продукційних систем, поряд з такими перевагами, як модульність і простота реалізації та модифікації; є, поряд зі складністю перевірки несуперечності нових правил, їх недетермінованість (неоднозначність вибору однієї продукції з множини готових до виконання продукцій). Можливі два основних шляхи їх вирішення: централізований і децентралізований. В першому випадку рішення про застосування тієї або іншої продукції приймається спеціальною системою керування, а в другому – воно визначається ситуацією, що склалася на даний момент часу. Серед найбільш поширених стратегій управління виконанням продукцій слід відзначити такі:

- принцип купки книг – оснований на припущенні, що найбільш корисною є та продукція, яка найчастіше використовується. Метод достатньо ефективний при наявності зворотного зв'язку, який дозволяє аналізувати результат застосування продукції. При цьому метод може використовуватися для самонавчання системи. Метод застосовується в умовах значного ступеня взаємної незалежності продукцій;

- принцип найбільш довгої умови – полягає у виборі з множини готових продукцій такої продукції, у якої істинним є найбільш довга умова можливості застосування ядра. Основою тут є логічне припущення, що окремі правила, які мають відношення до більш вузького класу ситуацій є більш важливими, ніж загальні правила, що мають відношення до широкого класу ситуацій, оскільки перші враховують більше інформації про ситуацію ніж другі. Метод добре працює на добре структурованих знаннях, на яких задано ієрархію вигляду „окреме-загальне”;
- принцип метапродукцій – базується на використанні спеціальних метапродукцій, які управляють вибором продукцій. Прикладом метапродукції може бути такий: ЯКЩО в ситуації присутній Х і серед готових продукцій є така, в умовній частині якої є А, ТО продукції в умовних частинах яких зустрічається В слід виконувати раніше ніж ті, в умовних частинах яких зустрічається А;
- принцип класної дошки – оснований на виділенні особливої зони пам’яті – класної дошки, на якій крейдою пишуть повідомлення і в разі необхідності стирають їх. На цій дошці процеси, що виконуються паралельно, знаходять інформацію, яка активізує їх запуск, і тут же залишають інформацію про наслідки своєї роботи, яка може знадобитися для роботи інших програм;
- принцип пріоритетного вибору – пов’язаний із введенням статичних або динамічних пріоритетів на продукції. Статичні пріоритети формуються експертами на етапі розробки бази знань. Динамічні – можуть формуватись, наприклад, в залежності від часу знаходження продукції у множині готових продукцій.

Проектування продукційних експертних систем

Продукційна модель – це така модель, яка основана на правилах, що дозволяють подати знання у вигляді речення типу: ЯКЩО (умова), ТО (дія), або системами з висновком, що використовує зіставлення за зразком.

Під умовою розуміється деяке речення-зразок, за яким здійснюється пошук в базі знань, а під дією – дії, що виконуються при успішному результаті пошуку (вони можуть бути проміжними, які виступають далі як умови, і термінальними або цільовими, які завершають роботу системи).

Обчислювальний процес в продукційних системах істотно відрізняється від класичної фоннеймановської схеми, прийнятої в більшості комп'ютерів, в якій і програми, і дані при обчисленнях знаходяться в одній області пам'яті.

Програма може перетворювати саму себе як дані, змінюючи тим самим послідовність своїх дій [33].

У противагу цьому продукційна система майже не має процедурних компонентів і практично повністю управляється даними, тобто є дескриптивною.

Продукційна система включає три основних складових: базу правил, робочу пам'ять і механізм виведення. Для підтримки роботи системи і реалізації інтелектуальної взаємодії з користувачем в неї входять і підсистема надбання знань, і засоби спілкування на природною мовою, і підсистема пояснень.

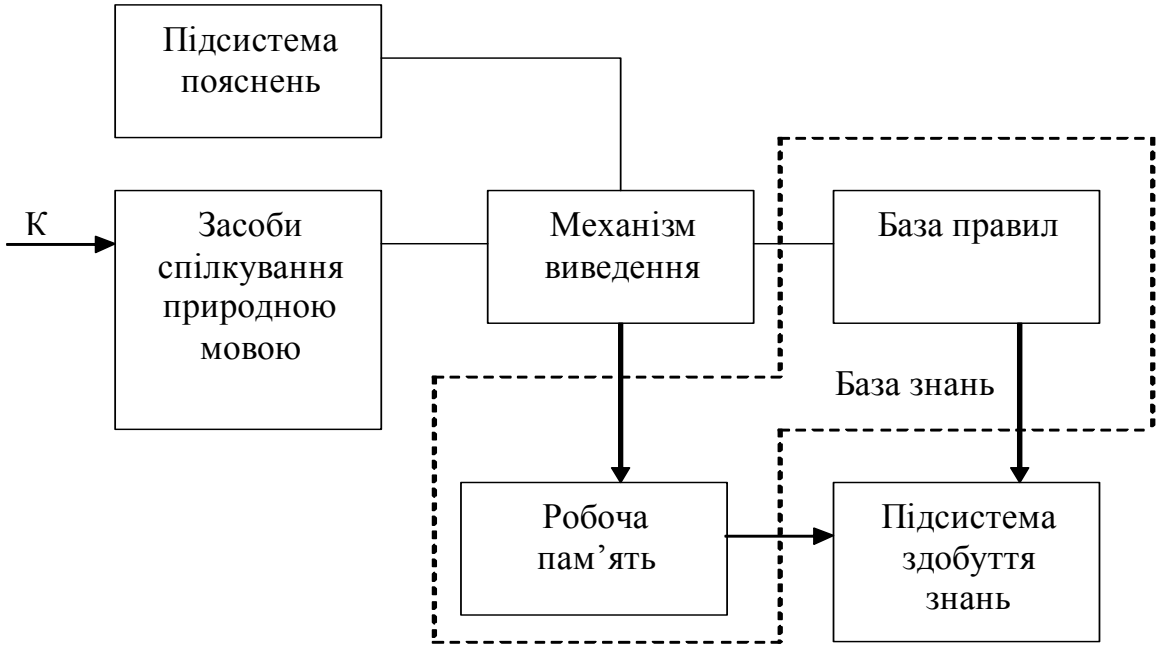

Рисунок 7.1 - Структура ЕС продукційного типу

Розглянемо окремі елементи структури ЕС продукційного типу.

База правил

База правил – це пам'ять для зберігання правил. Розглянемо на прикладі пошуку несправності в автомобілі.

Якщо за справу береться професіонал, діє він дуже цілеспрямовано, використовуючи певну стратегію або евристики. Знайшовши, що аналізована система (автомобіль) несправна, фахівець намагається визначити ту її підсистему, яка дала збій Після виявлення такої підсистеми, він шукає несправні компоненти. Цей процес виглядає так, ніби фахівець постійно звертається до неявно заданої множини правил, одержуючи проміжні висновки при виконанні будь-яких умов. Ці проміжні висновки стають умовами для виведення наступних висновків і т.д.

Приклад:

ЯКЩО двигун не заводиться
І стартер двигуна не працює
ТО неполадки в системі електроживлення стартера

ЯКЩО двигун не заводиться
І стартер двигуна працює
ТО неполадки в системі подачі палива

ЯКЩО неполадки в системі подачі палива
І показник рівня палива знаходиться па нулі
ВИСНОВОК паливна камера порожня

ЯКЩО неполадки в системі електроживлення стартера
І порушені контакти акумулятора
ВИСНОВОК погано приєднаний акумулятор.

Таким чином, система повинна включати пам'ять для зберігання правил, яка містить продукційні правила вигляду ЯКЩО -ТО.

Кожне правило складається з двох частин:

- перша частина – антецедент, або умова правила, складається з елементарних речень, що сполучаються логічними зв'язками І, АБО;
- друга – консеквент, або висновок, складається з одного або декількох речень, які утворюють рішення, або вказують на дію, що підлягає виконанню.

Продукційне правило для приведеного вище прикладу:

ЯКЩО двигун не заводиться | *антецедент*
І стартер двигуна не працює |
ТО неполадки в системі електроживлення стартера | *консеквент*

Антецедент – зразок правила, призначеного для розпізнавання ситуації, коли воно повинне спрацювати. Правило спрацьовує, якщо факти з робочої пам'яті при зіставленні збіглися із зразком, після чого правило вважається відпрацьованим.

Кожне правило в цьому простому прикладі містить атрибути і значення. Для вказаного прикладу пари атрибут–значення такі:

Атрибут	**Значення**
Двигун	Не заводиться
Стартер двигуна	Не працює
Система електроживлення стартера	Несправна

Будь-яке правило складається з однієї або декількох пар атрибут-значення і висновку.

Продукційну модель, що є основою для побудови бази правил, по суті можна розглядати як варіант моделі "стимул-реакція", яка була запропонована в психології і теорії автоматів задовго до появи ЕС. Багато учених вважають, що ця модель повністю адекватно описує поведінку людини.

Робоча пам'ять

Інша важлива частина системи, основаної на знаннях – *робоча пам'ять* – база даних або база фактів. У ній зберігається множина фактів, що описують поточну ситуацію, і всі пари атрибут-значення, які були встановлені до певного моменту.

Вміст робочої пам'яті з часом змінюється, збільшуючись в обсязі у міру спрацьовування правил, або зменшуючись, якщо дія правила полягає у видаленні будь-яких фактів.

У наведеному прикладі до початку процесу виведення в робочій пам'яті знаходилися тільки такі факти: двигун не заводиться, стартер двигуна не працює. Після застосування першого правила в робочу пам'ять додається новий факт: система електроживлення стартера несправна. Врешті-решт, буде виведене остаточне рішення, яке теж заноситься в робочу пам'ять.

Робоча пам'ять є *динамічною* частиною бази знань, що змінюється з часом. Нові факти, що додаються в робочу пам'ять, є результатом виведення, який полягає в застосуванні правил до наявних фактів. База знань включає сукупність правил і вміст робочої пам'яті.

У системах з монотонним виведенням, факти, що зберігаються в робочій пам'яті, статичні, тобто не змінюються в процесі рішення задачі.

У системах з немонотонним виведенням допускається зміна або видалення фактів з робочої пам'яті.

Як приклад системи з немонотонним виведенням можна привести ЕС, призначену для складання перспективного плану капіталовкладень компанії. У такій системі за бажанням користувача можуть бути змінені навіть ті дані, які після введення вже викликали спрацьовування будь-яких правил. Іншими словами, є можливість модифікувати значення атрибутів у складі фактів, що знаходяться в робочій пам'яті.

Зміна фактів, у свою чергу, приводить до необхідності видалення з робочої пам'яті виведень, одержаних за допомогою згаданих правил. Тим самим, виведення виконується повторно для того, щоб переглянути ті рішення, які були одержані на основі фактів, що піддалися зміні.

Механізм виведення

Механізм виведення або *інтерпретатор правил* виконує дві основні функції:

- Здійснює перегляд існуючих фактів з робочої пам'яті і правил з бази знань і додає, в міру можливості, нові факти в робочу пам'ять.
- Визначає порядок перегляду і застосування правил.

Цей механізм управляє процесом консультації, зберігаючи для користувача інформацію про одержані висновки, і запитує у нього інформацію, коли для спрацьовування чергового правила в робочій пам'яті виявляється недостатньо даних.

У деяких системах прийнятий прямий порядок виведення – від фактів, що знаходяться в робочій пам'яті, до висновку. У інших системах виведення здійснюється в зворотному порядку – висновки проглядаються послідовно до тих пір, поки не будуть знайдені в робочій пам'яті або одержані від користувача факти, які підтверджують один з них.

У переважній більшості систем, основаних на знаннях, механізм виведення є невеликою за обсягом програмою. Основну частину пам'яті ПК займають правила.

Відповідно до функцій механізм виведення включає два компоненти – один з них реалізує безпосередньо виведення, інший управляє цим процесом.

Компонент виведення виконує першу задачу, проглядаючи наявні правила і факти з робочої пам'яті і, додаючи в неї нові факти при спрацьовуванні будь-якого правила.

Управляючий компонент визначає порядок застосування правил.

Компонент виведення. Його дія основана на застосуванні правила виведення, визначеного як „модус поненс”, його суть така:

нехай відомо, що істинне твердження А і існує правило вигляду: "ЯКЩО А, то В", тоді твердження В також істинне.

Правила спрацьовують, коли знаходяться факти, що задовольняють їх ліву частину: якщо істинне посилання, то повинен бути істинним і висновок.

На перший погляд здається, що такий висновок може бути легко реалізований на ПК, проте, на практиці людський мозок виявляється ефективнішим при вирішенні задачі.

Приклад.

Розглянемо речення: Наталка шукала ключ.

Тут для слова "ключ" допустимі як мінімум два значення – "джерело" і "ключ від квартири".

У наступних реченнях одне і теж слово має абсолютно різні значення:

На горі знаходиться стародавній замок.

Замок не відчинявся, мабуть зламався.

Зрозуміти факти стає ще складніше, якщо вони є складовими частинами продукцій, які використовують правило „модус поненс” для виведення висновків.

Приклади:

ЯКЩО білий автомобіль легко помітити вночі

І автомобіль Діми білий

ТО автомобіль Діми легко помітити вночі.

Цей висновок виявляється не під силу жодній з сучасних ЕС, хоча в житті його зробить навіть дитина.

ЯКЩО Микола був в ресторані Вадима
І замовив там устриці,
І заплатив офіціанту за устриці,
ТО Микола отримав устриці в ресторані Вадима.

І в цьому випадку твердження висновку наведеної продукції очевидне для будь-якої людини, але виведення його без додаткових правил знаходиться за межами можливостей існуючих ЕС.

Підводячи підсумки можна сказати, що людина здатна зробити велике число висновків за допомогою дуже великої бази знань, яка зберігається в її пам'яті.

Експертні системи можуть зробити лише порівняно невелике число висновків, використовуючи задану множину правил.

Компонент виведення повинен мати здатність функціонувати в умовах недостачі інформації. У прикладі з пошуком несправності автомобіля факт порушення контактів акумулятора міг би бути відсутнім в робочій пам'яті системи на початку сеансу консультацій. Компонент виведення міг би звернутися до користувача за відомостями про стан контактів. А якщо і користувачу це не відомо?

Механізм виведення повинен бути здатний продовжити міркування і з часом знайти рішення навіть при недостачі інформації. Це рішення може і не бути точним, проте ЕС у жодному випадку не повинна зупинятися через те, що відсутня будь-яка частина вхідної інформації (як, наприклад, звична програма, що визначає доцільність надання позики).

Управляючий компонент (інтерпретатор продукцій). Цей компонент визначає порядок застосування правил і визначає, чи є ще факти, які можуть бути змінені у разі продовження консультації.

Управляючий компонент виконує чотири функції:

1. Зіставлення – зразок правила порівнюється з наявними фактами.
2. Вибір – якщо в конкретній ситуації можуть бути застосовані декілька правил, то з них вибирається одне, найбільш відповідне за заданим критерієм.
3. Активізація – якщо зразок правила при зіставленні збігається з будь-якими фактами з робочої пам'яті, то правило активізується.
4. Дія – робоча пам'ять змінюється, в неї додаються висновки правила, що активізувалось. Якщо в правій частині правила міститься вказання на будь-яку дію, то воно виконується (як наприклад в системах забезпечення безпеки інформації).

Інтерпретатор продукції працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити серед них ті, умови яких збігаються з відомими на даний момент фактами з робочої пам'яті.

Інтерпретатор визначає порядок застосування правил.

Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, потім цикл повторюється спочатку.

У одному циклі може спрацювати тільки одне правило. Якщо декілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, яке і спрацює в даному циклі.

Інформація з робочої пам'яті послідовно зіставляється з умовами правил для виявлення збігу. Сукупність відображених правил складає так звану *конфліктну множину*. Для вирішення конфлікту інтерпретатор має критерій, за допомогою якого він вибирає єдине правило, після чого воно спрацьовує. Спрацьовування полягає в занесенні фактів, з яких слідує висновок правила, в робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо у висновок правила входить назва будь-якої дії, то воно виконується, наприклад, подається звуковий сигнал, запускається двигун, виконується процедура і т.д.

На рис. 7.2 наведена схема, що ілюструє цикл роботи інтерпретатора.

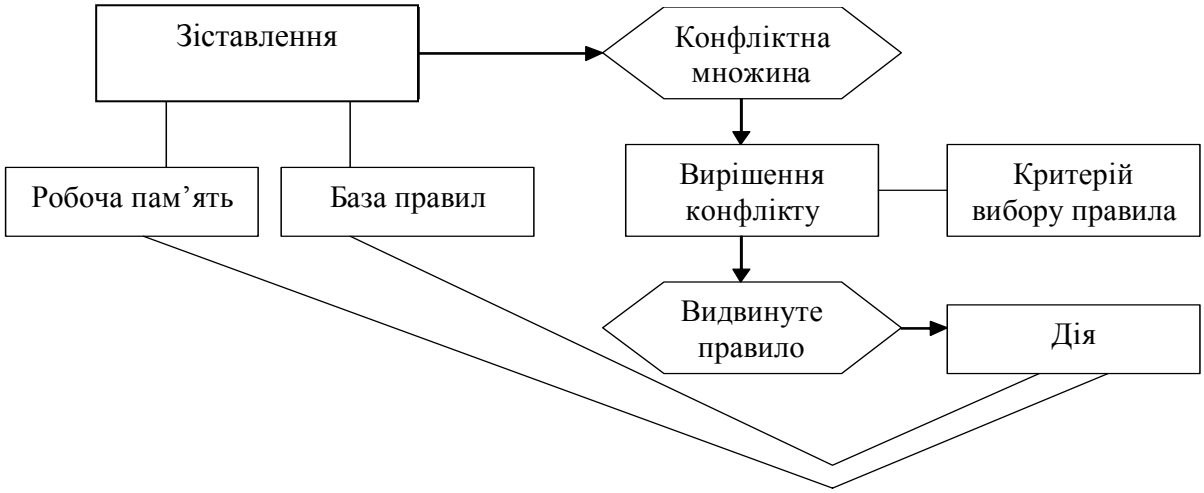

Рисунок 7.2 - Цикл роботи інтерпретатора

Нові дані, введені в систему правилом, що спрацювало, можуть в свою чергу змінити критерій вибору правила.

Приклад

Є комп'ютерна система, призначена для гри в шахи.

Якщо вона розігрує партію на двох гравців, то вона може ухвалити рішення дотримуватися атакуючої стратегії через рік, тобто атакувати буде один з партнерів.

Якщо користувач сам грає з системою, то в якийсь момент вона може перейти до оборонної тактики (принаймні тимчасово), а потім знов повернутися до наступу.

Зміна критерію ґрунтується на висновках, одержаних після аналізу положення на дошці, яке подане в робочій пам'яті системи, на правилах гри (структурних статичних знаннях) і на структурних динамічних знаннях (евристиках).

Насправді ЕС не мають в своєму розпорядженні процедур, які могли б побудувати в просторі станів відразу весь шлях вирішення задачі. Більш того, часто навіть не вдається визначити, чи є взагалі яке-небудь рішення задачі. Проте, пошук рішення виконується, і траєкторія пошуку рішення повністю визначається даними.

При розробці управляючого компонента механізму виведення необхідно вирішити питання про те, за яким критерієм слід вибирати правило, яке буде застосоване в конкретному циклі.

Підсистема здобуття знань

Підсистема здобуття знань призначена для додавання в базу знань нових правил і модифікації тих, що існують. У її задачу входить приведення правила до вигляду, що дозволяє механізму виведення застосовувати його в процесі роботи.

У простому випадку в якості такої підсистеми може виступати простий редактор (або текстовий процесор), який заносить правила у файл.

Складніші системи мають засоби для перевірки правил, що вводяться, на несуперечність з тими, що є. У деяких системах здобуття знань здійснюється не одним, а декількома різними методами.

Приклад.

Розглянемо систему пошуку несправностей в електронних пристроях - DETEKTOR. Ця система розроблена фахівцями компанії Tektronix, які розрізняють три типи знань і для кожного типу відповідно мають три різні методи:

- операціональні знання, необхідні для вирішення задачі у теперішній момент;
- допоміжні, сюди відносяться факти, які експерт може і не пам'ятати, але зобов'язаний знати, де їх можна знайти;
- додаткові, відомості про зовнішні умови, в яких відбуватиметься вирішення задачі.

У системі DETEKTOR операціональні знання вводяться за допомогою "спеціалізованої граматики опису знань".

Граматика задає форму подання правил пошуку несправностей для їх введення в систему експертом, наприклад, ЯКЩО напруга у вузлі 4 рівна напрузі у вузлі 5, ТО резистор 2 не працює.

Для пошуку неполадок вводяться *допоміжні знання* – це такі знання, які використовуються для перевірки допустимості конкретного з'єднання, правильності принципових схем і розташування плати. Для цього потрібна спеціальна підсистема здобуття знань у формі зображення, яка здатна сприймати схему в графічному виконанні і перевіряти за нею правильність з'єднання елементів.

Відомості про зовнішні умови, в яких відбуватиметься процес вирішення задачі (додаткові знання) безпосередньо в систему не включаються, але для того, щоб спроектувати ефективну систему, їх необхідно ретельно проаналізувати.

Додаткові знання використовуються для обґрунтування і перевірки різних проектних рішень, наприклад, при включенні в електронний пристрій спеціальних схем, які б прискорили пошук несправностей.

Засоби спілкування природною мовою

Оскільки всі продукційні експертні системи реалізовані на ЕОМ, то і вхідна інформація сприймається на машинній мові. Але непідготовлений користувач цієї мови не знає, тому ЕС просто зобов'язана мати засоби спілкування природною мовою.

Переважна більшість систем, основаних на знаннях, має примітивний інтерфейс на природній мові: допустимі вхідні повідомлення користувача обмежені набором понять, що містяться в базі знань. З часом користувачам буде надана можливість спілкуватися з такими системами повними реченнями, що включають будь-які частини мови: іменники, дієслова, прийменники, додатки і ін.

У простих системах діалог з користувачем дозволяє обійтися словами "так", "ні" і іноді включити питання "чому?". У складніших системах програма взаємодії з користувачем дозволяє виробити граматичний розбір вхідного речення, який полягає у визначенні взаємопов'язаних частин речення. Складний алгоритм граматичного розбору визначає у введеному реченні підмет, присудок і інші граматичні члени речення.

Підсистема пояснення

Більшість фахівців-користувачів не зможе довіряти виведеному системою висновку, доки не знатимуть, як він був одержаний. Розглянемо життєву ситуацію: якщо лікар встановив у Вас наявність деякого захворювання, Вам обов'язково потрібно знати, як він дійшов такого висновку, на підставі яких аналізів або результатів обстеження. Ви можете попросити від нього висловити хід своїх міркувань, який і дозволив поставити такий діагноз.

До ЕС висуваються ті ж самі вимоги.

Компонент ЕС, який відповідає на питання користувача про те, як саме одержане рішення, називається *підсистемою пояснення*. Під час

проведення консультації ця підсистема повинна бути здатна у будь-який момент привести обґрунтування ухваленого рішення.

Висновки щодо структури систем продукційного типу

Враховуючи вище сказане, можна зробити деякі загальні висновки щодо структури систем продукційного типу:

1. Продукційна система або система основана на правилах, працює циклічно. У кожному циклі продукції правила з бази знань проглядаються інтерпретатором правила в певному порядку, який встановлюється його управляючим компонентом. Якщо виявляється правило, умова якого при зіставленні збіглася з деякими фактами з робочої пам'яті, то правило спрацьовує і його висновок додається в робочу пам'ять. Потім цикл повторюється.
2. Цикл має чотири фази: зіставлення, вибір, активізація і виконання дії (зміна стану робочої пам'яті).
3. Будь-яке правило є керованим модулем даних (КМД). КМД є автономним, незалежним від інших правил, елементом бази знань. Це дає можливість легко нарощувати і відлагоджувати програму, що складається з продукцій, оскільки при додаванні нового правила або редагуванні, не вимагається вносити зміни в решту правил. Кожному КМД присвоюється індивідуальний номер, що дозволяє досить просто викликати для перегляду ланцюжок правил, що спрацювали.
4. Система, основана на правилах, містить сукупність КМД, що складаються з двох частин: посилки (антецедента) і висновку (консеквента).
5. Механізм виведення може бути названий системою виведення, що використовує зіставлення за зразком.
6. Продукційна система – це система, основана на правилах, механізм виведення якої включає компоненти власне виведення і управління ним.

7.4 Семантичні мережі як модель подання знань. Використання семантичних мереж при розробці експертних систем

Одним з найбільш поширених способів подання знань є так звані семантичні мережі, які були створені спочатку як модель подання довгострокової пам’яті людини в психології. В словнику термін «*семантика*» визначається як зміст слова, художнього твору, дії і т.і. В науці-семіотиці під *семантикою* розуміють певні відношення між символами та об’єктами, які зображають ці символи. Спочатку в психології вивчали об’єкти, які назвали семантичними з точки зору асоціативних властивостей, які накопичувались в системі навчання та поведінки людини. З розвитком когнітивної психології виникло поняття семантичних структур, які включали певні об’єкти. Пізніше було

досліджено принцип дії пам’яті людини, висунуті гіпотези про структуру довгострокової пам’яті і створені моделюючі програми, які розуміли зміст слів. Наслідком цих досліджень і з’явилось поняття семантичної мережі.

Мережна структура була запропонована вченим Куілліаном для опису структури довгострокової пам’яті як спосіб подання семантичних відношень між концептами (словами). Вона моделювала природне розуміння і використання мови людиною. Тому її основою були опис значень класу, до якого належить об’єкт, його прототипу та встановлення зв’язків зі словами, які відображують властивості об’єкта. Наприклад, концептуальний об’єкт «студент» можна зобразити такою семантичною мережею:

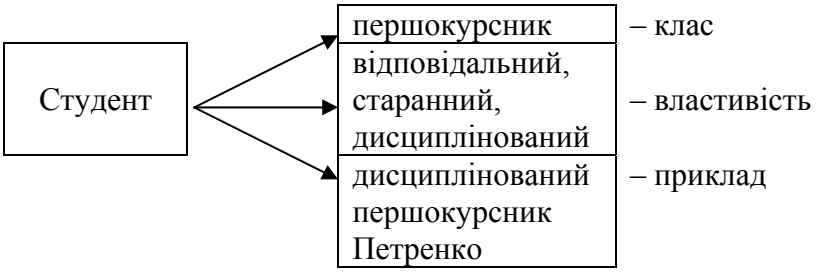

Рисунок 7.3 - Приклад семантичної мережі для зображення простого концептуального об’єкта

Тут визначені оператори відношень, які називаються класом, властивістю та прикладом, для яких описані значення. При цьому концептуальні об’єкти подані асоціативними мережами, які складаються з вершин, що відображають об’єкти, і дуг, які показують відношення між об’єктами. Наприклад, з відношенням клас асоціюється значення „першокурсник”, з властивістю – „відповідальний”, „дисциплінований” і т.і.

Таким чином, дані, які основані на фактах, можна подати в довгостроковій пам’яті за допомогою структур трьох типів: елементів, властивостей та покажчиків. Елемент має прототип (об’єкт, подію, поняття), властивість – атрибут і значення, а покажчики встановлюють зв’язки між елементами та властивостями.

Семантичні мережі дозволяють реалізувати такі важливі можливості:

1. *Одночасне зображення «знаків» і «типів».*

Під *знаком* розуміється конкретне значення, або екземпляр об’єкта, а під *типом* – клас подібних знаків. Узагальнення знаків в типи є найпростішою операцією абстрагування, що необхідна для кращого розуміння складних об’єктів. Абстракція може бути багаторівневою і може породжувати складні типи. Наприклад, тип «домашнє господарство» містить об’єкти типу «посуд», «меблі», «одяг» і т.і. Тип «меблі», в свою чергу, може включати «стіл», «стул», «крісло» і т.і. Породження типів за

допомогою абстракції дозволяє краще зрозуміти властивості, взаємозв'язки та загальні способи застосування знаків.

2. *Створення ієрархій об'єктів*.

До основних операцій абстрагування відносять: ідентифікацію, узагальнення та агрегацію. Узагальнення, яке дозволяє співвіднести множину знаків до одного типу називається *класифікацією*, а узагальнення, яке дозволяє співвіднести множину типів до одного типу – власне, *узагальненням*. Наприклад, подання окремих службовців одним типом СЛУЖБОВЕЦЬ є класифікацією, а подання типів СЛУЖБОВЕЦЬ і УЧЕНЬ одним типом ОСОБА – це узагальнення. Узагальнення акцентує увагу на подібності об'єктів, абстрагуючись від їх відмінностей.

Узагальнений тип має всі властивості, які є загальними для базових об'єктів і типів, тобто всі властивості узагальненого типу можуть бути успадковані базовими типами. Але можна і заборонити успадкування певних властивостей, які характеризують тип в цілому і не успадковуються підтипами або базовими знаками. Узагальнення створює ієрархію вигляду „Є_ДЕЯКИЙ" і відображає той факт, що одним типом об'єкта виконується узагальнення іншого типу, наприклад, „СЛУЖБОВЕЦЬ Є_ДЕЯКА ОСОБА".

Для конструювання об'єктів з інших базових об'єктів як на рівні знаків, так і на рівні типів, використовується операція *агрегації*, яка відображає ієрархію вигляду „Є_ЧАСТИНА". Агрегація розкриває структуру об'єкта. Наприклад, тип „СЛУЖБОВЕЦЬ" можна сконструювати з типів-властивостей „ПРІЗВИЩЕ", „РІК_НАРОДЖЕННЯ", „АДРЕСА". Такі властивості, які тлумачать поняття, називають *інтенсіональними*. Конкретну ж реалізацію типу „СЛУЖБОВЕЦЬ", конструюють з відповідних знаків, наприклад, „Іванов, 1970, м. Вінниця, Соборна,15". Такі властивості є фактичними значеннями і називаються *екстенсіональними*. В семантичних моделях екстенсіональні знання використовуються як основа бази даних, а інтенсіональні – як основа бази знань.

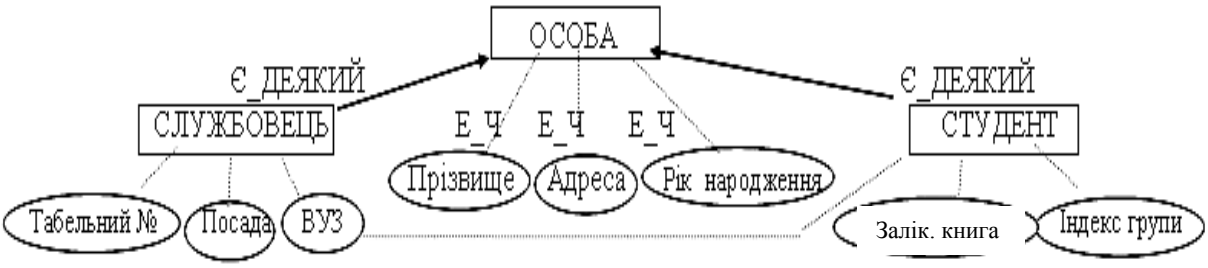

Рисунок 7.4 - Приклад семантичної мережі для подання ієрархії типів „ОСОБА", „СЛУЖБОВЕЦЬ", „СТУДЕНТ"

3. *Використання концепції «ролі».*

Узагальнення дозволяє будувати складні системи категорій, структура яких не обов'язково повинна бути деревовидною. Наприклад, в наведеній мережі тип „ОСОБА" може виступати, в ролі „СЛУЖБОВЕЦЬ", „ВІДПРАВНИК_ВАНТАЖУ", „ОДЕРЖУВАЧ_ВАНТАЖУ":

Відправником або одержувачем вантажу може бути і організація, яка виступає також в ролі виробника.

4. *Обчислення семантичних відстаней.*

Може здійснюватись шляхом визначення числа дуг в семантичній мережі на шляху від однієї вершини до іншої. В деяких семантичних мережах можна явно задавати велику семантичну відстань за допомогою спеціальних дуг нерелевантності, які відображають відсутність взаємозв'язків між двома вершинами.

5. *Відображення часу.*

Здійснюється шляхом приписування відповідним об'єктам атрибута, тобто вершини-характеристики – «час початку» та «час закінчення».

Розробимо за допомогою даної моделі зображення такої ситуації : «Іванов, 1970 року народження, працює програмістом в обчислювальному центрі, який знаходиться в корпусі А».

Спочатку потрібно зобразити цю ситуацію у вигляді однієї з можливих сукупностей простих бінарних фактів (тверджень):

1. Іванов працює програмістом.
2. Іванов працює в обчислювальному центрі (ОЦ).
3. Іванов 1970 року народження.
4. ОЦ знаходиться в корпусі А.

Тепер необхідно визначити вершини і дуги семантичної мережі.

1. Вершини-події: ПРАЦЮЄ.
2. Вершини-концепти: ІВАНОВ, ПРОГРАМІСТ, ОЦ, КОРПУС А.
3. Вершини-характеристики РІК НАРОДЖЕННЯ.
4. Вершини-значення 1970.

Визначимо дуги:

1) дуга, яка вказує на агента ПРАЦЮЄ-ІВАНОВ;
2) дуга, яка вказує на об'єкт впливу ПРАЦЮЄ-ПРОГРАМІСТОМ;
3) дуги, які зображають твердження ПРАЦЮЄ_В (ІВАНОВ,ОЦ), ЗНАХОДИТЬСЯ_В (ОЦ,КОРПУС_А).

Проблема проектування варіанта семантичної мережі є дуже складною, оскільки одну і ту ж інформацію можна подати багатьма способами. В цьому і полягає основна перевага і основний недолік семантичних мереж.

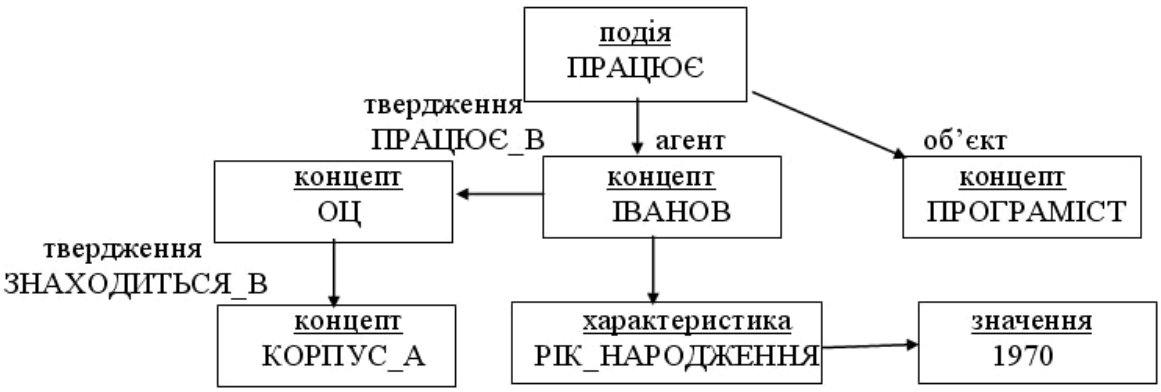

Рисунок 7.5 - Результат проектування семантичної мережі

Розглянемо тепер один з найбільш загальних підходів до побудови процедури виведення на семантичних мережах, оснований на зіставленні частин структури мережі. Він полягає в побудові підмережі, яка описує запит, з подальшим зіставленням цієї підмережі з базою даних семантичної мережі. Наприклад, запит, «Ким працює Іванов» буде зображено у вигляді такої підмережі:

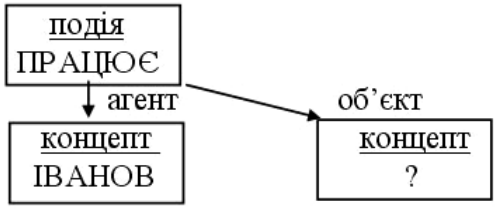

Рисунок 7.6 - Результат проектування підмережі для процедури виведення

Далі система управління виконує зіставлення, внаслідок якого формується відповідь: „ПРОГРАМІСТ”.

Розглянемо більш формальне визначення семантичної мережі через поняття інтенсіоналу та екстенсіоналу. Нехай задано кінцеву множину символів $A=\{A_1, \ldots, A_n\}$ і кінцеву множину відношень $R=\{R_1, \ldots, R_m\}$. Схемою або інтенсіоналом відношення R називається набір пар:

$$INT(R_i)=\{\ldots |A_j, DOM(A_j)|, \ldots\},$$

де R - ім’я відношення;

$DOM(A_j)$ – домен A_j, тобто множина значень атрибута A_j відношення R_j.

Об’єднання всіх доменів є екстенсіоналом відношення R_j, тобто множиною

$$EXT(R_i)=\{F_i, \ldots, F_p\},$$

де F_k (k=1,p) – факт відношення R_i.

Факт задається сукупністю пар типу «атрибут-значення». Під *фактом* розуміється конкретизація певного відношення між зазначеними об’єктами. В графічній інтерпретації факт – це підграф семантичної мережі, який має зіркоподібну структуру. З вершини факту виходять ребра, які позначаються іменами атрибутів факту й спрямовуються до вершин базової множини, що є значеннями цих атрибутів.

Розглянемо приклад в якому базовою множиною моделі (тобто множиною об’єктів) є цілі числа {0,1,2}, а множиною відношень множина R={<, +}. Тоді інтенсіонали відношень можна записати у вигляді

INT(+)={[перший доданок (0,1,2)],[другий доданок (0,1,2)],[сума(0,1,2)]},
INT(<)={[менше(0,1)],[більше(1,2)]}.

Екстенсіонали відношень можна задати у вигляді фактів:
F_1(<, менше 0, більше 1);
F2(<, менше 1, більше 2);
F_n(+, перший доданок 0, другий доданок 1, сума 1) і т. і.

Таким чином інтенсіональна семантична мережа описує ПО на узагальненому, концептуальному рівні, а екстенсіональна – здійснює конкретизацію і наповнення конкретними фактами.

Класифікація семантичних мереж

Класифікація семантичних мереж за кількістю типів відношень:
- однорідні (з єдиним типом відношень);
- неоднорідні (з різними типами відношень).

Класифікація семантичних мереж за типом відношень.
- бінарні (відношення пов'язують два об'єкти);
- парні (у яких є спеціальні відношення, що пов'язують більше двох понять).

Найчастіше в семантичних мережах використовуються такі відношення:
- зв'язки типу "частина-ціле" ("підклас-клас", "елемент-множина", „підвид-вид” і т.д.);
- функціональні зв'язки (описуються за допомогою дієслів, таких як "виробляє", "впливає"...);
- кількісні (більше, менше, рівне...);
- просторові (далеко від, близько від, за, під, над,...)
- тимчасові (раніше, пізніше, протягом,...);
- атрибутивні (мати властивість, мати значення, ...)
- логічні зв'язки (і, або, ні,...) і т.д.

Як і в системах, основаних на фреймах, в семантичній мережі можуть бути подані родовидові відношення, які дозволяють реалізувати

спадкоємство властивостей від об'єктів-батьків. Ця обставина приводить до того, що семантичні мережі мають всі переваги і всі недоліки фреймової моделі знань.

Основна перевага цієї моделі – це відповідність сучасним уявленням про організацію довготривалої пам'яті людини. Основні недоліки моделі - складність пошуку висновку в семантичній мережі, складність обробки виключень.

Для реалізації семантичних мереж існують спеціальні мережеві мови, наприклад, „NET”.

Висновки: Семантичні мережі можуть мати найрізноманітніші структури для зображення семантики. Але складність створення відповідних засобів висунула задачу пошуку шляхів, які б дозволили внести в мережу деяку регулярність з метою полегшення її програмування. Одним з таких шляхів є фреймове подання знань.

7.5 Фреймові моделі подання знань. Основи теорії фреймів

Фрейм (від англ. frame - каркас, рамка) запропонований вченим М. Мінським в 70-ті роки XX ст. як структура знань для сприйняття просторових станів. Ця модель, як і семантична мережа має глибоке психологічне обґрунтування [8].

Фреймом називається структура для опису абстрактного образу або стереотипної ситуації, що складається з характеристик цього образу (ситуації) і їх значень. Характеристики називаються *слотами*, а значення - *заповнювачами слотів*.

У психології і філософії відоме поняття *абстрактного образу*. Наприклад, слово "кімната" викликає у слухаючів образ кімнати: житлове приміщення з чотирма стінами, підлогою, стелею, вікнами і дверима, площею 10-30 кв.м. З цього опису нічого не можна прибрати (наприклад, прибравши вікна, одержимо не кімнату, а комору), але в цьому описі є "дірки" або "*слоти*" – незаповнені значення деяких атрибутів: кількість вікон, колір стін, висота стелі, покриття підлоги і т.д. Слот може містити не лише конкретне значення, але і ім'я процедури, що дозволяє обчислити його за заданим алгоритмом, а також одну або декілька евристик, за допомогою яких це значення можна знайти [33].

У теорії фреймів такий образ називається *фреймом*. Фреймом називається також і формалізована модель для відображення образу.

Проте повернемось до основоположних в теорії фреймів досліджень вченого Марвіна Мінського. Згідно з його означенням, *фрейми* – це мінімальні структури інформації, необхідні для зображення класу об’єктів, явищ або процесів. В основі теорії фреймів, яка також відноситься до області психології, лежить припущення про те, що сприйняття фактів людиною здійснюється шляхом зіставлення отриманої ззовні інформації з конкретними елементами та значеннями, а також з фреймами, які

визначені для кожного концептуального об’єкта в пам’яті людини. Коли людина потрапляє в нову ситуацію, вона викликає з пам’яті основну структуру, що називається фреймом. Тобто *фрейм* – це одиниця подання знань, яка була запам’ятована в минулому, і деталі якої можуть бути змінені при необхідності у відповідності до поточної ситуації.

Фрейм – це структура даних, за допомогою якої можна створити опис, наприклад, деякої кімнати, або місця, де можна було б відпочити. Кожен фрейм можна доповнювати різною інформацією, яка може мати відношення до способів застосування даного фрейму, наслідків його застосування, дій, які необхідно виконати у разі, якщо не виправдається прогноз і т.і. Складні об’єкти можуть зображатися комбінацією декількох фреймів, тобто відповідати мережі з фреймів.

В загальному випадку структуру фрейму можна зобразити у такому вигляді:

(ім'я 1-го слота: значення 1-го слота),
(ім'я 2-го слота: значення 2-го слота),
.....
(ім'я N-го слота: значення N-го слота)

Той же запис можна зобразити у вигляді таблиці, доповнивши його двома стовпцями:

Таблиця 7.2 – Табличне зображення структури фрейму

Ім'я фрейму			
Ім'я слота	Тип слота	Значення слота	Приєднана процедура
...	...	...	...

У таблиці додаткові стовпці призначені для опису типу слота і можливого приєднання до того або іншого слота спеціальних процедур, що допускається в теорії фреймів.

Також той же запис можна зобразити у графічному вигляді такою схемою:

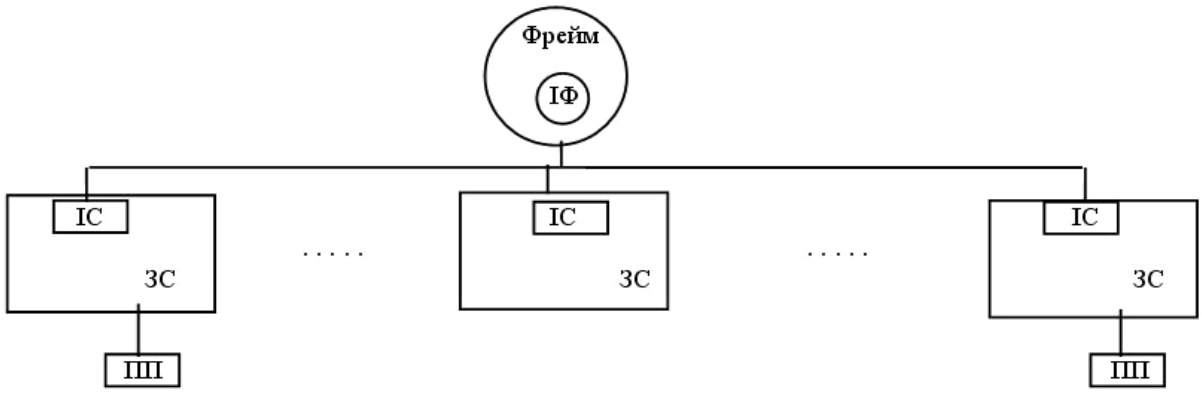

Рисунок 7.7 - Графічне зображення структури фрейму

і описати таким рядком:

(ІФ, (ІС,ЗС,ПП)...,(ІС,ЗС,ПП)),

де ІФ – ім’я фрейму,
ІС – ім’я слота,
ЗС – значення слота,
ПП – ім’я приєднаної процедури (необов’язковий параметр).

Слоти – це деякі незаповнені підструктури фреймів, заповнення яких приводить до того, що даний фрейм ставиться у відповідність до деякої конкретної ситуації, факту або явища.

На найвищих рівнях фреймів подається фіксована інформація, - факти, які відносяться до стану об’єкта і вважаються істинними. На більш низьких рівнях розташовано множину так званих термінальних слотів, які обов’язково повинні бути заповнені конкретними даними та значеннями.

З кожним фреймом пов’язується інформація: як використовувати фрейм, що робити, якщо трапиться щось непередбачене, відсутні значення для слотів. Фрейм із заповненими слотами називається *екземпляром фрейму*. Для організації зв’язків між об’єктами ПО будується мережа фреймів. Зв’язок може бути організовано шляхом використання імен інших фреймів як значень деяких слотів одного фрейму.

Таким чином у фреймових системах можна виділити три основних процеси:

- *Створення екземпляра фрейму.*

Для цього необхідно знайти придатний фрейм і заповнити його слоти інформацією, яка описує специфіку ситуації, що розглядається.

- *Активація фреймів.*

Якщо фрейм визнається придатним для опису даної ситуації, здійснюється його активація. Якщо знаходиться занадто багато відмінностей змісту фрейму від специфічних особливостей ситуації, що розглядається, виконується пошук іншого, більш придатного фрейму. При цьому відкидуваний фрейм може містити підказку на те, які саме фрейми необхідно досліджувати замість нього (наприклад більш загальні, чи, навпаки, більш спеціалізовані).

- *Організація виведення*, яка полягає в послідовному пошуку і активації в мережі фреймів до моменту знаходження найбільш відповідного і побудові на його основі екземпляра фрейму.

Вчений Террі Виноград запропонував об’єднати у фреймах переваги декларативного та процедурного подання знань. При цьому знання, які описують ситуацію, зберігаються у фреймах в декларативній формі, а знання про використання фреймів зберігаються у процедурній формі.

Наприклад, процедури можуть зберігати знання, які дають відповіді на такі питання:

- Коли активізувати фрейм? Подібно до демонів, фрейми можуть самі активізувати себе у випадку, коли розпізнана відповідна ситуація.
- В якому випадку вважати, що даний фрейм неадекватний ситуації, і що при цьому робити? Фрейм, наприклад, може передати управління іншому фрейму або деактивізувати себе.
- Коли здійснювати заповнення слотів, в момент виклику чи пізніше, по мірі необхідності?

Групи подібних фреймів об'єднують в системи фреймів, які містять опис залежностей (причинно-наслідкових, часових та інших), між фреймами системи. Наслідки деяких суттєвих дій викликають певні перетворення між фреймами системи. При цьому забезпечується ефективний перехід від одного стану до іншого без обчислення заново значень всіх параметрів ситуації, яка змінилась, а лише перераховуються параметри, що змінились, або нові параметри. Наприклад, при аналізі зорових сцен різні фрейми системи можуть відображати різні точки зору на об'єкт. Це ж відноситься і до аналізу ситуацій прийняття рішень.

Великою перевагою фреймів є їх можливість зображення очікувань та інших типів припущень. Слотам фрейму можуть бути заздалегідь приписані за замовчуванням деякі стандартні значення. Це дозволяє аналізувати за допомогою фреймів ситуації, в яких відсутній опис цілого ряду деталей. Стандартні значення, що приписані за замовчуванням, закріпляються не жорстко і можуть бути легко замінені при надходженні більш придатних значень, якщо вони знайдені в описі ситуації, що обробляється.

Таким чином до основних властивостей фреймів відносять:

1. Наявність базового типу, який містить найважливіші деталі класу об'єктів (ситуацій). Фрейми більш низького рівня містять лише специфічні особливості конкретних ситуацій і покажчики на базовий фрейм.
2. Застосування процесу зіставлення для здійснення перевірки правильності вибору фрейму.
3. Ієрархічна структура.
4. Організація міжфреймових мереж. Дозволяє шукати відповідний даній ситуації фрейм за допомогою покажчиків відмінностей, які забезпечують створення мережі подібних фреймів.
4. Наявність значень за замовчуванням.
5. Відображення відношень «абстрактне-конкретне» та «ціле-частина».

Іноді слот включає компонент, що називається *фасетом*, який задає діапазон або перелік його можливих значень. Фасет вказує також граничні значення заповнювача слота.

Як вже сказано, в слоті можуть зберігатися процедури і правила, які викликаються при необхідності обчислення цього значення.

Якщо, наприклад, фрейм, що описує людину, включає слоти „ДАТА НАРОДЖЕННЯ” і „ВІК” і в першому з них знаходиться деяке значення, то в другому слоті може стояти ім'я процедури, що обчислює вік за датою народження і поточною датою.

Процедури, розташовані в слоті, називаються *зв'язаними процедурами*.

Розрізняють фрейми-зразки, або прототипи, що зберігаються в базі знань і фрейми-екземпляри, які створюються для відображення реальних ситуацій на основі даних, що постійно надходять.

Модель фрейму є достатньо універсальною, оскільки дозволяє відобразити всю різноманітність знань про світ через:

- фрейми-структури, для позначення об'єктів і понять (займ, застава, вексель);
- фрейми-ролі (менеджер, касир, клієнт);
- фрейми-сценарії (банкрутство, збори акціонерів, святкування іменин);
- фрейми-ситуації (тривога, аварія, робочий режим пристрою і т.д.).

Також, однією з найважливіших властивостей теорії фреймів є запозичене з теорії семантичних мереж спадкоємство властивостей.

У фреймах і в семантичних мережах спадкоємство відбувається з використанням АКО-зв'язків („A-Kind-of” – „це”). Слот АКО вказує на фрейм вищого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Сукупність фреймів, що моделює будь-яку предметну область, є ієрархічною структурою, в якій фрейми з'єднуються за допомогою родовидових зв'язків.

Фрейми мають здатність успадковувати значення характеристик своїх батьків, що знаходяться на вищому рівні ієрархії.

Приклад. Розглянемо приклад спадкоємства (наслідування) властивостей:

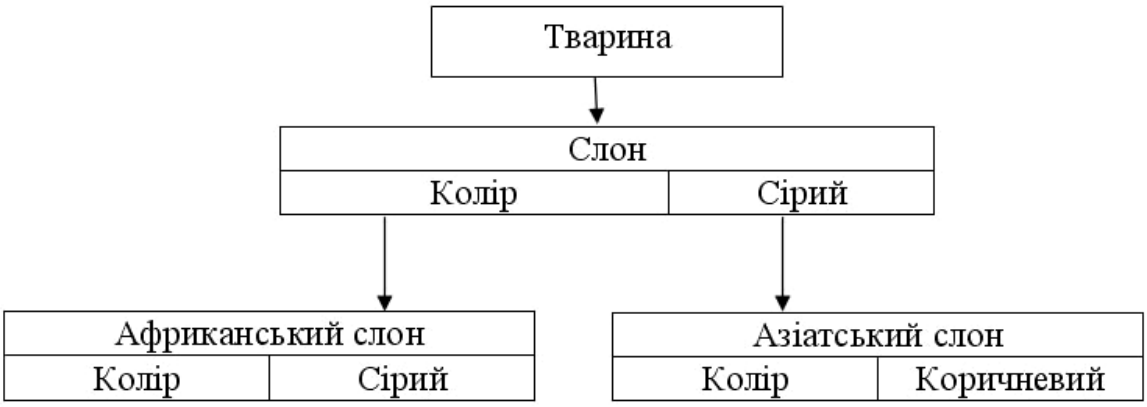

Рисунок 7.8 - Приклад властивості наслідування в фреймах

У прикладі фрейм „АФРИКАНСЬКИЙ СЛОН” успадковує від фрейма „СЛОН” зі значень „СІРИЙ” характеристику „КОЛІР”.

Значення характеристик фреймів можуть передаватися за замовчуванням фреймам, що знаходяться нижче за них в ієрархії, але якщо останні містять власні значення даних характеристик, то як істинні приймаються саме вони. Ця обставина дозволяє легко враховувати у фреймових системах різного роду винятки.

Приклад:

Рисунок 7.9 - Приклад винятку із властивості наслідування в фреймах

Поняття "УЧЕНЬ" успадковує властивості фреймів "ДИТИНА" і "ЛЮДИНА", які знаходяться на вищому рівні ієрархії. Так, на питання чи "Люблять учні солодке?" слідує відповідь "Так", оскільки цією властивістю володіють всі діти, що вказано у фреймі "ДИТИНА".

Спадкоємство властивостей може бути частковим, так "ВІК" для учнів не успадковується з фрейма "ДИТИНА", оскільки вказаний чітко в своєму власному фреймі.

Розрізняють *статичні* і *динамічні* системи фреймів. В перших фрейми не можуть бути змінені в процесі вирішення задачі, в других – це допустимо.

Про системи програмування, основані на фреймах, відзначають, що вони є об'єктно-орієнтованими. Кожен фрейм відповідає деякому об'єкту предметної області, а слоти містять дані, що описують цей об'єкт (у слотах знаходяться значення ознак об'єктів).

Фрейм може бути зображений у вигляді списку властивостей, а якщо використовувати засоби баз даних, то у вигляді запису.

Найчіткіше переваги фреймових систем подання знань виявляються в тому випадку, якщо родовидові зв'язки змінюються дуже рідко і предметна область налічує небагато винятків.

У фреймових системах дані про родовидові зв'язки зберігаються явно як і знання всіх інших типів.

Значення слотів подаються в системі в єдиному екземплярі, оскільки включаються тільки в один фрейм, що описує найзагальніше поняття зі всіх тих, які містить слот з даним ім'ям. Така властивість систем фреймів дає можливість зменшити обсяг пам'яті, проте основним призначенням родовидової ієрархії є не економія пам'яті, а подання в базі знань семантичних зв'язків, існуючих між поняттями предметної області.

Ще одна перевага фреймів полягає у тому, що значення будь-якого слота при необхідності може бути обчислене за допомогою відповідних процедур або знайдених евристичними методами. Для цього інженер зі знань повинен наперед розробити всі необхідні процедури і евристичні методи, щоб включити їх в систему на етапі її проектування. Перевагою фреймів як моделі подання знань є здатність відображати концептуальну основу організації пам'яті людини, а також її гнучкість і наочність.

Як недолік фреймових систем слід зазначити їх відносно високу складність, що виявляється в зниженні швидкості роботи механізму логічного виведення і в збільшенні трудомісткості внесення змін в родовидову ієрархію. У фреймових системах також затруднена обробка винятків.

7.6 Модель дошки оголошень. Принципи організації систем з дошкою оголошень

В останні роки в розробці архітектури експертних систем з'явився новий напрямок, що одержав назву *системи з дошкою оголошень* (blackboard systems). Системи з такою архітектурою можуть емулювати режим побудови як прямого ланцюжка логічного виведення, так і зворотного, а також поперемінно застосовувати ці режими в процесі роботи. Крім того, застосування систем з дошкою оголошень спонукає інженерів зі знань до ієрархічної організації знань щодо предметної області і простору часткових і повних рішень. Таким чином, ця архітектура дуже добре підходить для вирішення задач проектування, для яких характерний великий, але факторизовний багатовимірний простір рішень. Системи з подібною архітектурою вже успішно застосовувалися для інтерпретації даних (наприклад, розпізнавання графічних зображень і мови), аналізу і синтезу багатокомпонентних структур (наприклад, структури протеїнів) і планування.

Але слід зазначити, що реалізація і впровадження систем з дошкою оголошення – досить складний процес, а самі системи вимагають значних обчислювальних ресурсів.

Принципи організації систем з дошкою оголошень

В основу організації систем цього типу покладена така ідея [8].

- Уявіть собі групу експертів, що сидять біля класної дошки (чи великої дошки оголошень) і намагаються вирішити яку-небудь проблему.
- Кожен експерт є фахівцем у якійсь певній області, що має відношення до вирішення проблеми.
- Формулювання проблеми і вихідних даних записані на дошці.
- Експерти допитливо вдивляються в те, що написано на дошці, і кожний з них думає над тим, чим він може допомогти у вирішенні проблеми.
- Якщо хто-небудь з експертів почуває, що йому є що сказати з цього приводу, він виконує відповідні обчислення і записує результат на тій же дошці.
- Цей новий результат може дозволити й іншим експертам внести певний вклад у вирішення проблеми.
- Процес припиняється (а експерти розходяться), коли проблема буде вирішена.

Така методика спільного вирішення проблем буде ефективна, якщо дотримуються визначені заздалегідь домовленості, а саме:

- всі експерти повинні говорити однією і тією ж мовою, хоча при записі результатів на дошці можуть використовуватися і різні схеми позначень;
- повинен існувати певний протокол визначення черговості "виступів", що набирає сили в ситуації, коли відразу кілька експертів „хапаються” за крейду і направляються до дошки.

Ці домовленості є не що інше, як уже знайомі нам схеми *подання і керування*.

Якщо розглядати працюючу за таким принципом систему з погляду організації процесу обчислень, то в системі можна виділити такі структурні компоненти. Знання про предметну область розділені між незалежними *джерелами знань (KS - knowledge sources)*, що працюють під керуванням *планувальника (scheduler)*. Рішення формується в деякій глобально доступній структурі, яку ми будемо називати *дошкою оголошень (blackboard)*. Таким чином, у цій системі всі знання "як зробити" будуть зображені не у вигляді єдиного набору правил, а у вигляді набору програм. Кожний з компонентів цього набору може мати у своєму розпорядженні власний набір правил або поєднання правил і процедур.

Функції дошки оголошень багато в чому подібні з функціями робочої пам'яті в продукційних системах, але її організаційна структура значно складніша. Як правило, дошка оголошень розділяється на кілька *рівнів* опису, причому кожен рівень відповідає певному ступеню деталізації. Дані в межах окремих рівнів дошки оголошень зображають ієрархії об'єктів чи графи, тобто структури більш складніші, ніж вектори, що використовувалися в робочій пам'яті продукційних систем. У найсучасніших системах може бути навіть кілька дошок оголошень.

Джерела знань формують об'єкти на дошці оголошень, але це виконується за допомогою планувальника. Як правило, *записи активізації*

джерел знань (knowledge source activation records) містяться в спеціальному списку вибору (agenda), звідки їх отримує планувальник. Джерела знань спілкуються між собою тільки через дошку оголошень і не можуть безпосередньо передавати дані один одному чи запускати виконання будь-яких процедур. Тут є певна аналогія з організацією роботи продукційних систем, у яких правила також не можуть безпосередньо активізувати одне одне: все повинно проходити через робочу пам'ять.

Система HEARSAY

Архітектура на основі дошки оголошень розвилась з розробленої наприкінці 70-х років XX ст. системи розпізнавання мови HEARSAY [8]. Програмування комп'ютера з метою розпізнавання мови – це одна з найбільш складних задач, за які коли-небудь бралися фахівці в галузі штучного інтелекту. Її вирішення вимагає:

- складної обробки сигналів;
- зіставлення фізичних характеристик звукових сигналів із символічними елементами природної мови;
- виконання пошуку у великому просторі можливих інтерпретацій, у котрому об'єднані ці різні за своєю природою елементи.

Для вирішення цієї проблеми була обрана методика, основана на виділенні декількох рівнів абстракції опису аналізованих даних. Найнижчим є рівень фізичних акустичних сигналів, на якому формується звуковий спектр аналізованих сигналів. На наступних рівнях інформація проходить через нашарування лінгвістичних абстракцій з рівнем спільності понять, що збільшуються, – фонеми, силаби (співзвуччя), морфеми, слова, вирази і пропозиції.

Приступаючи до розробки системи, її творці прекрасно розуміли, що з кожним рівнем аналізу пов'язана окрема галузь знань – аналіз звукових сигналів, фонетика, лексичний аналіз, граматика, семантика, ораторське мистецтво. Жодна з цих галузей окремо не здатна надати достатньо інформації для того, щоб вирішити проблему. Уявимо, наприклад, що, користуючись методами обробки акустичних сигналів, ми змогли розкласти вихідний звук на фонеми. Але без додаткової інформації все одно не вдасться виділити зміст виразів, подібних таким: „I scream” (я кричу) і „ice cream” (морозиво) чи „please let us know” (будь ласка, дайте нам знати) і „please lettuce no” (будь ласка, без салату). Таким чином, хоча кожен окремий вид (набір) знань відіграє істотну роль у вирішенні проблеми і кожний з них може бути зображений у програмі більш-менш незалежно від інших, автоматичне розпізнавання мови вимагає використання всіх цих знань разом.

При розпізнаванні мови дослідникам приходиться стикатися ще з однією проблемою, що також можна віднести до числа ключових, - проблемою невизначеності. Вона виявляється на всіх рівнях подання інформації:

- дані неповні і „зашумлені”;
- відсутня однозначна відповідність між даними на сусідніх рівнях; прикладом може служити відповідність між рівнями фонем і лексичних одиниць при аналізі фраз „I scream” та „ice cream”;
- важливу роль відіграють лінгвістичний і змістовний контексти; інтерпретація сусідніх елементів робить більш-менш ймовірними різні варіанти інтерпретації поточного сегмента.

Більш традиційні підходи до розпізнавання мови основані на використанні статистичних моделей з теорії передачі інформації для визначення кореляційного зв'язку між сегментами. Підхід, що базується на знаннях, вимагає істотного перегляду методів обробки невизначеності.

У роботі [8] перераховані такі вимоги, які повинна задовольняти система розпізнавання мови, яка працює ефективно і основана на знаннях.

1. Із усіх можливих послідовностей операцій (частин рішень) хоча б одна повинна приводити до коректної інтерпретації.
2. Процедура аналізу наявних варіантів інтерпретації повинна давати коректному варіанту більш високу оцінку, ніж іншим конкуруючим варіантам. Іншими словами, правильна інтерпретація з урахуванням вимови повинна бути оцінена вище, ніж інші варіанти інтерпретації, які не враховують особливостей індивідуальної дикції.
3. Обчислювальні ресурси (пам'ять і час обчислень), необхідні для відшукання правильної інтерпретації, не повинні перевищувати певний поріг. Система розпізнавання, яка через пару днів видасть результат, нехай і правильний, і буде вимагати пам'яті обсягом декілька гігабайт, навряд чи кому-небудь буде потрібна.

У наведеному списку перша і третя вимоги у певній мірі суперечать одна одній. Для того, щоб коректне рішення із самого початку було присутнє в області гіпотез, вона повинна бути досить великою на стадії формування гіпотез, що при великому словнику може призвести до комбінаторного вибуху розв’язків. Вихід може бути знайдений тільки при використанні надзвичайно винахідливих евристик. Таким чином, найважливішою передумовою досягнення успіху в створенні такої системи є розробка придатної процедури оцінювання варіантів (друга з перерахованих вище вимог).

Використання джерел знань у HEARSAY-II

Для генерації, комбінування і розвитку гіпотез інтерпретації в системі HEARSAY-II використовується кілька джерел знань. Створені гіпотези (інтерпретації) різного рівня абстракції зберігаються на дошці оголошень.

Кожне джерело знань можна вважати в першому наближенні набором пар "умова-дія", хоча вони можуть бути реалізовані й у формі, яка відрізняється від породжуючих правил (наприклад, умови і дії можуть бути в дійсності довільними процедурами). Потік керування в цій системі

також відрізняється від потоку керування в продукційних системах. Замість того щоб у кожному циклі інтерпретатор аналізував виконання умов, специфікованих у джерелах знань, джерела знань завчасно повідомляють про активізовані в цих умовах, сповіщаючи, який вид модифікації даних буде впливати на виконання цих умов. У результаті система керується перериваннями, а цей режим керування значно ефективніший, ніж режим циклічного перегляду стану, що є основним для продукційних експертних систем. Такий режим нагадує використання демонів у фреймових системах, де потік керування регулюється відновленням даних.

Джерела знань пов'язуються з рівнями дошки оголошень у такий спосіб. Умови, специфіковані в джерелі знань, будуть задовольнятися в результаті відновлення даних на певному рівні дошки оголошень. Джерело знань також може записувати дані у певний рівень, причому не обов'язково в той же, котрий впливає на виконання умов. Більшість джерел знань у системі HEARSAY-II організовано так, що вони розпізнають дані на певному рівні: лінгвістичного аналізу, а виконувані ними операції відносяться до наступного за порядком рівня. Наприклад, деяке джерело активізується даними на силабічному рівні і формують лексичну гіпотезу на рівні слів.

У трохи спрощеному вигляді архітектура системи HEARSAY-II зображена на рис. 7.10. Стрілки, спрямовані від рівнів дошки оголошень до джерел знань, вказують, дані якого рівня змінюють виконання умов, специфікованих у джерелі знань. Стрілки в зворотному напрямку вказують, на який рівень розташовує дані те чи інше джерело знань. Відгалуження від стрілки "дії" джерела знань до монітора дошки оголошень означає, що зміна даних, яка виконується одним джерелом знань, фіксується в моніторі і потім використовується планувальником для активізації іншого джерела знань.

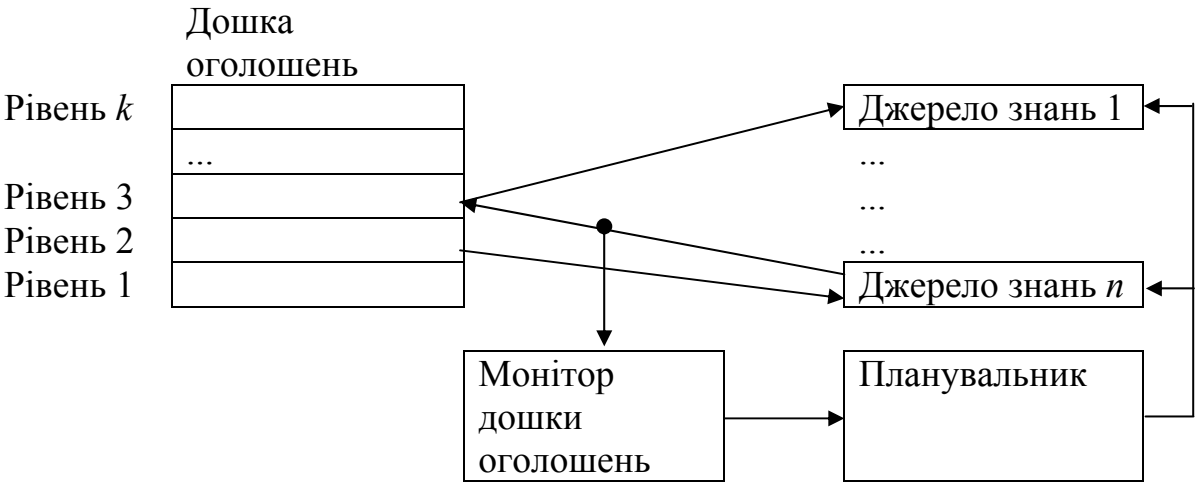

Рисунок 7.10 - Спрощена структурна схема системи дошки оголошень „HEARSAY-II”

Найголовніша відмінність архітектури з дошкою оголошень від усіх, розглянутих раніше, полягає в тому, що така система не диктує проектувальнику певний режим керування знаннями в системі, наприклад спадну чи висхідну стратегію побудови міркувань. Наприклад, у тій галузі, для якої створювалася система HEARSAY-II, можна застосовувати і спадну стратегію – будувати гіпотези про слова, а потім шукати підтвердження цим гіпотезам на рівні фонем, а можна і висхідну – збирати гіпотези про фонеми і формувати за ними гіпотези про слова. Яке джерело знань буде активізоване, визначається монітором і планувальником системи, а це рішення можна зробити або незалежним від предметної області, тобто від відповідних джерел знань, або залежним від них. Тут архітектура системи ніяк не обмежує розробника у виборі проектного рішення.

Система HEARSAY-III – оболонка для створення систем з дошкою оголошень

Система HEARSAY-III – це оболонка системи з дошкою оголошень, створена на базі HEARSAY-II точно так само, як оболонка продукційної системи EMYCIN була створена на базі MYCIN [8]. У структуру HEARSAY-III, крім джерел знань і дошки оголошень, включена ще й реляційна база даних, за допомогою якої виконується обслуговування об'єктів дошки оголошень і планування. Це дозволило істотно спростити механізм вибору записів активізації джерел знань. Мова керування базою даних „АРЗ" основана на мові „interLISP" і дозволяє програмувати виконання ряду функцій оболонки [8].

- Структурні компоненти дошки оголошень – це об'єкти мови АРЗ. Такий об'єктно-орієнтований підхід спрощує подання й операції з даними і частковими рішеннями.
- Активізація джерел знань реалізується за допомогою керованих зразками демонів АРЗ.
- АРЗ підтримує й операції з базою даних контекстів, а умови перетворення контексту зображені у вигляді обмежень мови. Контексти можна використовувати для організації набору альтернативних способів продовження обчислень.

Джерело знань в експертній системі, побудованій на базі оболонки HEARSAY-III, повинно складатися із стартового зразка (trigger), первинної програми (immediate code) і тіла (body). Знайшовши відповідність між поточним вмістом дошки оголошень і стартовим зразком, оболонка створює вузол запису активізації для цього джерела знань і запускає на виконання первинну програму. Через деякий час запис активізації джерела знань вибирається планувальником і тоді запускається на виконання тіло джерела знань, що являє собою програму мовою LISP. До складу HEARSAY-III входить найпростіший планувальник, що виконує базові функції планування в експертній системі: вибір чергового запису в списку

актуальних і запуск на виконання програмного коду відповідного джерела знань.

Стартовий зразок має вигляд шаблона мовою АР3 і являє собою предикат, примітивами якого є шаблони фактів і довільні предикати мови LISP. Всякий раз, коли база даних модифікується і виявляється, що поточні дані в ній вже порівняні з усіма шаблонами в зразку, створюється вузол запису активізації, що зберігає назву джерела знань, пусковий контекст і значення змінних, отриманих в результаті зіставлення. При створенні запису активізації виконується первинна програма джерела знань. Ця програма, написана мовою LISP, може пов'язувати з вузлом запису активізації деяку інформацію, що пізніше може бути використана при виконанні тіла джерела. Первинна програма виконується в пусковому контексті і у ній можуть використовуватися конкретизовані в цьому контексті змінні зразка. Значення, що повертається первинною програмою після завершення, – це ім'я якого-небудь із класів вузлів дошки оголошень. Потім запис активізації розміщується на дошку оголошень як вузол цього класу.

Через деякий час базовий планувальник системи, що входить до складу оболонки HEARSAY-III, ініціює виконання будь-якої операції з записом активізації. Як правило, це виконання тіла джерела знань у пусковому контексті із зазначеними змінними. Кожен сеанс виконання тіла джерела знань неподільний – це аналог транзакції в системах керування базами даних. Сеанс продовжується до повного завершення і не може бути перерваний для активізації будь-якого іншого джерела знань.

При проектуванні прикладної експертної системи на базі оболонки HEARSAY-III потрібно самостійно розробити процедуру базового планувальника, що буде викликатися оболонкою після запуску. Ця процедура може бути досить простою, оскільки велика частина знань про планування може бути включена в планувальні джерела знань. Наприклад, базовий планувальник може являти собою найпростішу циклічну процедуру, що вибирає перший елемент із черги, організованої планувальними джерелами знань, і запускає його виконання. Якщо виявляється, що черга порожня, базовий планувальник завершує роботу системи.

Основні переваги середовища HEARSAY-III – це, по-перше, використаний у ньому режим керування, що надає розробникам прикладних експертних систем велику свободу у виборі способів подання і застосування евристик відбору активізованих записів джерел знань, а по-друге, структуризація множини об'єктів дошки оголошень.

7.7 Контрольні питання

1. Поняття та принципи подання знань.
2. Логічні моделі подання знань: загальні положення.

3. Формальні мови.
4. Числення висловлювань.
5. Числення предикатів.
6. Логічне програмування. Мова PROLOG.
7. Принцип резолюцій.
8. Продукційні моделі подання знань: загальні положення.
9. Проектування продукційних експертних систем.
10. Характеристика структурних елементів ЕС продукційного типу.
11. Семантичні мережі як модель подання знань: загальні положення.
12. Класифікація семантичних мереж.
13. Використання семантичних мереж при розробці експертних систем.
14. Фреймові моделі подання знань: загальні положення.
15. Основи теорії фреймів.
16. Основні властивості фреймів.
17. Модель дошки оголошень: загальні положення.
18. Принципи організації систем з дошкою оголошень.
19. Система HEARSAY: загальна характеристика.
20. Використання джерел знань у HEARSAY-II.
21. Оболонка для створення систем з дошкою оголошень HEARSAY-III.

БІБЛІОГРАФІЧНИЙ ОПИС

1. Дж. Ф. Люгер. Искусственный интеллект: стратегии и методы решения сложных проблем. 4-е издание: Пер. с англ. – М.: ИД «Вильямс», 2003. – 864 с.
2. Поспелов Г.С. Искусственный интеллект – основа новой информационной технологии. – М.: Наука, 1988. – 274 с.
3. Ракитов А.И. Философия компьютерной революции. – М.: Полит-издат, 1991. – 287с.
4. Компьютер обретает разум / Под ред. В.Л. Стефанюка. - М.: Мир, 1990. – 240 с.
5. Широчин В. Слово об интелекте. – Киев: Наукова думка, 1998. – 250 с.
6. Искусственный интеллект. В 3-х кн.: Справочник / Под ред. З.В. Попова. Кн.1; Д.А.Поспелова Кн.2; В.Н.Захарова Кн.3 – М.: Радио и связь, 1990.
7. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бэйсике / Р.Левин, Д.Дранг, Б.Эдельсон: Пер.с англ. – М.: Финансы и статистика, 1990 – 239 с.
8. П. Джексон. Экспертные системы. - М.: ИД «Вильямс», 2001. – 609 с.
9. Ситник В.Ф. та ін. Основи інформаційних систем: Навчальний посібник. – К.: КНЕУ, 2001. – 420 с.
10. Алексєєв А.А., Костіна Н.І., Кононець О.Я. Фінансово-економічні експертні системи. Навчальний посібник // За ред. Н.І. Костіної. – К.: Видавничий дім „Скарби”, 2004. - 208 с.
11. Уотермен Д. Руководство по экспертным системам. – М.: Мир, 1989.
12. Петров В.М., Яблонский А.И. Математика и социальные процессы. – М.: Знание, 1980.
13. Представление и использование знаний. / Под. ред. Х. Уэно, М. Исидзука: Пер. с японск. – М.: Мир, 1989. – 220 с.
14. Базы знаний интеллектуальных систем. / Т.А. Гаврилова, В.Ф. Хорошевский – СпБ: Питер, 2000. – 384 с.
15. Базы и банки знаний. / Под ред. В.Н. Четвирикова. – Москва: Высшая школа, 1992. – 348 с.
16. Нейлор К. Как построить свою экспертную систему: Пер. с англ. – М.: Энергоатомиздат, 1991. – 286 с.
17. Дж. Элти, М. Кумбс. Экспертные системы: концепции и примеры / Пер. с англ. и предисл. Б.И. Шитикова. – М.: Финансы и статистика, 1987. – 191 с.
18. Разработка и применение экспертно-обучающих систем: Сб. научн. трудов. – М.: НИИВШ, 1989. – 154 с.
19. Таунсенд К., Фохт Д. Проектирование и програмная реализация экспертных систем на персональных ЭВМ: Пер. с англ. – М.: Финансы и статистика, 1990.

20. Кнут Д. Искусство программирования для ЭВМ: В 3 т. – М.: Мир. – 1978.
21. Джонс М.Т. Программирование искусственного интеллекта в приложениях. Пер. с англ. Осипов А.И. – М.: ДМК Пресс , 2004. – 312 с.
22. Buchanan et al. Constructing an expert systems. – MA, Addison-Wesley, 1983.
23. Гладун В. П. Процессы формирования новых знаний. – К.: Наукова думка, 1988. – 269 с.
24. Wielinga et al. Models of expertise. – Proc. of 7th European Conference on Artificial Intelligence, 1992. – p. 306-318.
25. ClanceyW.J. Heuristic Classification. - Artificial Intelligence, 1985. – p. 289-350.
26. Alexander el al. Knowledge level engineering: ontological analysis. - Proc. of National Conference on Artificial Intelligence, 1986. – p. 963-968.
27. Бублик Б.Н. Модели и системы обработки информации.– К.: Лыбидь, 1991 – 105 с.
28. Winston P.H. Artificial Intelligence. - MA, Addison-Wesley, 1984.
29. Цикризис Д., Лоховски Ф. Модели данных. – М.: Финансы и статистика, 1985. – 274 с.
30. Kowalski R.A. Logic for problem solving. – Amsterdam, North Holland, 1979, p. 88-89.
31. Robinson J.A. Logic: Form and Function. – Edinburgh, Edinburgh Press., 1979.
32. Nilsson N.J. Principles of Artificial Intelligence. – Palo Alto, CA, Tioga, 1980.
33. Бойко В.В., Савинков В.П. Проектирование базы данных информационных систем. – Москва: Финансы и статистика, 1989. – 347 с.

Навчальне видання

Володимир Іванович Месюра
Андрій Анатолійович Яровий
Ігор Ростиславович Арсенюк

ЕКСПЕРТНІ СИСТЕМИ

Частина 1

Навчальний посібник

Оригінал-макет підготовлено Яровим А.А.

Редактор В. О. Дружиніна
Коректор З. В. Поліщук

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 15.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7х42 1/4
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 15.12.2001