Є. Н. Федорчук

Видавництво Л

# ПРОГРАМУВАННЯ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

## ЕКСПЕРТНІ СИСТЕМИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

**Є. Н. Федорчук**

# ПРОГРАМУВАННЯ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

## ЕКСПЕРТНІ СИСТЕМИ

Навчальний посібник

*Рекомендувала Науково-методична рада*
*Національного університету "Львівська політехніка"*



Львів
Видавництво Львівської політехніки
2012

УДК 004
ББК 32.97 3
~~Ф 81~~
ФО 32

**Рецензенти:**

**Камінський Р.М.,** доктор технічних наук, професор кафедри інформаційних систем і мереж Національного університету "Львівська політехніка";

**Сопрунюк П.М.,** доктор технічних наук, професор, провідний науковий співробітник Фізико-механічного інституту імені Г.В. Карпенка НАН України

*Рекомендувала Науково-методична рада*
*Національного університету "Львівська політехніка",*
*як навчальний посібник для студентів напряму "Програмна інженерія"*
*(Протокол № 9/2011 від 14.06.2011 р.)*

**Федорчук Є.Н.**

Ф 81 Програмування систем штучного інтелекту. Експертні системи: навч. посібник / Є.Н. Федорчук. – Львів: Видавництво Львівської політехніки, 2012. – 168 с.

ISBN 978-617-607-257-7

Для студентів вищих навчальних закладів, які навчаються за спеціальністю "Комп'ютерні науки" та "Програмна інженерія". Розглядаються основні моделі знань з чіткими та нечіткими даними, алгоритми пошуку розв'язку експертних задач, технології і засоби програмування баз знань та експертних систем. Подано приклади програмування експертних систем мовою Visual Basic у середовищі Excel.

**УДК 004**
**ББК 32.97**



ISBN 978-617-607-257-7



# ВСТУП

Дисципліна "Системи штучного інтелекту" є однією із базових у підготовці фахівців за спеціалізацією "Програмна інженерія". Вивчення технологій експертних систем займає центральне місце в процесі викладання дисципліни.

На основі тривалого досвіду розроблення експертних систем у різних предметних областях з'явилися експертні технології, призначені для проектування широкого класу систем штучного інтелекту. Основою цих технологій є відпрацьована методологія використання різних типів моделей знань та алгоритмів їх моделювання.

Під час вивчення експертних технологій важливо ознайомитися з використанням для задач моделювання знань різних парадигм програмування. Це насамперед спеціалізовані мови подання знань і системи інженерії знань, об'єктно-орієнтоване та предметно-орієнтоване програмування, логічне, функційне та еволюційне програмування й інші парадигми.

Вважаємо, що програмування експертних задач є привабливим і цікавим для людей з логічним типом мислення. Воно розвиває інтерес до програмування нетрадиційних задач зі складними типами даних. Для цих задач є логічно вмотивованим поєднання класичних та евристичних алгоритмів пошуку розв'язків.

Матеріал посібника містить всі основні теми дослідження і проектування експертних систем. Глибина викладення матеріалу відповідає навчальним програмам освітнього рівня для бакалаврів напряму "Програмна інженерія".

Посібник призначений для студентів вищих навчальних закладів, які навчаються за спеціальностями "Програмна інженерія" та "Комп'ютерні науки". Метою посібника є ознайомлення студентів з основами теорії експертних систем. Розглянуто основні типи моделей знань для експертних задач, зокрема задач з нечіткими даними. Наведено алгоритми розв'язання експертних

задач на основі продукційних правил. Подано приклади проектування і програмування експертних систем. Приклади алгоритмів і програмних засобів подано засобами мови Visual Basic у середовищі Excel. Такий підхід задовольняє концепцію швидкого розроблення прототипу ЕС недорогими засобами. Прототип може надалі розвиватись на основі більш досконалих і ресурсомістких технологій.

Посібник складається з семи частин, які поділяються на розділи, а розділи – на параграфи. Кожний розділ закінчується набором контрольних запитань і тем для обговорення, до яких іноді додаються задачі і вправи. Наведено описи і схеми алгоритмів, візуальні форми, які сприяють кращому розумінню процесу проектування експертних систем.



# Розділ 1
# СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

## 1.1. Огляд досліджень у галузі штучного інтелекту

Штучний інтелект (ШІ) як науково-технічний напрям інформатики почав інтенсивно розвиватись у 50-ті роки XX ст. Поштовхом до розвитку послужили теоретичні дослідження в галузі інформатики та прикладні дослідження в галузі комп'ютерної обчислювальної техніки.

Початок досліджень у галузі ШІ пов'язують із роботами Ньюела, Саймана і Шоу, які вивчали і моделювали процеси розв'язування людиною різноманітних задач [22]. Ці роботи започаткували **перший етап** досліджень у цій галузі, пов'язаний із моделюванням процесів мислення та розробленням програм, за допомогою яких розв'язують задачі різноманітними евристичними методами. Предметом досліджень для розвитку методів ШІ на цьому етапі були головоломки, ігрові та математичні задачі. Вагомим практичним результатом першого етапу стали перші програмні системи ШІ – експертні системи.

**Другим етапом** досліджень у галузі ШІ стало моделювання поведінки живих істот. Першим, хто висунув ідею про машини та системи, які самі себе налаштовують, регулюють та контролюють свою роботу, був Норберт Вінер – батько науки, яку він сам назвав кібернетикою в книжці з однойменною назвою "Кібернетика" [10]. Ідеї Вінера зацікавили цією проблемою вчених та громадськість. Відтоді в розроблення обчислювальної техніки почали вкладати величезні кошти, що спричинило її бурхливий розвиток.

Кібернетичний напрям досліджень у галузі ШІ привів до створення програмно керованих механізмів та робототехнічних систем. Такі системи акумулювали перспективні результати першого етапу щодо задач планування, управління, розпізнавання

в реальному часі. Їх розвиток спричинив створення нових парадигм моделювання знань на основі нейронних мереж, дерев розв'язків, агентів.

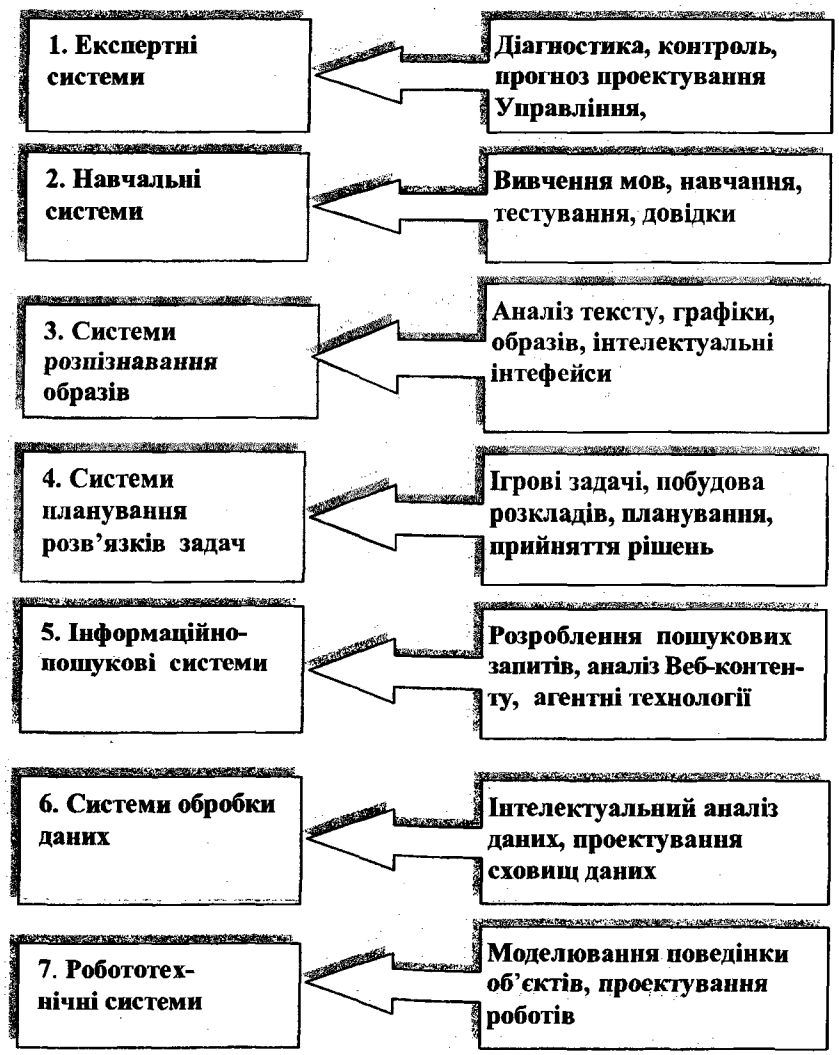



*Рис. 1.1. Класифікація задач і систем штучного інтелекту*



Характерною рисою **третього етапу** дослідження систем штучного інтелекту було переміщення центру уваги дослідників зі створення автономно функціонуючих систем, до створення людино-машинних систем. Ці системи інтегрують у єдине ціле інтелект людини і можливості комп'ютерної техніки для досягнення загальної мети – розв'язання задачі, поставленої перед інтегральною людино-машинною обчислювальною системою.

В.М. Глушков вважав, що перспектива розвитку засобів обчислювальної техніки істотно залежить від розмаїтості інструментарію підтримки вирішення проблеми людино-машинної взаємодії [3].

Сьогодні триває поєднання основних результатів всіх трьох етапів досліджень ШІ та розвиток нових напрямів. Цей процес привів до створення цілого класу систем штучного інтелекту (СШІ). Узагальнену схему класифікації СШІ та вирішуваних за їх допомогою задач подано на рис. 1.1. На схемі подано 7 розвинутих класів СШІ, які якнайширше представлені в класі як дослідницьких, так і практичних, комерційних розробок.

## 1.2. Розвиток і галузі використання експертних систем

На початку 80-х років XX ст. у дослідженнях ШІ виник самостійний напрям – так звані "експертні системи" (ЕС). Метою досліджень ЕС було розроблення програм, які, розв'язуючи задачі, важкі для експерта-людини, дають результати, що не поступаються за якістю та ефективністю рішенням експерта.

Експертні системи автоматизують процес розв'язання важко формалізованих задач у реальних предметних галузях. Для цього необхідний значний досвід і знання фахівців. Багато програм з галузі штучного інтелекту є суто дослідницькими. Основна увага в них приділяється абстрактним математичним проблемам або спрощеним варіантам реальних проблем. Метою виконання таких програм є відпрацьовування методики розв'язання певної задачі.

Експертні системи мають яскраво виражену практичну спрямованість у науковій або комерційній галузі.

На основі досвіду використання ЕС у різних предметних областях розвинулись експертні технології, призначені для проектування широкого класу систем ШІ. Основою цих технологій є відпрацьована методологія використання різних типів моделей знань та алгоритмів їх моделювання. До основних рис методології належать:

- розроблення описів і онтологій для експертних задач і предметних областей ;
- широке використання апарату нечіткої логіки і нечіткого моделювання знань;
- застосування методів інтелектуального аналізу та нейромережевих технологій;
- розроблення нових методів і технологій проектування систем ШІ;
- розроблення методів побудови комбінованих моделей знань для задач, розв'язуваних у реальному часі;
- інтегрування методів ШІ в програмну інженерію.

Експертні технології надають для ефективного розв'язання задач ШІ та інших прикладних задач такі можливості:

- засоби набуття знань і створення баз знань;
- засоби для проектування експертних і споріднених систем;
- засоби для моделювання знань з нечіткими даними;
- засоби для інтегрування експертних технологій у середовища таких програмних систем, як: КСП, ГІС, СППР.

Сьогодні експертні технології використовуються для автоматизації розв'язання різних типів задач (діагностика, інтерпретація, прогнозування, планування, проектування, контроль, налагодження, керування) у різноманітних проблемних галузях:

- фінанси;
- нафтова і газова промисловість;



- енергетика і транспорт;
- фармацевтичне виробництво;
- космос;
- металургія;
- гірська справа;
- хімія;
- освіта;
- агробізнес;
- телекомунікації і зв'язок та ін.

Експертні системи репрезентують ефективну конкурентну технологію на ринку індустрії програмного забезпечення. Ринок програмного забезпечення на основі технологій ШІ розвивається дуже динамічно і оцінюється у мільйони доларів. Компанія DuPont експлуатує понад 600 експертних систем і отримує щорічні заощадження до податку, що оцінюються в сотнях мільйонів. Близько 20 цих систем працюють у режимі реального часу. Решта використовуються в консультативному режимі або в режимі підтримки рішення.

Credit Authorizer's Assistant – експертна система кредитного дозволу – ідентифікує ризики понад 23 млн. власників кредитних карток. Вона відстежує крок за кроком процес міркування одного із своїх досвідчених аудиторів кредиту, скорочуючи час прийняття рішення на 25 %. American Express отримала на основі цієї системи до 60 % скорочень витрат від фальсифікованих операцій. Очікувані переваги від зменшення ризику, вдосконалення дохідних статей оцінюються в 27 млн. дол. на рік.

British Petroleum (Британська нафта) розробила і розгорнула багато експертних систем. Можливо, найкращою серед відомих систем є радник для проектування розділювачів газу/нафти. Використання її забезпечило економію витрат у кілька мільйонів фунтів стерлінгів на рік.

Багато японських компаній здійснили реальні інвестиції в експертні системи із значним обсягом і виграшем. Експертні

системи розробляють сотні японських фірм. ЕС застосовуються в інжинірингу інтенсивних індустрій, важких машин і індустріях матеріалів, в конструюванні, хімії, страхуванні і фінансових послугах.

Інтенсивний розвиток інфокомунікаційних технологій та збільшення інвестицій розширює сферу застосування експертних технологій і систем. Така тенденція підтримує на високому рівні інтерес до проектування і програмування ЕС у широких колах фахівців та програмістів.

## Контрольні запитання

1. Проаналізуйте завдання першого етапу досліджень для ШІ.
2. Проаналізуйте завдання другого етапу досліджень для ШІ.
3. Проаналізуйте області застосування експертних систем.
4. Поясніть задачі систем розпізнавання образів.
5. Поясніть задачі експертних систем.
6. Поясніть задачі робототехнічних систем.
7. Поясніть задачі систем обробки даних.
8. Проаналізуйте можливості експертних технологій.
9. Поясніть задачі робототехнічних систем.
10. Проаналізуйте основні риси методології використання моделей знань.



# Розділ 2
# МОДЕЛІ ЗНАНЬ ЕКСПЕРТНИХ ЗАДАЧ

## 2.1. Формалізація моделі знань

Важливим питанням під час розроблення ЕС є моделі подання знань про предметну область. Для цього використовують різні форми моделей знань.

***Представлення знань*** – це методологія моделювання і формалізації концептуальних знань, орієнтована на комп'ютерну обробку. Ця методологія і відповідні комп'ютерні технології є предметом інженерії знань. Можливості представлення знань пов'язані із організацією та використанням різноманітних структур даних: змінних, масивів, структур, сімейств, таблиць, дерев.

Комп'ютерні технології потребують тривалої підготовки, пов'язаної зі збиранням, представленням знань, класифікації поточної задачі і вибором технології її вирішення. Тут основні зусилля спрямовуються на уніфікацію структур даних, технології їх представлення і опрацювання, на ефективність (точність та швидкість) процедури пошуку розв'язків.

Модель **знань** формально складається з двох частин:

$$\mathbf{M_Z = <O,P>}, \qquad (2.1)$$

де $\mathbf{O=<A, R_A>}$ – декларативний структурований опис реальної ПО з її об'єктами **A** і відношеннями між ними $\mathbf{R_A}$. Опис подано у формі гіпертексту, списків, деревоподібних структур; $\mathbf{P=<B, R_B, S>}$ – процедурні знання про методи розв'язування задач у ПО. До **P** належать: **B** – об'єкти певних класів; $\mathbf{R_B}$ – відношення між об'єктами; **S** – стратегії і алгоритми операцій з відношеннями для отримання рішення.

Підстановка **O** та **P** в (2.1) дає розширений опис моделі ПО:

$$\mathbf{Mz = <A, R_A, B, R_B, S>}.$$

У різних ПО співвідношення **O** і **P** є різним. Для їх опису можна використовувати такі моделі знань: для **O** – файли (текстові,

графічні, мультимедійні, комбіновані), семантичні, фреймові або гіпертекстові моделі; для **P** – моделі продукційних правил, алгоритмічні, обчислювальні процедури. Узагальнений запит до моделі знань **Z** можна подати у вигляді множини інформаційних об'єктів

$$Z=<X,Mz,Y>, \qquad (2.2)$$

де **X** – сукупність вхідних даних, яка підлягає інтерпретації в моделі **Mz**; **Y** – сукупність проміжних результатів, які формує система, що оперує з моделлю **Mz.**

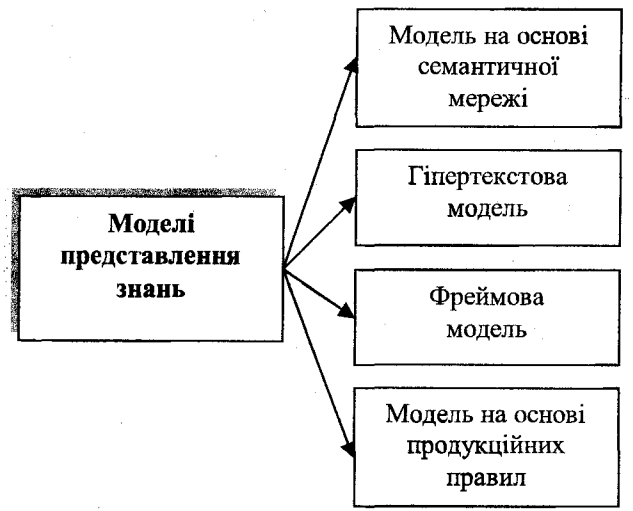



*Рис. 2.1. Моделі знань в ЕС*

За результатами аналізу інформаційної структури моделі виду (2.1) можна виділити такі типові діалогові операції інтерфейсу:

- аналіз запиту, інтерпретація вхідних даних **X**;
- вибір точок входу в модель **Mz** і активація операцій пошуку;
- виведення та аналіз вихідних даних **Y**;
- обробка: діагностика помилок, підтримка діалогу.



Означені операції фактично є функціями для розроблення інтерфейсу ЕС.

Розглянемо характеристики найпоширеніших моделей, поданих на рис. 2.1.

## 2.2. Моделі знань у формі семантичних мереж

Семантична мережа (СМ) як об'єднання базових елементів дає змогу відображати складні сукупності об'єктів ПО і відношень між ними. Формальний опис мережевої моделі як інформаційної структури можна подати у вигляді

$$M_C = <I,C_1,C_2,\ldots,C_n,\Gamma>.$$

Тут I – множина інформаційних одиниць; $C_1,\ldots,C_n$ – множина типів зв'язків між ними; Г – множина зв'язків із заданого набору типів [1]. Залежно від типів зв'язків розрізняють: класифікуючі мережі, функціональні мережі, сценарії. Абстрактно мережу можна зобразити у вигляді орієнтованого графу. Вершини графу – це поняття, а ребра (дуги) – асоціативні ієрархічні зв'язки між вершинами.

Визначення семантичної мережі ґрунтується на гіпотезі, що пам'ять людини формується через асоціації між поняттями. Формалізм асоціацій було розвинуто для подання класів об'єктів у різних предметних областях. Це області з просторовими та структурними зв'язками (фізичні і технічні системи), області з причинно-наслідковими зв'язками (медицина, біологія, екологія), функціональними зв'язками (техніка).

**Графічне зображення**. Базовий функціональний елемент семантичної мережі – це структура, яка містить два елементи: вузол і дугу. Вузли – це деякі поняття(об'єкти), дуги – відношення між парами понять. Базовий елемент семантичної мережі подано на рис. 2.2.

Об'єктами можуть бути: узагальнені поняття, події, дії, абстрактні об'єкти. Розглянемо приклад семантичної мережі, поданої на рис. 2.3, яка дає змогу розв'язувати задачу класифікації поштових об'єктів.

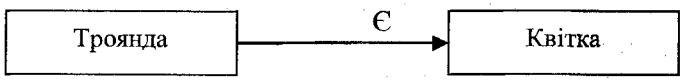



*Рис. 2.2. Базовий елемент семантичної мережі*

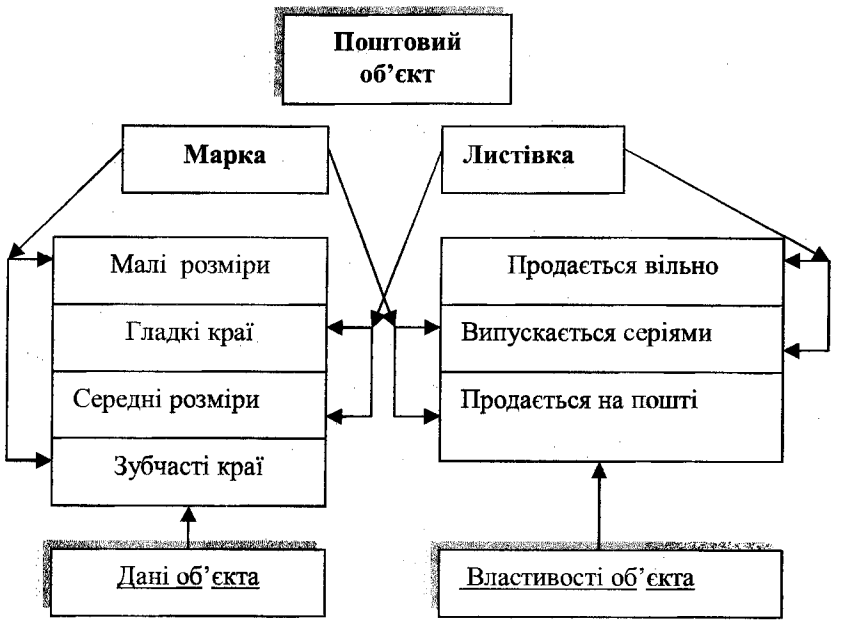



*Рис. 2.3. Приклад семантичної мережі для класифікації поштових об'єктів*

Ця мережа на основі аналізу відношень (дані, властивості) дає змогу класифікувати два поштові об'єкти: марку та листівку. Наприклад, марка класифікується за такими критеріями: ***має дані*** – малі розміри і зубчасті краї; ***має властивості*** – випускається серіями і продається тільки на пошті.

**Цікавим прикладом СМ** є звичайний кросворд із його різноманітними графічними зображеннями. Це СМ із чітко вираженою семантикою: об'єктами-словами і відношеннями у формі просторової структуризації – розміщенням слів на перетині спільних букв. Розпізнавання у такій мережі правильних слів потребує



"експертних" семантичних знань. Розроблення таких семантичних мереж постійно перебуває в полі зору дослідників ШІ.

Для задач класифікації в семантичних мережах виділяють базові типи відношень:

- бути – належати до класу;
- мати властивості – наслідувати властивості елементів класу;
- мати дані (значення) – отримувати значення властивостей.
- Інші можливі типи відношень:
- відношення структуризації;
- функціональні відношення;
- каузальні (причинно-наслідкові )відношення;
- семантичні відношення.

На множині інформаційних одиниць у деяких випадках корисно задавати відношення, що характеризують ситуативну (асоціативну) близькість. Такі зв'язки називають релевантними відношеннями (близькими за змістом). У семантичних мережах найчастіше застосовують два типи логічного виведення (міркування): наслідування і розпізнавання.

*Наслідування* – це міркування, на основі якого властивості класу приписуються індивідуальному об'єкту з класу. Наприклад: птахи літають, орел – літає, отже, орел – птах.

*Розпізнавання* – це міркування, як за набором властивостей індивідуального об'єкта визначають клас, до якого він належить. Наприклад: птах літає високо, має широкі крила, отже, птах – орел.

Крім ЕС, семантичні мережі широко використовуються в різних системах ШІ для розв'язання таких задач:

- аналіз семантичних структур мов програмування;
- лінгвістичний аналіз природних мов;
- організація пошуку в базах даних на основі запитів;
- семантична організація Web-сторінок.

Семантична мережа описує ПО в термінах об'єктів-понять (іменовані вершини мережі) і бінарних асоціацій-зв'язків між об'єктами (позначені орієнтовані дуги). Серед об'єктів мережі можна виділити власне понятійні об'єкти, об'єкти-події, об'єкти-характеристики. Отже, у семантичній мережі можна подати вираз різної складності, наприклад, висловлення, що відображають причинно-наслідкові зв'язки, виражаються введенням відношень типу: умова–висновок.

Однією із переваг семантичних мереж є ефективний інформаційний пошук, оскільки асоціації між об'єктами визначають шляхи доступу до інформаційних елементів БЗ. Іншою перевагою семантичних мереж є можливість відображення структури, властивої знанням, тому що відношення частина-ціле, елемент-множина, клас-підклас можуть бути явно зазначені.

Головним недоліком мережних моделей є відсутність достатньо розробленої операційної семантики. Остання, як правило, використовує прості міркування, пов'язані з реалізацією алгоритмів пошуку за шляхами доступу. Численні формалізми, що підтримують мережеві схеми, не мають необхідного рівня повноти синтаксису і семантики. Відомо багато варіантів реалізації семантичних мереж, що дає підставу говорити про цілий клас семантичних представлень.

## 2.3. Засоби побудови семантичних мереж

**Створювати СМ** на концептуальному рівні можна двома шляхами: знизу догори та згори донизу. Перший реалізується, коли потрібно систематизувати вже наявну множину фактів у ПО. Створюють СМ у цьому випадку шляхом визначення нових понять і відношень, які можна пов'язати із фактами.

Другий шлях вибирають у разі створення вузькоспеціалізованих інформаційних систем. Тоді спочатку структурують ПО



за допомогою базових понять, а потім відповідно до заданої структури групують дані–факти.

Для ефективного візуального подання семантичних мереж та організації пошуку з довільним доступом використовують спискові, графічні та деревоподібні структури – об'єкти. Їх проектують з використанням відповідних елементів управління в середовищі об'єктно-орієнтованих мов програмування. На основі цих структур проектують також інший тип пошукових систем – навігатори та в'ювери для пошуку у файлових підсистемах. Ієрархічні деревоподібні структури є поширеною моделлю для інтегрованих довідкових систем у таких програмних засобах, як Visual Basic, Delfi, Ci Bilder. Наприклад, в середовищі Visual Basic є елементи Treeview, Listview для створення деревоподібних та лінійних структур даних. Для швидкого розкриття дерева використовують рекурсивні процедури сканування вузлів дерева. Приклад двох таких програм-процедур мовою Visual Basic подано нижче.

**Програма 1. Операції з елементами дерева.**

Виведення умісту усіх вузлів дерева. Це операція із сімейством

```
Dim node as node
Set tree1= treeview1
For each node in tree1.nodes
Textbox1.text = node.text
next
```

**Програма 2** – програма сканування кореневого вузла дерева і запис їх назв у список

```
ScanNode Treeview1.Nodes(1) – виклик програми
Sub ScanNode(aNode As Node)
Dim ThisNode As Node
Dim as long
List1.Additem aNode.text – список всіх вузлів
```

```
If aNode.Childern >0 Then
Set thisNode = aNode.Child
For i=1 to aNode.Childern
thisNode
Set thisNode = thisNode.Next
Next i
End if
End sub
```

Засоби представлення і перетворення знань, заданих у вигляді СМ, містять:

- мовні засоби визначення елементів і перетворення структури СМ;
- засоби непроцедурного маніпулювання даними;
- спеціальні процедури графічної візуалізації СМ та її відображення для відповідних фрагментів БД і БЗ.

**Особливості проектування семантичних мереж.** У разі великих обсягів даних у мережі складно якісно показати всі деталі. Необхідно розбивати мережу на частини. Це заважає цілісному візуальному сприйняттю, сповільнює опрацювання інформації. Доволі складно модифікувати мережу, особливо видаляти вузли, які мають внутрішні посилання.

Для побудови моделей знань на основі семантичної мережі розроблено мови, технології і засоби програмування. Розглянемо характеристики деяких з них [6]. Технологія **Semp-T** поєднує комплекс засобів і методів подання й обробки знань, що містить:

- високорівневі засоби задання семантики об'єктів предметної області шляхом специфікації обмежень на значення їхніх параметрів і локальних правил висновку;
- ієрархічну семантичну мережу з обумовленими властивостями відносин.

На рис. 2.4 і 2.5 наведено фрагмент оболонки **Semp-T** й один з її елементів: вікно редагування властивостей класу.



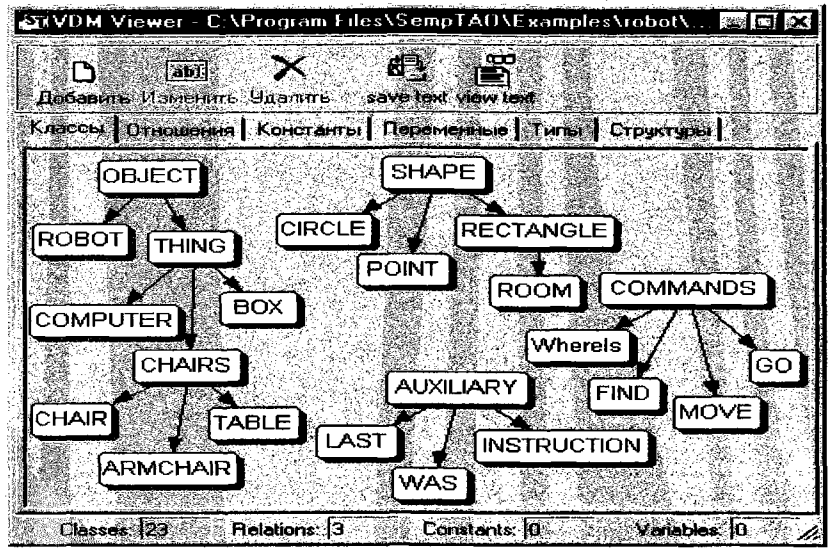



*Рис. 2.4. Фрагмент семантичної мережі в середовищі **Semp-T***

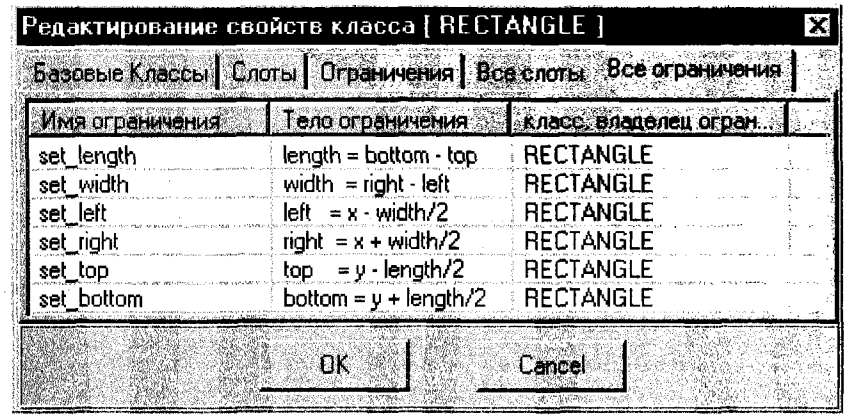



*Рис. 2.5. Вікно редагування елементів класу*

**Вправи**

Задано описи об'єктів. Побудувати семантичну мережу для розпізнавання або класифікації об'єктів з використанням типових

відношень. Записати у загальному вигляді продукційні правила розпізнавання. Описи об'єктів:

1. ***Тип овоча*** (зелений колір, плоди лежать на землі, плоди круглі, великий розмір, насіння (подовгасте, їстівне), середній розмір, плететься по землі, плоди подовгасті);
2. ***Тип нагороди*** (виготовляється з кольорових металів, має дві частини, продається, є суцільним, вручається, виготовляється з цінних металів, має спеціальний дублікат);
3. ***Тип ягоди*** (росте на кущі, плоди (червоні, круглі, гладкі), росте на стеблі, плоди (зернисті, червоні), кисла на смак, росте з кореня, солодка на смак);
4. ***Літальний апарат*** (два двигуни, середня швидкість, малі крила, потребує великої смуги, шасі нерухоме, довгі крила, мала швидкість, злітає з малої площі, шасі підйомне);
5. ***Спорт*** (круглий м'яч, гра руками, висока сітка, середнє поле, овальний м'яч, велике поле, гра руками, колективна гра, є ворота);
6. ***Спорт*** (великий м'яч, висока сітка, середнє поле, малий м'яч, колективна гра, низька сітка, парний);
7. ***Комп'ютер*** (окремий монітор, автономне живлення, середні габарити, живлення від мережі, суміщений монітор, малі габарити і вага);
8. ***Тип квітки*** (червоний колір, стебло гладке, квітка багатопелюсткова, розмножується з цибулини, стебло колюче, дерев'янисте, квітка чашоподібна, розмножується з кореня).

## 2.4. Фреймові моделі знань

Фрейм як структуру для представлення знань запропонував М. Мінський у 70-ті роки XX ст. для опису сценаріїв реального світу [11]. В інженерії знань фреймом називається інформаційна



структура для опису типових об'єктів, подій, процесів. Формально фрейм подають у вигляді табл. 2.1

*Таблиця 2.1*

**Структура фрейму**

| **Ім'я фрейму** | | |
|---|---|---|
| Ім'я слоту 1 | Значення слоту 1 | Приєднана процедура |
| Ім'я слоту 2 | Значення слоту 2 | Приєднана процедура |

Кожен рядок таблиці визначає множину характеристик об'єкта-фрейму. Кожен стовпець, слот описує значення характеристик. Слот – приєднана процедура, яка означає звертання до іншої процедури обчислення або пошуку значення слоту. Роль процедури може виконувати звертання до іншого фрейму. Для різних предметних областей структура таблиці може змінюватися за рахунок збільшення (зменшення) кількості слотів. Тому концептуально фрейм можна вважати мінібазою знань, в якій виконуються операції пошуку і обчислення даних.

У моделях знань на основі фреймів реалізують два типи логічного виведення (міркування): наслідування і розпізнавання. Завдяки цьому фрейми об'єднують в ієрархічні деревоподібні структури. У таких структурах застосовують різні види інформаційних зв'язків через вершини дерева або через слоти фреймів.

Фрейми придатні для моделювання знань у предметних областях, де є фіксовані зв'язки: родові, просторові, причинно-наслідкові, функціональні. Зв'язки є фіксовані і зміна їх є не частою. Внаслідок складної структури зв'язків важко вносити зміни у фреймову модель. Існують ПО, для яких розв'язується низка задач інформаційного обслуговування, наприклад:

- довідкові задачі (про погоду, товари, літературу);
- задачі резервування (місць, білетів, товарів).

Такі задачі оперують обмеженою кількістю понять, які є фактично параметрами задачі обслуговування. Ці параметри

відповідають слотам фрейму, а весь вид обслуговування описується одним фреймом або мережею фреймів. Робота з інформаційною фреймовою системою полягає у заповненні слотів і певних операцій з фреймами, в яких заповнені всі слоти.

Особливості застосування фреймових моделей:

- відсутність потреби у компіляції моделі знань. Пошук у системі фреймів йде в режимі інтерпретації слотів за заданими правилами;
- швидкість візуального аналізу фрейму для прийняття рішень;
- зручність ручної обробки за допомогою маніпулятора-миші;
- легкість приєднання виклику процедур або інших фреймів;
- адаптованість до різного типу структур: фрейми –структури; фрейми – сценарії; фрейми – події або ситуації;
- жорстка прив'язка до алгоритму виведення.

## 2.5. Програмування фреймових моделей знань

У сучасних базах знань фрейми програмують за допомогою об'єктно-орієнтованого програмування (ООП) у вигляді готових об'єктів – статичних або динамічних форм з відповідною до задачі структурою фрейму. Вона об'єднує знання про ситуацію, подію, стан і визначає за допомогою змінних та процедур, які об'єкти входять у цю ситуацію і які можливі події. Така структура фреймів має добру підтримку в межах ООП і відповідних мов, що підтримують технологію ООП.

Наповнення (програмування) фрейму декларативними, функціональними та процедурними елементами дає змогу створювати ефективні структури для управління процесами пошуку, виведення, аналізу в моделях знань. Узагальнену структуру фрейму-форми подано на рис. 2.6.



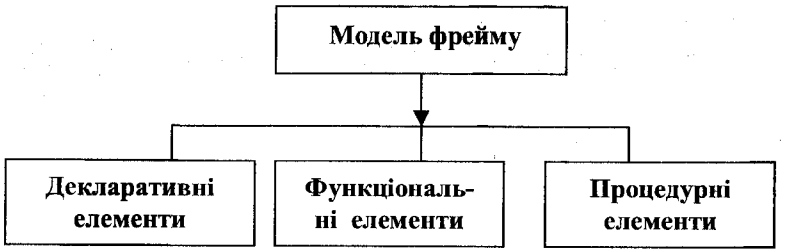



*Рис. 2.6. Структурні елементи фрейму*

***Декларативні елементи***: текст, графіка, таблиці, гіпертекст. Програмуються за допомогою елементів: текстових вікон, графічних вікон, вікна веб-браузера, деревоподібних структур.

***Функціональні елементи:*** введення даних, вибір даних. Програмуються за допомогою елементів: текстових вікон, перемикачів, кнопок, табличних елементів, вікна веб-браузера.

***Процедурні елементи***: меню, виклик процедур обчислень, правил. Програмуються за допомогою елементів управління.

Основні причини поширення фреймів: добра підтримка людських даних-факторів: швидкого візуального аналізу, прийняття рішень і керування за допомогою ручного маніпулятора-миші.

Розглянемо приклади реалізації фреймових моделей. **EXSYS**. Демонстраційна версія містить різні WEB-прикладні демонстрації, що охоплюють 22 проблемні напрями. Для кожного напряму подається від 5 до 10 системних методологій та засобів розв'язання експертних задач. На рис. 2.7–2.9 подано фрагменти фреймової мережі для надання довідкових знань.

Фреймову модель часто застосовують для проектування і розроблення електронних анкет. Таку анкету розробляють у формі мережі фреймів. Окремий фрейм призначається для опрацювання відповідей на одне або декілька питань анкети. На фрейм-формі можна вказати типи і необхідну кількість елементів для опрацювання питань анкети. Створюється мережа фреймів. Окремі фрейми залежно від структури запитання можуть містити одну або декілька локальних ієрархій, які позначають переходи в середині мережі.

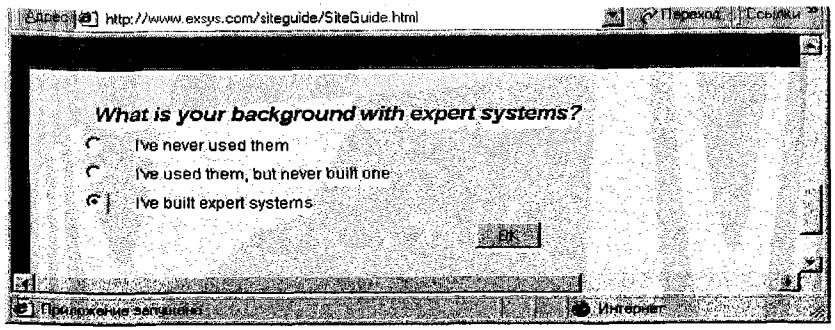



*Рис. 2.7. Фрейм входу в мережу*

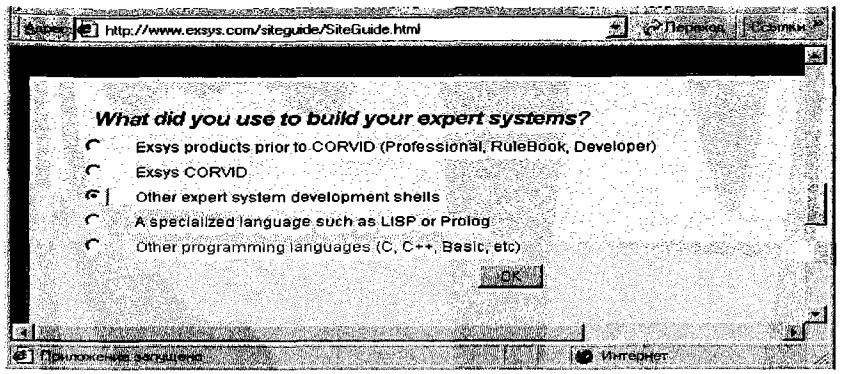



*Рис. 2.8. Проміжний фрейм мережі*

Створену мережу можна надалі оптимізувати з метою об'єднання декількох фреймів в одній формі.

В архітектурі довідкових систем використовують ієрархічну фреймову модель, до якої зручно застосовувати послідовний алгоритм інформаційного пошуку. Логічну модель узагальненої бази знань для надання довідки про технології розв'язання класу задач подано на рис. 2.10.

Модель містить чотири рівні обробки. На кожному рівні задано фіксовану кількість фреймів. У кожному фреймі задано кількість елементів моделі знань. Як правило, вибирають візуально оптимальну кількість: від 4 до 10 позицій.



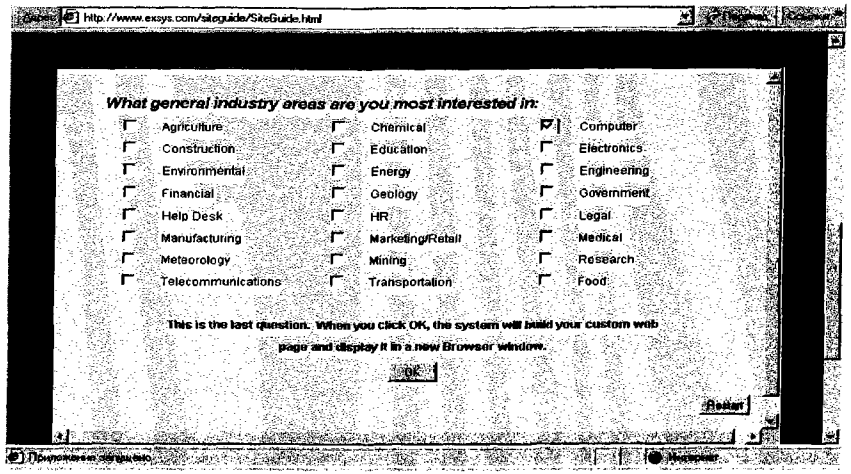



*Рис. 2.9. Кінцевий фрейм мережі для надання довідки в системі EXSYS*

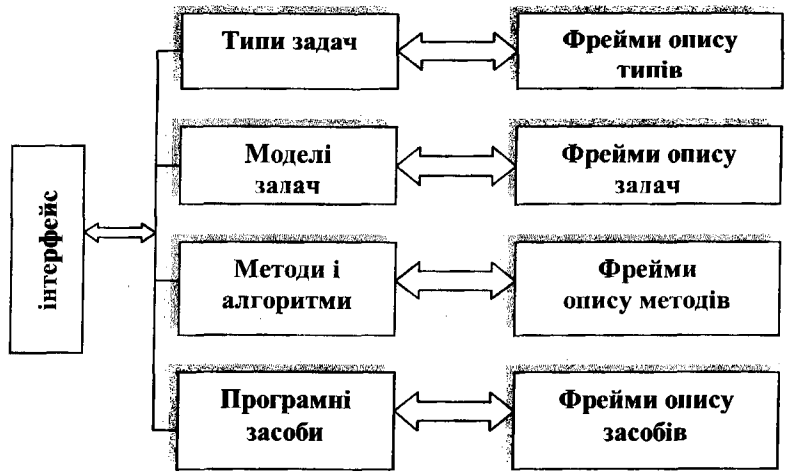



*Рис 2.10. Логічна модель бази знань довідкової системи*

В ЕС застосовують часто фреймово-продукційні моделі, в яких власне фрейми описують декларативні знання, а продукції задають різні алгоритми пошуку розв'язків.

## Контрольні запитання

1. Проаналізуйте зміст фреймової моделі знань.
2. Поясніть особливості фреймових моделей.
3. Проаналізуйте можливості програмування фреймів засобами ООП.
4. Назвіть типові елементи фрейм-форми.
5. Наведіть приклади застосування фреймових моделей знань.

## 2.6. Визначення і функції гіпертекстової моделі знань

Гіпертекстова модель є різновидом семантичної мережі. Автор ідеї – В. Буш. Послідовники Д. Енгельбард і Нельсон розвинули ідею до практичної реалізації [21]. Семантична мережа, вбудована в гіпертекст, відрізняється від звичайної наявністю у вузлах великих об'ємів інформації. Гіпертекст містить три елементи: текстову БД, семантичну мережу для зв'язків фрагментів тексту і засоби зв'язку та перегляду фрагментів тексту (текстова БД + + семантична мережа + інтерфейс). Підготовка тексту містить операції: складання поміток, формування схеми (плану) тексту. Ці операції відображають послідовні етапи процесу опрацювання інформації людиною – дослідження, організація, кодування.

У гіпертексті знання розбиваються на окремі елементи. Елементи подають у різній формі (текст, графіка, мультиплікація, звук) і пов'язують різними відношеннями (послідовностями обробки). Такі зв'язки мають нелінійний характер з погляду їх відображення в лінійній структурі, яка відображає весь текст. Виникають проблеми побудови оптимальної мережевої структури гіпертексту.

Функціонально гіпертекст розв'язує задачі представлення масивів знань, дослідження проблем збирання та керування інформацією. Структурно гіпертекст містить:

- Вузли (текст, дані, малюнки);



- Зв'язки (між вузлами);
- Атрибути (вузлів або міжвузлових зв'язків).

Технологія гіпертексту задає правила маніпулювання елементами за певною схемою керування. Гіпертекстова модель реалізує операції, які дають змогу:

- будувати та знищувати фрагменти гіпертексту;
- встановлювати та знищувати зв'язки між фрагментами;
- редагувати фрагменти, здійснювати імпорт та експорт даних;
- заносити додаткову інформацію до фрагментів – "нотатки на полях";
- відбирати фрагменти відповідно до цієї інформації;
- формувати відповідно до деяких критеріїв користувача перелік фрагментів – контекст у гіпертексті;
- впорядковано переглядати фрагменти гіпертексту відповідно до контексту;
- формувати документ відповідно до заданого контексту;
- отримувати тверді копії документів;
- інтегрувати інформацію інших видів (графіка, звук, відео, анімація).

## 2.7. Гіпертекстові технології

Комп'ютерна технологія, орієнтована на роботу з гіпертекстом, реалізується в спеціальних гіпертекстових системах. У гіпертекстовій мережі легко відображаються з'єднані зв'язками, ідеї розуміння, судження фахівців тощо. Тому такі системи містять засоби, що дають змогу формувати і досліджувати гіпертекст в інтерактивному режимі одночасно колективом розроблювачів. Використовують також відеодискову технологію й інші засоби, що дають змогу представляти як вузли не тільки письмові тексти, але й різну візуальну й акустичну інформацію. Практично всі гіпертекстові системи функціонують на основі багатовіконної технології.

Класифікація гіпертекстових систем містить такі системи:

1. Бібліотечні макросистеми для збереження й актуалізації текстів зі зв'язками і вікнами ( TEXTNET, HiperCard). Використано мову запитів SQL. Об'єкти гіпертексту записуються відношеннями в БД. За допомогою гіпертексту вирішуються проблеми ручної трансформації лінійних документів для великих об'ємів;
2. Системи перегляду БД, що забезпечують легкість доступу до великих масивів інформації і застосовувані в навчальних системах (ZOG, UYPERTIES, SDE);
3. Системи дослідження проблем, використовувані для аналізу гіпертекстової мережі, виділення в ній деяких вузлів і підструктур за визначеними ознаками (IBIS, SINVIEW);
4. Системи для експериментування на різних напрямках застосування гіпертексту, використовувані для дослідження можливостей гіпертекстової технології (NOTECSRDS, INTERMEDIA, TEXTRONIX, NEPTUNE);
5. Спеціалізовані мови та програмні засоби для мережевих технологій з використанням гіпертексту (HTML, XML, Perl).

ЕС 70-х років MICIN і INTERLIST містять гіпертекстові моделі для обробки експертного тексту, що характеризується наявністю динамічних зв'язків.

Технологія гіпертексту стимулює розвиток складних інформаційних структур. Істотні риси гіпертекстової технології мають графічні методи моделювання як статичних так і динамічних структур, підтримувані програмними засобами уведення вузлів і зв'язків, їхньої актуалізації і контролю.

Обробка гіпертексту відкрила нові можливості освоєння інформації, якісно відмінні від традиційних. Замість пошуку інформації за відповідним пошуковим ключем гіпертекстова технологія передбачає переміщення від одних об'єктів інформації до інших з урахуванням семантичної зв'язаності.



Сфера застосування гіпертекстових технологій дуже широка. Це видавнича діяльність, бібліотечна робота, навчальні системи, розроблення документації, баз даних, баз знань. Гіпертекстові моделі знань широко використовують структури у вигляді Веб-сторінок. Хоча багато Веб-сторінок вражають розмаїттям свого оформлення, але практично всі вони передбачають три структурні рівні розміщення інформаційних елементів:

1. Вертикальна ієрархія – елементи гіпертексту розміщені у вигляді рядка, який розміщується згори або знизу форми.
2. Горизонтальна ієрархія – елементи гіпертексту розміщені у вигляді стовпця, який поміщається збоку форми, ліворуч або праворуч.
3. Внутрішня ієрархія – елементи гіпертексту розміщені в середині форми в довільному порядку.

Розміщення елементів ієрархії гіпертексту подано на рис. 2.11. Така організація сторінок дає змогу раціонально керувати пошуком інформації на Веб-сайті.

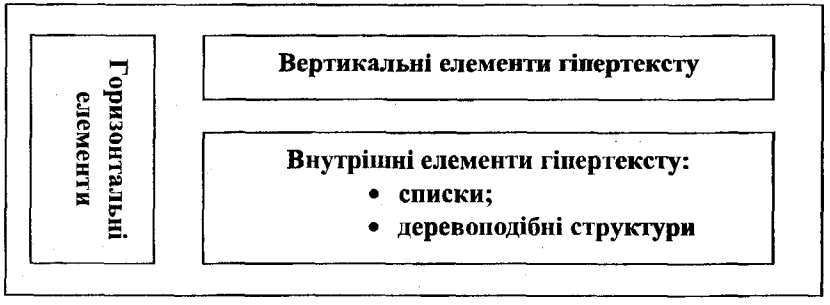



*Рис. 2.11. Розміщення елементів гіпертексту на Веб-сторінці*

## Контрольні запитання

1. Проаналізуйте зміст гіпертекстової моделі знань.
2. Поясніть особливості гіпертекстових моделей.

3. Проаналізуйте зміст і призначення гіпертекстової технології.
4. Проаналізуйте засоби створення гіпертекстових Веб-сторінок.
5. Назвіть типові гіпертекстові системи.

## 2.8. Моделі знань у формі продукційних правил

Вивчення процесів розв'язування задач людиною стимулювало появу методів комп'ютерної обробки інформації для моделювання людської свідомості і пам'яті. Вчений Ньюел із Станфордського університету запропонував ідею, суть якої полягала в тому, щоб подати довготривалу пам'ять людини як послідовність правил типу "людина–дія", названих продукціями, а оперативну пам'ять подати як набір ситуацій. Кожне правило по суті має вигляд: "Якщо розпізнана ситуація **S** в оперативній пам'яті, то реалізується дія **A**." Дія приводить до зміни змісту оперативної пам'яті. Після зміни змісту виникає нова ситуація, яка активізує нові правила.

Ньюел використав процедуру активізації правил і зміну пам'яті для того, щоб змоделювати процес пошуку людиною розв'язку проблеми і назвав отриману систему "продукційною системою" [27]. Подальший розвиток продукційних систем привів до появи терміна "інженерія знань".

Поширеність моделей знань на основі правил-продукцій обумовлена такими характеристиками:

1. Продукційні правила адекватно відображають знання людини у вигляді асоціативних міркувань, емпіричних спостережень, навичок. Це забезпечує моделювання знань на поверхневому рівні, без відображення глибинної суті знань і причинних зв'язків;
2. Правила є ефективним способом подання висновків у формі рекомендацій, директив, стратегій, які обумовлюються певними фактами-знаннями з людського досвіду;
3. Правила також забезпечують природний спосіб опису процесів, які виникають і керуються в складному і швидкозмінному зовнішньому середовищі.



Схематично продукція подається у формі символьного виразу:

**продукція = умова + наслідок.**

Тут умова може бути символьною або функціональною. Формально символьна умова продукції перевіряється на основі зіставлення певного зразка (твердження) із даними в моделі знань. Функціональна умова (вираз, формула, процедура) продукції перевіряється на основі відповідних обчислень. Якщо зіставлення (підтвердження) успішне, то виконується наслідок (дія, процедура, рекомендація).

Найпоширенішими в літературі є подання правил у формі логічних тверджень:

**Якщо** (умова), **то** (наслідок, дія);

**Якщо** (умова 1, умова 2, умова 3), **то** (наслідок 1, наслідок 2).

У поданих записах умови і наслідки об'єднуються за правилом кон'юнкції, тобто можуть наставати одночасно. Дією або наслідком можуть бути приєднана обчислювальна процедура, рекомендація, варіант рішення. Правила, які містять декілька висновків, іноді називають багатокритеріальними. Система продукцій утворює модель знань для розв'язання задачі.

Відповідним аналогом для продукційного правила є поширене в класичних мовах програмування логічне правило :

IF (умова, сукупність умов), THEN (дія), ELSE (дія).

Якщо умови правила зв'язані логічними зв'язками АБО, то правило з одним висновком чи декількома, об'єднаними зв'язкою І, можна подати за допомогою оператора варіанта – CASE.

Структури даних для правил можна зображувати у вигляді графових та деревоподібних структур або у формі таблиць. Таке представлення використовується у підсистемах пояснення експертних систем за допомогою графічних засобів візуалізації знань.

**Мережі виведення для правил.** Системи продукцій зручно зображати у вигляді графів, які отримали назву мереж виведення, або семантичних мереж правил. Умови продукційного правила

являють собою набір умов, об'єднаних І- і АБО-логічними зв'язками. Сукупність таких правил можна представити у виді І/АБО-графа, роль вузлів у ньому будуть виконувати асоціативні групи умов, або їхні комбінації. Дуги між вузлами відповідатимуть зв'язкам між асоціативними групами даних-фактів з передумов і висновками продукційних правил.

Мережі виведення в літературі називають семантичними мережами правил.

Приклад продукційних правил для задачі діагностики систем автомобіля:

1. *Якщо двигун не заводиться і стартер працює, то перевірити електричну систему і систему подачі пального;*
2. *Якщо стартер працює незадовільно, то перевірити акумулятор;*
3. *Якщо акумулятор розряджений, то перевірити якість електроліту і зарядити;*
4. *Якщо стартер працює задовільно, то перевірити систему запалювання;*
5. *Якщо висока напруга генерується і розподільник є справний, то перевірити свічі запалювання;*
6. *Якщо свічі запалювання містять нагар або мають неправильний люфт, то усунути ці дефекти.*

Продукційні моделі застосовують для розв'язання широкого класу експертних задач та багатьох задач ШІ:

- діагностики;
- класифікації;
- прийняття рішень;
- керування;
- в ігрових задачах;
- задачах бізнес-аналізу.

У цих задачах процес розв'язання визначається даними, розгалуження є частими, скоріше нормою. Тому правила є ефективним способом вибору стратегій, директив, розв'язків. Правила



можна модифікувати: розширювати, ускладнювати. Для детального моделювання знань продукційні моделі поєднують з мережевими, фреймовими та гіпертекстовими моделями знань.

Зазначимо основні характеристики продукційних моделей:

1. Універсальність для широкого класу задач. Правила з використанням шаблонів, стратегій, аксіом можна записувати незалежно від програми у вигляді моделі або бази знань;
2. Природність виведення розв'язків за аналогією з міркуваннями експерта або фахівця;
3. Можливість приєднання нечітких обчислень для знань з нечіткими даними;
4. Зручність для обчислень в системах з паралельною архітектурою. Це обумовлено природним паралелізмом в системі продукцій та асинхронним процесом виконання правил;
5. Відсутність чіткої теорії побудови продукційних правил. Часто вони будуються на основі досвіду, евристик. Поняття продукцій бувають розпливчастими. Способи керування системою продукцій також використовують евристичні алгоритми.

Теорія продукційних систем може бути побудована на основі теорії паралельних інформаційних процесів, яка не є розробленою.

Інша проблема – перехід від статичних моделей до динамічних, де змінюється набір продукцій в системі або перебудовується алгоритм керування вибором продукцій. Такі системи називаються адаптивними. В них використовуються гнучкі стратегії керування різного рівня складності і швидкості обробки. Тут важливу роль відіграють засоби апаратної підтримки.

Основні вимоги до побудови системи правил:

– ***Повнота***. Набір правил вважають повним, якщо для не порожньої бази фактів є досяжним логічний висновок для кожного правила;

– ***Коректність***. Правило є коректним, якщо воно не містить рекурсивних висновків (викликів) для самого себе;
– ***Несуперечливість***. Два правила є суперечливими, якщо вони за однакових умов породжують різні висновки.

## 2.9. Засоби для розроблення продукційних моделей знань

Розвиток продукційних обчислень сприяв розробленню мов програмування – логічних та функційних – таких, як Пролог та Ліспі [15]. Ці мови забезпечують ефективне опрацювання задач із символьними даними та підтримують ряд ефективних процедур символьних обчислень, таких як рекурсія, бектрекінг, пошук у мережевих ієрархічних структурах. Сьогодні існують об'єктно-орієнтовані версії цих мов. Ефективнішими для побудови і відлагодження продукційних моделей знань є середовища і оболонки СШІ. Розглянемо характеристики деяких засобів [17].

**ART (Inference Corporation)** поєднує два основні формалізми представлення знань: правила для процедурних і фреймоподібні структури для декларативних знань [ ]. Головним є формалізм продукційних правил. Декларативні знання описуються через факти і схеми [schemata] і в деяких випадках через зразки [patterns]. Крім факторів числової невизначеності, що зв'язуються з індивідуальними фактами, в ART розрізняються факти, що явно приймаються за помилкові, і факти, істинність яких невідома. Можливе використання логічних залежностей. Користувачу надається добрий графічний редактор правил, використовуваний для початкового введення продукцій і корекції їх у процесі налагодження, і засоби графічного трасування висновків.

**Механізм Viewpoint** допускає утворення декількох конкуруючих моделей, для яких пробуються альтернативні рішення.

Схема в ART – це макроформа для подання таксономічних знань у структурованому вигляді. Це фактично є семантична



мережа, описана згідно з таксономією ПРО. ART забезпечує 11 системно визначених властивостей, спадкування яких підтримується автоматично. Допускається множинне спадкування. Однак засоби задання активних значень, вказівки обмежень на слот і прив'язки процедурних знань до слотів тут відсутні. Отже, роль компонента, заснована на фреймах, є суто описовою.

Спадкування, обумовлене користувачем, не допускається, але теж може бути змодельоване за допомогою правил. Відповідно до концепції фірми Inference Corporation це є перевагою, тому що статичне спадкування передбачає перекомпіляцію ефективного коду.

Продукційні знання описують за допомогою правил п'яти видів:

1. Правила висновків.
2. Продукційні правила.
3. Гіпотетичні правила.
4. Правила обмежень.
5. Правила посилань.

*Правила висновків* додають факти в базу знань, тоді як продукційні правила змінюють факти в робочій пам'яті (наприклад, значення атрибута об'єкта).

*Гіпотетичні правила* дозволяють в ART використовувати можливості формування гіпотез і представляються у такій формі: "ЯКЩО це сталось, То розглядати це як гіпотезу".

*Правила обмежень* описують ситуацію, що ніколи не може з'явитися за правильного погляду на реальність.

*Правила посилань* використовуються для припущень (які приймають як правильні) точки зору (про гіпотези). При підтвердженні гіпотезою деякої умови вона приймається як правильна, поєднується з усіма гіпотезами, що її породили, і стає новим коренем у деревоподібній структурі моделі правил. Інші, несумісні, гіпотези відкидаються.

Виклик процедур, визначених користувачем, може бути використаний як у лівих, так і в правих частинах правил ART. Зразки використовуються в умовній частині правил. Вони повинні бути зіставлені з фактами робочої пам'яті. Зразки можуть містити змінні і логічні зв'язки, що забезпечують сполучення фрагментів моделі (І, АБО, НЕ, ІСНУЄ, ДЛЯ УСІХ). У складі ART використовуються і класичні правила типу OPS.

**Графічний інтерфейс.** Графічне оточення ART добре розвинуто. Інтерфейс **ARTStudio** містить базу знань з демонстрацією гіпотез, утиліти налагоджувача програм, що запускаються, систему підказок, доступну в будь-який час, систему меню і графічний пакет ARtist (ART Image Syntesis Tool) з віконним редактором.

**ARTist** дає змогу створювати доступні правилам меню і керувати вікнами користувацького інтерфейсу, а також створювати самі вікна. Графічні конструкції описуються за допомогою схем і посилаються на правила. Це забезпечує функціонування за принципом керування звертаннями до даних. ART часто подають як кращу ІС для створення експертних систем, але це середовище відповідає лише всім вимогам підходу на основі поверхневих знань. Завдяки компілятору правил система пошуку висновку в ART є швидкою за своєю природою. Перші версії ART ґрунтувалися на мові Лісп, останні реалізовані безпосередньо на Сі.

**GUESS/1** є мовою інженерії знань для подання, заснованого на правилах. Підтримує такі структури, як таблиці відношень, ієрархічні дерева, семантичні мережі і фрейми [21]. Знання, використовувані для керування, представлені у фреймах і правилах, що допускають прямий і зворотний ланцюг міркувань. Фрейми можуть викликати виконання правил, і правила можуть викликати фрейми.



## Контрольні запитання

1. Проаналізуйте зміст моделі знань на основі продукційних правил.
2. Поясніть зміст і форму продукційного правила.
3. Проаналізуйте основні характеристики продукційних моделей.
4. Поясніть призначення мереж виведення для правил.
5. Назвіть типові системи для побудови продукційних моделей.
6. Поясніть основні характеристики середовища ART.

# Розділ 3
# МОДЕЛЮВАННЯ ЗНАНЬ З НЕЧІТКИМИ ДАНИМИ

## 3.1. Елементи теорії нечітких множин

Неточні методи призначені для операцій з нечіткими даними і відіграють важливу роль у розробленні експертних систем. Багато суперечок викликає питання, які саме методи повинні використовуватися. Враховують на практиці дві головні причини:

- точних методів не існує;
- точні методи існують, але не можуть бути застосовані на практиці через відсутність необхідного обсягу даних, або неможливості їхнього збирання з огляду на вартість, ризик, або обмеженнями часу на збирання.

Розглянемо основні поняття і визначення в теорії нечітких множин, взяті з [2, 8, 25]. Математична теорія нечітких множин, яку запропонував Л. Заде, дає змогу описувати нечіткі поняття і знання, оперувати цими знаннями і робити нечіткі висновки. Л. Заде розширив класичне поняття множини, припустивши, що характеристична функція (функція належності елемента множині) може набувати довільних значень в інтервалі [0;1]. Такі множини він назвав **нечіткими** (*fuzzy*). Л. Заде визначив також ряд операцій над нечіткими множинами.

**Визначення 1.** *Нечіткою множиною (fuzzy set)* A на універсальній множині X називається сукупність пар $(x,\mu_A(x))$, де $\mu_A(x)$ – функція належності елемента x до нечіткої множини A. Функція належності – це число з діапазону [0, 1]. Вище значення функції належності показує, що елемент універсальної множини більшою мірою відповідає властивостям нечіткої множини.

**Визначення 2.** *Функцією належності (membership function)* називається функція, що дає змогу обчислити ступінь належності довільного елемента універсальної множини до нечіткої множини.



**Приклад 1.** Формалізуємо за допомогою нечіткої множини поняття "високий зріст". Для цього визначимо функцію належності відповідної нечіткої множини. Базовою множиною є множина невід'ємних дійсних чисел, що характеризують зріст людини. Функцію належності можна задати аналітичною формулою:

$$\mu_A(x) = \begin{cases} 0, x\langle 1.5; \\ 2x-3, 1.5 \le x \le 2; \\ 1, x\rangle 2. \end{cases}$$

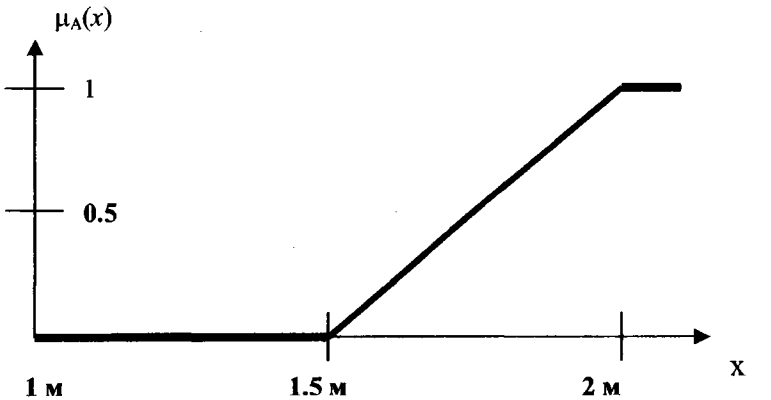



*Рис. 3.1. Функція належності поняття "високий зріст"*

Нечітку множину запишемо у вигляді:

A= {(1.5,0),(1.6,0.2), (1.7,0.4), (1.8,0.6), (1.9,0.8), (2,1)}.

На основі цієї множини можна інтерпретувати зріст 1.85 м для людини за ступенем належності 0.7 як доволі високий.

Особливості вибору форми функції належності:

- функція належності є суб'єктивною характеристикою і не може бути визначена однозначно; кожний фахівець може запропонувати власне її значення;
- самі ступені належності можна розглядати як ступені першого роду (з конкретними числовими значеннями) і

ступені другого роду (з інтервальними оцінками). На практиці найчастіше використовують нечіткі множини першого роду;

- незважаючи на те, що функція належності має суб'єктивний характер, її не можна вибирати довільно. Її вибір слід проводити на основі досвіду, спостережень за відповідними явищами, процесами, характеристиками, які подаються за допомогою моделей знань.

## 3.2. Операції над нечіткими множинами

Над нечіткими множинами є визначеним широкий клас операцій. Розглянемо деякі з них [10].

***Рівність***. **A** і **B** рівні, якщо $\forall x \in E\ \mu_A(x) = \mu_B(x)$.

***Перетин***. Перетином двох множин **A** і **B** називається множина C=**A**∩**B** з функцією належності

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)).$$

***Об'єднання.*** Перетином двох множин **A** і **B** називається множина C= **A** ∪ **B** з функцією належності

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)).$$

***Різниця.*** Різницею двох множин **A** і **B** називається множина

C = **A** – **B** з функцією належності:

$$\mu_C(x) = \min(\mu_A(x), 1 - \mu_B(x)).$$

**Алгебраїчний добуток A і B** позначається **A·B** і визначається так:

$$\forall x \in E\ \mu_{A\cdot B}(x) = \mu_A(x)\mu_B(x).$$

**Алгебраїчна сума множин** позначається $A \dot{+} B$ і визначається так:

$$\forall x \in E\ \mu_{A\dot{+}B} = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x).$$



**Приклад 1.**

Нехай задано множини:

$$\mathbf{A} = 0{,}4/\mathbf{x}_1 + 0{,}2/\mathbf{x}_2 + 0/\mathbf{x}_3 + 1/\mathbf{x}_4;$$
$$\mathbf{B} = 0{,}7/\mathbf{x}_1 + 0{,}9/\mathbf{x}_2 + 0{,}1/\mathbf{x}_3 + 1/\mathbf{x}_4;$$
$$\mathbf{C} = 0{,}1/\mathbf{x}_1 + 1/\mathbf{x}_2 + 0{,}2/\mathbf{x}_3 + 0{,}9/\mathbf{x}_4.$$

Результати застосування операцій:

1. **Рівність. A ≠ B ≠ C.**
2. **Перетин. A∩B** $= 0{,}4/\mathbf{x}_1 + 0{,}2/\mathbf{x}_2 + 0/\mathbf{x}_3 + 1/\mathbf{x}_4$.
1. **Об'єднання A∪B** $= 0{,}7/\mathbf{x}_1 + 0{,}9/\mathbf{x}_2 + 0{,}1/\mathbf{x}_3 + 1/\mathbf{x}_4$.
2. **Різниця. A – B** $= 0{,}3/\mathbf{x}_1 + 0{,}1/\mathbf{x}_2 + 0/\mathbf{x}_3 + 0/\mathbf{x}_4$;
3. **Різниця. B – A** $= 0{,}6/\mathbf{x}_1 + 0{,}8/\mathbf{x}_2 + 0{,}1/\mathbf{x}_3 + 0/\mathbf{x}_4$.

## 3.3. Нечіткі відношення

Нехай $\mathbf{E} = \mathbf{E}_1 \times \mathbf{E}_2 \times \ldots \times \mathbf{E}_n$ – прямий добуток універсальних множин і **M** – деяка множина функцій належностей (наприклад **M** = = [0,1]). Нечітке n-не відношення визначається як нечітка підмножина **R** на **E**, що набуває своїх значень в **M**. У випадку **n**=2 і **M** = [0,1] нечітким відношенням **R** між множинами **X** = $\mathbf{E}_1$ і **Y** = $\mathbf{E}_2$ буде називатися функція **R:(X,Y)**→ [0,1], що ставить у відповідність кожній парі елементів (x,y)∈**X**×**Y** величину $\mu_R(x,y) \in [0,1]$. Позначення: нечітке відношення на **X**×**Y** запишеться у вигляді: x∈**X**, y∈**Y**: **xRy**. У випадку, коли **X** = **Y**, тобто **X** і **Y** збігаються, нечітке відношення **R:** X×X→[0,1] називається нечітким відношенням на множині **X**.

**Приклади:**

1. Нехай **X** = $\{x_1,x_2,x_3\}$, **Y** = $\{y_1,y_2,y_3,y_4\}$, **M** = [0,1]. Нечітке відношення **R=XRY** може бути задано у формі таблиці:

| | $y_1$ | $y_2$ | $y_3$ | $y_4$ |
|---|---|---|---|---|
| $x_1$ | 0 | 0 | 0,1 | 0,3 |
| $x_2$ | 0 | 0,8 | 1 | 0,7 |
| $x_3$ | 1 | 0,5 | 0,6 | 1 |

**Операції над нечіткими відношеннями**

**Об'єднання двох відношень** позначається $\mathbf{R_1 \cup R_2}$ і визначається виразом:

$$\mu_{R1 \cup R2}(x,y) = \max(\mu_{R1}(x,y) \vee \mu_{R2}(x,y))$$

| $\mathbf{R_1}$ | | | |
|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ |
| $x_1$ | 0,1 | 0 | 0,8 |
| $x_2$ | 1 | 0,7 | 0 |

| $\mathbf{R_2}$ | | | |
|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ |
| $x_1$ | 0,7 | 0,9 | 1 |
| $x_2$ | 0,3 | 0,4 | 0,5 |

| $\mathbf{R_1 \cup R_2}$ | | | |
|---|---|---|---|
| | $y_1$ | $y_2$ | $y_3$ |
| $x_1$ | 0,7 | 0,9 | 1 |
| $x_2$ | 1 | 0,7 | 0,5 |

**Перетинання двох відношень** $\mathbf{R_1}$ і $\mathbf{R_2}$ позначається $\mathbf{R_1 \cap R_2}$ і визначається виразом:

$$\mu_{R1 \cap R2}(x,y) = \min(\mu_{R1}(x,y) \wedge \mu_{R2}(x,y)).$$

**Алгебраїчний добуток двох відношень** $\mathbf{R_1}$ і $\mathbf{R_2}$ позначається $\mathbf{R_1 \cdot R_2}$ і визначається виразом:

$$\mu_{R1 \cdot R2}(x,y) = \mu_{R1}(x,y) \cdot \mu_{R2}(x,y).$$

**Композиція двох нечітких відношень**

Нехай $R_1$ – нечітке відношення $\mathbf{R_1}$: $(\mathbf{X} \times \mathbf{Y}) \rightarrow [0,1]$ між X і Y, і $R_2$ – нечітке відношення $\mathbf{R_2}$: $(\mathbf{Y} \times \mathbf{Z}) \rightarrow [0,1]$ між Y і Z. Нечітке відношення між X і Z, позначуване $\mathbf{R_2 \bullet R_1}$, подається виразом

$$\mu_{R1 \bullet R2}(x,z) = \max[\mu_{R1}(x,y) \wedge \mu_{R1}(y,z)],$$

і називається (max-min)-композицією відношень $\mathbf{R_1}$ і $\mathbf{R_2}$. Наведемо програмну реалізацію алгоритму (max-min)-композиції двох відношень, написану мовою Visual Basic. Програма використовує в ролі масивів для відношень три активні елементи-об'єкти типу Spreadsheet.

**Програма 1.**

```
/ АЛГОРИТМ НЕЧІТКИХ ВІДНОШЕНЬ
Private Sub CommandButton1_Click()
Dim irow, icol, r1, c1, c2 As Integer
```



```
Dim mp(10) As Single
Set spree1 = Spreadsheet1
Set spree2 = Spreadsheet2
Set spree3 = Spreadsheet3
'вибір(відмічення) масиву даних
spree1.Range("c3:l7").Select
spree2.Range("c3:g12").Select
spree3.Range("c3:g7").Select
' Set Mdata – об"єктна змінна
'Set Mdata = Application.Selection
Set mdata1 = spree1.Selection
Set mdata2 = spree2.Selection
Set Mdata3 = spree3.Selection
For r1 = 1 To 5
 For c1 = 1 To 5
 For c2 = 1 To 10
 mp(c2) = mdata2(c2, c1)
 If mdata1(r1, c2) < mdata2(c2, c1) Then mp(c2) = mdata1(r1, c2)
 Next c2
 max1 = mp(1)
 For c2 = 2 To 10
 If mp(c2) > max1 Then max1 = mp(c2)
 Next c2
 Mdata3(r1, c1) = max1
 Next c1
 Next r1
 End Sub
```

## 3.4. Методи побудови функцій належності

У багатьох задачах використовують ***прямі*** методи, коли експерт або просто задає для кожного $x \in E$ значення $\in_A(x)$, або визначає функцію належності. Як правило, прямі методи

обчислення функції належності використовуються для вимірних, параметричних понять, таких як швидкість, час, відстань, тиск, температура тощо, або коли виділяються полярні значення об'єктів.

Часто для характеристик об'єкта можна виділити набір ознак і для кожної з них визначити полярні значення, що відповідають значенням функції належності, 0 або 1. Наприклад у задачі розпізнавання осіб можна застосувати класифікатор з бінарною шкалою, поданою у формі:

| | | 0 | 1 |
|---|---|---|---|
| $x_1$ | висота чола | низький | широкий |
| $x_2$ | профіль носа | кирпатий | з горбинкою |
| $x_3$ | довжина носа | короткий | довгий |
| $x_4$ | розріз очей | вузькі | широкі |
| $x_5$ | колір очей | світлі | темні |
| $x_6$ | форма підборіддя | гостре | квадратне |
| $x_7$ | товщина губ | тонкі | товсті |
| $x_8$ | колір обличчя особи | темний | світлий |
| $x_9$ | обличчя особи | овальне | кругле |

Для конкретної особи А (для якої, можливо, існує фотографія) експерт на основі наведеної шкали задає $\mu_A(x) \in [0,1]$, формуючи векторну функцію належності $\{\mu_A(x_1), \mu_A(x_2),..., \mu_A(x_9)\}$.

Як прямі методи використовують **також групові прямі методи**, коли, наприклад, групі експертів пред'являють конкретну



особу і кожний повинен дати одну із двох відповідей: "*ця людина лиса*" *або "ця людина не лиса"*, тоді кількість позитивних відповідей, поділена на загальне число експертів, дає значення для $\in$ *"лисий"* (даної особи).

***Непрямі*** методи визначення значень функції належності використовуються у випадках, коли немає елементарних вимірних властивостей, через які визначається нечітка множина. Як правило, це методи попарних порівнянь. Якби значення функцій належності були нам відомі, наприклад, $\mu_A(x_i) = w_i$, $i=1,2,...,n$, то попарні порівняння можна подати матрицею відношень $\mathbf{A} = \{a_{ij}\}$, де $a_{ij}=w_i/w_j$ (операція розподілу).

На практиці експерт сам формує матрицю А, при цьому передбачається, що діагональні елементи дорівнюють 1, а для елементів симетричних щодо діагоналі $a_{ij} = 1/a_{ji}$, тобто якщо один елемент оцінюється в к разів сильніше ніж інший, то цей останній має бути в 1/к разів сильнішим ніж перший. У загальному випадку задача зводиться до пошуку вектора $w$, що задовольняє рівняння виду $\mathbf{A}_w = \mu_{max} w$, де $\mu_{max}$ – найбільше власне значення матриці А.

## 3.5. Нечітке виведення на основі нечітких множин

Процес нечіткого виведення містить процедуру або алгоритм отримання нечітких висновків на основі нечітких умов з використанням апарату нечіткої логіки [2]. Нечіткі правила їх умови і висновки формулюються у формі нечітких висловлювань відносно значень тих чи інших нечітких змінних.

Дано нечітке продукційне правило – Якщо А, то В.

Спостерігається параметр, належний частині умови в правилі А.

Яким повинен бути висновок В?

Нехай задано базові носії – множини параметрів U, V та їх нечіткі множини A та B. Задано: або елемент u є U, або нечітка множина A' є U. Потрібно визначити елемент v є V, що є *висновком* нечіткого логічного правила.

**Приклад 1**

**Задано нечітке продукційне правило:**

*Якщо студент багато працює в бібліотеці,*

*То він отримає високу оцінку.*

У цьому правилі є дві нечіткі оцінки: вхідна – *багато працює*; вихідна – *висока оцінка*. Якщо ці нечіткі оцінки подати за допомогою нечітких множин, то виникає задача пошуку чіткого значення вихідної оцінки. Приклад і його аналіз взято з [2].

Задамо в ролі множини U множину чисел, що визначають кількість годин, які проводить студент у бібліотеці за тиждень. Можна як U взяти діапазон чисел від 0 до 30. Візьмемо таку множину значень U = [0, 3, 6, 9, 12, 18, 21, 27].

Якщо рейтинги визначають за 100-бальною шкалою, то як V можна взяти множину V = [59, 72, 84, 91, 96, 100].

Задамо дві нечіткі множини: A – *робота в бібліотеці*, та B – рейтинг.

A = {(3,0), (6, 0.1), (9, 0.4), (12,0.6), (18,0.8), (21, 1), (27,1)}.

B = {(59,0), (72, 0.2), (84, 0.4), (91,0.7), (96,0.9), (100, 1)}.

Нехай задано кількість годин, які проводить студент в бібліотеці. Для отримання висновків можна застосувати метод простої підстановки нечіткого значення.

Нехай відомо, що студент працює 9 год. Тоді функція належності з нечіткої множини A дорівнює 0.4 і висновок є таким: студент повинен отримати оцінку **добре** за рейтингом 84, отриманим з множини B. Якби в множині B не виявилось значення для функції належності, яке дорівнює 0.4, тоді слід взяти найближче значення або провести лінійну інтерполяцію між двома сусідніми значеннями.



**Модель правила для програмування**

**IF** (задане значення часу належить множині A (точно або наближено) )

**THEN** (шукати за значенням функції належності з множини A відповідь у множині B)

**ELSE** (Повідомлення "ЗАДАНЕ ЗНАЧЕННЯ ЧАСУ НЕ ВІДПОВІДАЄ КОНТЕКСТУ ЗАДАЧІ")

**Приклад 2**

Задано не число, а нечітка множина, яка описує, наприклад, *середню кількість часу*, яку проводить студент у бібліотеці.

Для цього випадку застосовують алгоритм нечіткого логічного виведення на основі композиційного правила Заде. Обчислюють відношення нечіткої імплікації двох множин A – B. Результатом побудови такого відношення є таблиця (матриця) функцій належності. Розглянемо окремі види відношень нечіткої імплікації.

**Нечітка min-імплікація:**

$$m(u_i, v_j) = \min(m_A(u_i), m_B(v_j)).$$

**Нечітке розширення класичної імплікації:**

$$m(u_i, v_j) = \max(m_B(v_i), 1-m_A(u_j)).$$

**Нечітка імплікація Лукашевича:**

$$m(u_i,v_j) = \min(1,1-m_A(u_i)+ m_B(v_j)).$$

Після застосування операції імплікації можна отримати результуючу множину B' – нечітку множину висновків, які відповідають вхідній множині A'. Множина B' буде результатом композиції множини A' та нечіткої min-імплікації множин A і B:

$$B' = \max(\min(A' , (A^* B) )) ;$$

тут max-min – операція композиції.

На основі множини B' необхідно знайти числову відповідь для значення базового параметра – оцінки рейтингу. Відповідна процедура пошуку відповіді має назву "дефадзифікація". Найчастіше застосовують процедуру обчислення центра тяжіння нечіткої множини B' за формулою

$$v^* = \sum v_i \, m_B'(v_i) / m_B'(v_i). \qquad (3.1)$$

Розглянутий метод нечіткого виведення називають методом центру тяжіння композиції max-min.

Виконаємо практичне застосування методу нечіткого виведення на основі розглянутих вище нечітких множин:

A = {(3,0), (6, 0.1), (9, 0.4), (12,0.6), (18,0.8), (21, 1), (27,1)};

B = {(59,0), (72, 0.2), (84, 0.4), (91,0.7), (96,0.9), (100, 1)}.

Задамо вхідну нечітку множину A' – *середню кількість часу* роботи студента в бібліотеці. Це може бути віртуальний студент.

A' = {(3,0); (6,0.2); (9,0.7); (12,1); (18,0.6); (21,0.2); (27,0)}.

Обчислюємо відношення min – імплікації нечітких множин A та B. Результат подамо у формі матриці.

| 1 | | *59* | *72* | *84* | *91* | *96* | *100* |
|---|---|---|---|---|---|---|---|
| | ***A/B*** | ***0*** | ***0.2*** | ***0.4*** | ***0.7*** | ***0.9*** | ***1*** |
| ***3*** | ***0*** | 0 | 0 | 0 | 0 | 0 | 0 |
| ***6*** | ***0.1*** | 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| ***9*** | ***0.4*** | 0 | 0.2 | 0.4 | 0.4 | 0.4 | 0.4 |
| ***12*** | ***0.6*** | 0 | 0.2 | 0.4 | 0.6 | 0.6 | 0.6 |
| ***18*** | ***0.8*** | 0 | 0.2 | 0.4 | 0.7 | 0.8 | 0.8 |
| ***21*** | ***1*** | 0 | 0.2 | 0.4 | 0.7 | 0.9 | 1 |
| ***27*** | ***1*** | 0 | 0.2 | 0.4 | 0.7 | 0.9 | 1 |

Знайдемо max-min-композицію нечіткої множини A' та знайденого вище відношення у формі матриці. Результатом буде нечітка множина B'. Процес побудови подамо у формі таблиці. Операції побудови є такими. Порівнюється послідовно перший елемент множини A' (значення функції належності) з кожним елементом першого рядка матриці імплікації і вибирається min-елемент, записується у відповідну позицію таблиці. Такі операції проводяться з усіма елементами множини A'. Потім з кожного стовпця таблиці вибирається max-елемент і записується у позиції нечіткої множини B'. Водночас вибираються значення базового параметра – рейтингу із множини B.



| 2 | | *59* | *72* | *84* | *91* | *96* | *100* |
|---|---|---|---|---|---|---|---|
| | ***A'/B*** | ***0*** | ***0.2*** | ***0.4*** | ***0.7*** | ***0.9*** | ***1*** |
| ***3*** | ***0*** | 0 | 0 | 0 | 0 | 0 | 0 |
| ***6*** | ***0.2*** | 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| ***9*** | ***0.7*** | 0 | 0.2 | 0.4 | 0.4 | 0.4 | 0.4 |
| ***12*** | ***1*** | 0 | 0.2 | 0.4 | 0.6 | 0.6 | 0.6 |
| ***18*** | ***0.6*** | 0 | 0.2 | 0.4 | 0.7 | 0.8 | 0.8 |
| ***21*** | ***0.2*** | 0 | 0.2 | 0.4 | 0.7 | 0.9 | 1 |
| ***27*** | ***0*** | 0 | 0 | 0 | 0 | 0 | 0 |
| | ***B'*** | ***0*** | ***0.2*** | ***0.4*** | ***0.6*** | ***0.6*** | ***0.6*** |
| | | ***59*** | ***72*** | ***84*** | ***91*** | ***96*** | ***100*** |

Проведемо операцію дефадзифікації множини B' за формулою (3.1):

$$v^* = (59*0 + 72*0.2 + 84*0.4 + 91*0.6 + 96*0.6 + 100*0.6)/(0.2 + 0.4 + 0.6 + 0.6 + 0.6 + 0.6) = 91.75 \approx 92.$$

Отже, на основі проведеного розрахунку із застосуванням нечіткого правила виведення отримуємо висновок, що студент повинен отримати оцінку 92 бали.

**Модель правила для програмування**

**IF** (задано множини A, B, A')

**THEN** (шукати за описаним алгоритмом відповідь)

**ELSE**(Повідомлення "ЗАДАНІ вхідні дані НЕ ВІДПОВІДАЮТЬ КОНТЕКСТУ ЗАДАЧІ")

## 3.6. Продукційні правила з поданням нечітких даних на основі коефіцієнтів визначеності

Графічне подання мережі правил як моделі знань дає змогу наочно показати можливі способи приєднання до правил обчислень для нечітких даних. В мережах правил існують різні види логічних зв'язків між фактами та правилами. Для окремих

базових зв'язків застосовують формули нечіткої логіки Заде для обчислення коефіцієнтів визначеності.

Розглянемо ці базові зв'язки та формули для обчислень. Подані на рисунках позначення CF, CF1, CF2, CFP – це задані апріорі значення коефіцієнтів визначеності для елементів мережі – фактів F та правил P. У формулах використовують значення коефіцієнтів визначеності в межах від 0 до 100, як значення відсотків.

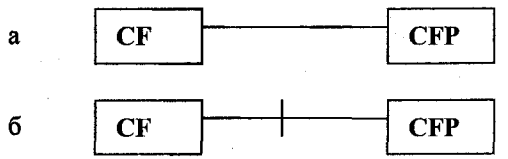



*Рис. 3.2. Графічне зображення двох елементів мережі*

а) ***Правило***: ЯКЩО ***F***, ТО ***P.*** Формули для обчислення CF:
CF(P) = CF, якщо CFP=0 ;
CF(P) = CF*CFP/100 , якщо CFP ≠0;

б) ***Правило***: ЯКЩО ***НЕ F***, ТО ***P.*** Формули для обчислення CF:
CF(P) =100 – CF, якщо CFP=0 ;
CF(P) = (100-CF)*CFP/100 , якщо CFP ≠0;

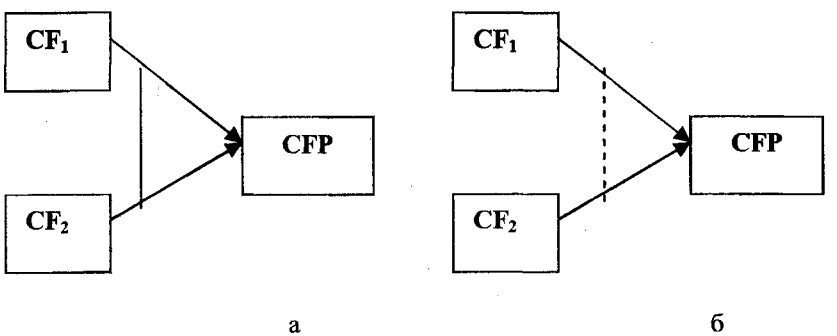



*Рис. 3.3. Графічне зображення для елементів мережі, об'єднаних за логічними формулами: а – AND (І); б – OR.(АБО)*



а) ***Правило***: ЯКЩО ***(F1 AND F2)***, ТО ***P.*** Формули для обчислення CF:
CF(P) = (CF1 and CF2) = min(CF1, CF2) якщо CFP=0 ;
CF(P) = (CF1 and CF2) = min(CF1, CF2)*CFP /100, якщо CFP ≠0;

б) ***Правило***: ЯКЩО ***(F1 OR F2)***, ТО ***P.*** Формули для обчислення CF:
CF(P) = (CF1 or CF2) = max(CF1, CF2); якщо CFP=0;
CF(P) = (CF1 or CF2) = max(CF1, CF2)*CFP /100, якщо CFP ≠0.

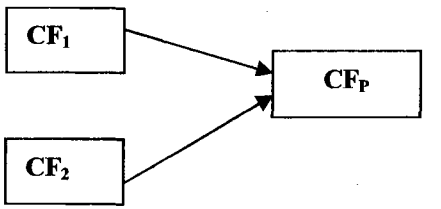



*Рис. 3.4. Графічне зображення для незалежних елементів мережі*

Обчислення коефіцієнтів визначеності правила для незалежних за впливом елементів мережі, зображених **на рис. 3,** виконують за нижчеподаними формулами:

- CF(P) = (CF1, CF2) = CF1+CF2 –CF1*CF2/100, якщо CFP=0 ;
- CF(P) = (CF1, CF2) = (CF1+CF2 –CF1*CF2/100)*CFP/100, якщо CFP ≠0.

Для складніших видів мережі правил використовують комбінування поданих вище формул. Для трьох і більше елементів об'єднання виділяють послідовно попарні об'єднання і проводять обчислення для всіх пар. Порядок об'єднання не має значення.

Вибір інтервалу значень для коефіцієнтів визначеності залежить від контексту задачі і має суб'єктивний характер.

**Приклад 2**

Розглянемо побудову мережі правил для задачі розпізнавання об'єкта на основі сукупності заданих ознак. Нехай об'єкт –

поштову марку – задано такою сукупністю фактів-характеристик та властивостей:

1. Малі розміри;
2. Кольоровий;
3. Зубчасті краї;
4. Випускається серіями;
5. Продається на пошті.

Можливу семантичну мережу для нечітких правил розпізнавання наведено на рис. 3.5.

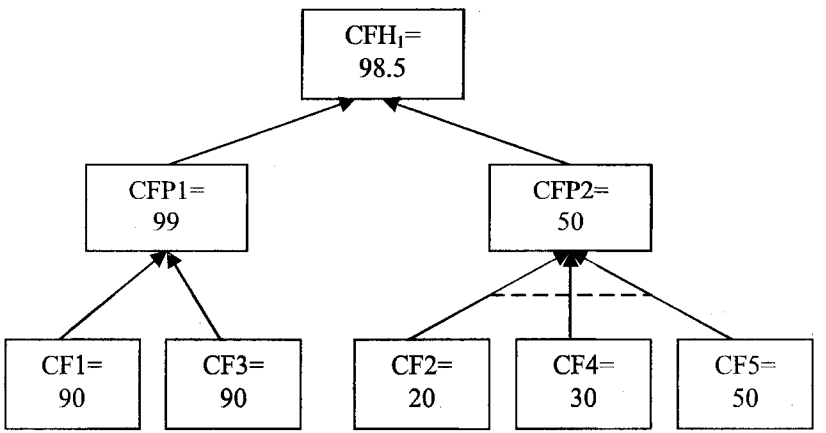



*Рис. 3.5. Мережі правил і обчислені коефіцієнти визначеності для розпізнавання поштового об'єкта*

Для поданої мережі логічні правила можна записати так:

- *Якщо об'єднати (F1 та F3), то обчислити CF(P1)* – це правило аналізу фактів- характеристик об'єкта;
- *Якщо (F2 або F4 або F5), то обчислити CF(P2)* – це правило аналізу фактів-властивостей об'єкта;
- *Якщо об'єднати (P1 та P2), то обчислити CF(H)* і показати.

**Особливості програмної реалізації.** Поданий приклад передбачає діалоговий режим розпізнавання. Різні способи об'єднання



фактів і правил дають різне значення коефіцієнта визначеності для розпізнаного об'єкта. Незалежно від виду мережі необхідно прагнути до максимізації коефіцієнта визначеності у результаті. Під час реалізації логічних правил їх необхідно протестувати на можливі варіанти діалогу (не всі введені чи вибрані факти, зовсім не вибрані тощо).

Вищерозглянуті моделі з нечіткими даними потребують:

- апріорі задання коефіцієнтів визначеності для фактів;
- приєднання формул обчислень до правил;
- вибір коефіцієнтів має суб'єктивний характер.

Цей приклад – це пошук висновку на основі одного правила. Однак на практиці використовують множину (базу) правил, яка опрацьовує набір вхідних значень і видає висновок.

Основні завдання для моделювання нечіткого виведення.

1. Формалізація задачі, визначення контексту і кількості правил;
2. Побудова вхідних нечітких множин для правил;
3. Побудова бази знань для правил;
4. Реалізація алгоритмів виведення;
5. Розроблення інтерфейсу для активізації, управління і контролю за процесом виведення.

## 3.7. Нечіткі і лінгвістичні змінні

Поняття нечіткої і лінгвістичної змінних використовують для опису об'єктів і явищ за допомогою нечітких множин. ***Нечітка змінна*** характеризується трійкою

<N, X, A>,

де N – найменування змінної, X – універсальна множина (область визначення N), A – нечітка множина на X, що описує значення функції належності (тобто $\mu_A(x)$) для значень нечіткої змінної N.

***Лінгвістичною змінною*** називається набір

<N,T,X,G,M>,

де N – назва лінгвістичної змінної; T – множина її семантичних значень (терм-множина), що являє собою назви нечітких змінних, областю визначення кожної з яких є базова множина X. Множина T називається базовою терм-множиною лінгвістичної змінної; X – базова змінна, яка задається параметричною множиною числових значень. Це змінна, відносно стану якої задається назва лінгвістичної змінної, наприклад, успішність у балах, температура у градусах; G – синтаксична процедура, що дає змогу оперувати елементами терм-множини T, зокрема, генерувати нові терми (значення). Множина T називається розширеною терм-множиною лінгвістичної змінної; M – семантична процедура, що дає змогу перетворити кожне значення лінгвістичної змінної, утвореної процедурою G, на нечітку змінну, тобто сформувати відповідну нечітку множину. Присвоєння декількох значень символам припускає, що контекст дає змогу можливі невизначеності.

**Приклад 3**

Нехай експерт визначає ціну виробу, що випускається, за допомогою понять *"мала* ціна", *"середня* ціна" і *"велика* ціна", при цьому мінімальна ціна дорівнює 10, а максимальна – 100. Нормалізувати такий опис можна за допомогою такої лінгвістичної змінної

<N, T, X, G, M>,

де b – ціна виробу; T – {*"мала ціна"*, *"середня ціна"*, *"велика ціна"*}; X – [10, 80]; G – процедура утворення нових термів за допомогою зв'язувань *"і"*, *"чи"* і модифікаторів типу *"дуже"*, *"не"*, *"злегка"* тощо. Наприклад: *"мала чи середня ціна"*, *"дуже мала ціна"* та ін.; M – процедура завдання на X = [10, 80] нечітких підмножин $A_1$= *"мала ціна"*, $A_2$ = *"середня ціна"*, $A_3$= *"велика ціна"*, а також нечітких множин для термів з G(T) відповідно до правил трансляції нечітких зв'язувань і модифікаторів *"і"*, *"чи"*, *"не"*, *"дуже"*, *"злегка"* та інші операції над нечіткими множинами.

Лінгвістична змінна використовується для пояснення лінгвістичної невизначеності, пов'язаної з якісними оцінками



об'єктів та їх характеристик за допомогою природної мови. Може вводитись для оцінювання інтенсивності, часу, логічних висновків. Лінгвістичні змінні як структури даних необхідні для моделювання висновків у задачах з нечіткими даними.

**Приклад 4**

Подамо приклад лінгвістичної змінної Z – рейтингу оцінок студентів:

Z=<A, T, X, M>, де

**Терм-множина:** T={низький, середній, високий};

**Множина значень:** X Є[51, 100].

**Правила відображення підмножин:**

- низький: X∈[51, 71];
- середній: X∈[72, 87];
- високий: X∈[88, 100].

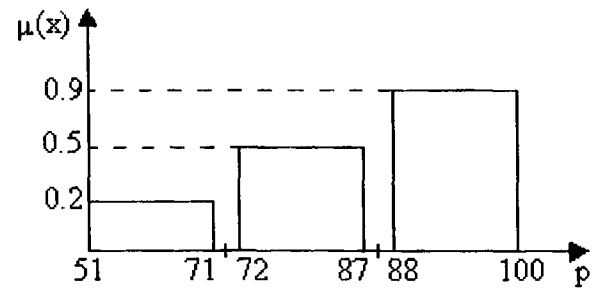



*Рис. 3.6. Графік функції належності*

Графік побудовано для таких значень нечіткої змінної

X∈[51, 71]; μ(x)=0,2;

X∈[72, 87]; μ(x)=0,4;

X∈[88, 100]; μ(x)=0,9.

Для обґрунтованого використання подана форма ЛЗ повинна відображати статистичні (або інші) апріорні оцінки значень для функції належності.

На основі наведеної ЛЗ можна побудувати такі гіпотетичні правила оцінювання шансів для безоплатного навчання в магістратурі.

ЯКЩО рейтинг низький,

ТО шанс потрапити на безоплатне навчання в магістратурі = 0.2.

ЯКЩО рейтинг середній,

ТО шанс потрапити на безоплатне навчання в магістратурі = 0.5.

ЯКЩО рейтинг високий,

ТО шанс потрапити на безоплатне навчання в магістратурі = 0.9.

**Програмування правила оцінювання рейтингу:**

- **Вхідні дані** – значення рейтингової оцінки;
- **Вихідні дані** – оцінка шансу потрапити на рівень магістра або спеціаліста з безоплатним навчанням;
- **Алгоритм.** Увести значення рейтингу. Знайти підмножину його належності в базовій множині. За значенням оцінки або її інтерполяцією до найближчого значення в підмножині отримати значення функції належності – шансу потрапляння.

**Нечіткі нейронні мережі.** Нечіткі нейронні мережі (fuzzy-neural networks) роблять висновки на основі апарата нечіткої логіки, однак параметри функцій належності набудовуються з використанням алгоритмів навчання НМ. Тому для підбору параметрів таких мереж застосовують метод зворотного поширення помилки, спочатку запропонований для навчання багатошарового персептрона. Для цього модуль нечіткого керування представляється у формі багатошарової мережі. Нечітка нейронна мережа, як правило, складається з чотирьох шарів: шару фазифікації вхідних перемінних, шару агрегування значень активації умови, шару агрегування нечітких правил і вихідного шару.

Найпоширенішими сьогодні є архітектури нечіткої НС виду ANFIS і TSK. Швидкі алгоритми навчання й інтерпретованість накопичених знань – ці фактори зробили сьогодні нечіткі нейронні мережі одним із найперспективніших і ефективних інструментів м'яких обчислень.



**Адаптивні нечіткі системи.** Класичні нечіткі системи мають той недолік, що для формулювання правил і функцій належності необхідно залучати експертів тієї або іншої предметної області, що не завжди вдається забезпечити. Адаптивні нечіткі системи (adaptive fuzzy systems) вирішують цю проблему. Підбір параметрів нечіткої системи виробляється в процесі навчання на експериментальних даних.

Алгоритми навчання адаптивних нечітких систем відносно трудомісткі і складні порівняно з алгоритмами навчання нейронних мереж і, як правило, складаються з двох стадій:

1. Генерація лінгвістичних правил;
2. Коригування функцій належності.

Перша задача належить до задачі перебірного типу, друга – до оптимізації в безперервних просторах. Завдяки цьому виникає певне протиріччя: для генерації нечітких правил необхідні функції належності, а для проведення нечіткого висновку – правила. Значна частина методів навчання нечітких систем використовує генетичні алгоритми. В англомовній літературі цьому відповідає спеціальний термін – Genetic Fuzzy Systems. Значний внесок у розвиток теорії і практики нечітких систем з еволюційною адаптацією зробила група іспанських дослідників на чолі з Ф. Херрера.

## 3.8. Мови для опису нечітких моделей

Мова нечіткого управління FCL (Fussy Control Language) розроблена для подання нечітких моделей систем управління, для програмування контролерів (Programabile Controllers) [9]. Мову введено в Стандарт IEC 1131-7.

У мові CLIPS передбачено можливість легко приєднувати за допомогою інтерфейсу код мови C або C++. Програма, написана C або C++, може викликатися повністю аналогічно до функції, заданої ключовим словом мови CLIPS [18]. Однією із

сильних сторін мови CLIPS є те, що вона розширювана, і користувач може легко вводити нові ключові слова на етапі компіляції для досягнення найбільшої ефективності. Крім того, завдяки наявності об'єктно-орієнтованих засобів мови COOL для розширення можливостей мови CLIPS можуть застосовуватися об'єкти. Інше програмне забезпечення, таке як штучні нейронні системи, генетичні алгоритми та інші програми, написані мовами C або C++, може вводитися або в ліві частини правил для ініціювання правил, або в праві частини правил для забезпечення висновку. Вихідний код мови CLIPS надається у загальне користування

Недоліками нечітких систем є:

- відсутність стандартної методики конструювання нечітких систем;
- неможливість математичного аналізу нечітких систем існуючими методами;
- застосування нечіткого підходу порівняно з імовірнісним не приводить до підвищення точності обчислень.

## Контрольні запитання

1. Назвіть причини використання нечітких методів.
2. Проаналізуйте визначення нечіткої множини.
3. Проаналізуйте визначення лінгвістичної змінної.
4. Поясніть прямі методи визначення функції належності.
5. Поясніть непрямі методи визначення функції належності.
6. Поясніть призначення коефіцієнтів визначеності.
7. Проаналізуйте зміст нечіткого відношення.
8. Назвіть операції з нечіткими множинами.
9. Проаналізуйте зміст max-min-композиції для нечітких відношень.
10. Охарактеризуйте особливості вибору функцій належності.

**Вправи**

1. Побудуйте нечітку змінну ВИХІДНИЙ ДЕНЬ на базовій множині – *дні тижня*.



2. Побудуйте нечітку змінну НАЙЯСКРАВІШИЙ КОЛІР на базовій множині – *кольори веселки*.
3. Напишіть програму для процедури нечіткого виведення з використанням нечітких множин з прикладу 1.
4. Побудуйте лінгвістичну змінну ІНТЕНСИВНІСТЬ РУХУ МАШИН У МІСТІ, використовуючи, наприклад, такі нечіткі змінні: "низька", "помірна", "висока" для базової множини – *години світлового дня*.
5. Напишіть програму для процедури нечіткого виведення з використанням нечітких множин з прикладу 2.

# Розділ 4
# МЕТОДИ І ТЕХНОЛОГІЇ ОПРАЦЮВАННЯ ЗНАНЬ В ЕКСПЕРТНИХ СИСТЕМАХ

## 4.1. Класифікація і типи експертних задач

ЕС призначені для розв'язування неформалізованих задач з нечіткими даними. Існує запропонована Ньюелом класифікація експертних задач на основі таких особливостей:

- відсутність алгоритмів розв'язання задачі;
- задача не може бути описана тільки числовими даними, необхідно застосовувати символьні дані: ознаки, характеристики, поняття;
- неможливо сформулювати точні цілі задачі, можлива їх множина;
- наявність складних зв'язків та неточності в даних задачі.

**Інтерпретація.** Задачі інтерпретації параметричних даних про природні, соціальні, технологічні процеси. Ці задачі часто потребують ЕС для роботи в реальному часі. Інформація збирається з датчиків, приладів. Вона часто є зашумленою, неповною, неточною. Для обробки використовують спеціальні методи реєстрації характеристик неперервних потоків даних (сигналів, зображень) і методи представлення.

Інтерпретуючі системи працюють з різними типами даних. Інтерпретація в області хімії використає дані дифракції рентгенівських променів, спектрального аналізу або ядерного магнітного резонансу для виведення хімічної структури речовин. Інтерпретуюча система в геології використовує каротажне зондування – вимірювання провідності гірських порід у свердловинах і навколо них, щоб визначити приповерхневі геологічні структури. У військовій справі інтерпретуючі системи, одержуючи дані від радарів, радіозв'язку й сонарних пристроїв, оцінюють ситуацію й ідентифікують військові цілі.



**Прогноз.** Експертні системи на основі прогнозу визначають імовірні наслідки заданих ситуацій (процесів). Приклади прогнозів:

- прогноз збитків врожаю від певного виду шкідників чи захворювань;
- прогноз покладів корисних копалин;
- прогноз військових конфліктів, аварійних ситуацій;
- прогноз погоди, землетрусів;
- прогноз фінансового стану в бізнес-процесах.

Системи прогнозу часто використовують імітаційні моделі. Такі моделі генерують можливі ситуації в системах для дослідження причинно-наслідкових зв'язків. Під прогнозом землетрусів розуміють таку класифікацію етапів прогнозу землетрусів: наддовгостроковий (понад 10 років), довгостроковий (від 1 року до 10 років), середньостроковий (від 1 місяця до 1 року), короткостроковий (години, дні, місяці). Створено безліч методик прогнозу (наприклад, методика числового прогнозу землетрусів, RTN-методика, розроблені, впроваджені алгоритми і програмні засоби для прогнозу землетрусів, серед них PPR-метод). Сьогодні ведуться роботи з їхнього удосконалювання і налагодження з урахуванням реальної сейсмічної обстановки і реальних спостережень.

**Діагностика.** Це задачі оцінювання імовірної причини неправильного функціонування технічних, біологічних систем. Приклади задач:

- діагноз захворювань за симптомами;
- локалізація несправностей в електронних схемах (системах);
- визначення несправних компонентів в автомобілях;
- відлагодження блоків (вузлів) технічних систем.

Цілі діагностики:

- мінімізація імовірності неточної діагностики;
- мінімізація часу діагностики і тестування;
- поєднання евристичних знань і функціонального опису систем.

Діагностичні системи взаємодіють з користувачем, щоби допомогти шукати несправності, а потім запропонувати порядок дій для їхнього усунення. В медичній галузі розроблено найбільшу кількість діагностичних систем порівняно з іншими галузями. Багато діагностичних систем розробляють для діагностики технічних і комп'ютерних систем.

**Моніторинг**. Експертні системи, що виконують моніторинг (спостереження), порівнюють реальну та очікувану поведінку системи. Прикладами можуть бути спостереження за показами вимірювальних приладів у ядерних реакторах з метою виявлення аварійних ситуацій або оцінювання даних моніторингу хворих, які знаходяться у блоках інтенсивної терапії. Експертні системи для моніторингу повинні працювати в режимі реального часу й здійснювати залежну як від часу, так і від контексту інтерпретацію поведінки спостережуваного об'єкта.

**Проектування.** Експертні системи, що призначені для проектування, можуть розробляти архітектуру об'єктів з врахуванням обмежень проекту. Під час проектування часто застосовують синтез для розроблення окремих елементів проекту та імітаційне моделювання з метою верифікації синтезованих елементів та тестування. Застосування технології штучного інтелекту не обмежується проектуванням мікросхем. Застосовуються вони і у різних CAD (computer aided design)-системах під час проектування будинків, літаків тощо, зазначаючи слабкі місця проектів та знаходячи помилки в них.

**Конфігурація**. Вибір конфігурації складних систем з багатьма частинами є комплексною задачею. Визначення того, які частини необхідні і які сумісні, вимагають докладних знань і аналізу. Це реальна проблема для експертної системи, що може гарантувати правильні і повні конфігурації. Система може також пояснити причину своєї рекомендації. Можуть аналізуватись інші аспекти задачі, наприклад: вибір устаткування, ціноутворення і вимоги клієнта. Експертна система також може імітувати та порівнювати різні сценарії.



**Планування**. Експертні системи, зайняті плануванням, проектують дії; вони визначають повну послідовність дій, перш ніж почнеться їхнє виконання. Прикладами можуть бути створення плану застосування послідовності хімічних реакцій до груп атомів з метою синтезу складних органічних сполук або створення плану повітряного бою з метою нейтралізації певного фактора боєздатності ворога.

**Навчання.** Навчальні системи містять підсистеми пояснень процесу розв'язування задач, будують модель учня і плани його навчання, можуть адаптуватись до нових задач. База знань ЕС може містити набір компетентних висловлювань (думок), бути довідником найкращих стратегій для спеціалістів. Така база знань має велику цінність, адже містить накопичений досвід поколінь спеціалістів. Тому ЕС з успіхом використовують для навчання і тренування провідних спеціалістів та інженерів.

Ще один приклад – довідкові системи. Щоб скористатися їхніми послугами, треба сформулювати питання та надіслати його електронною поштою на адресу системи. Запитати можна будь-що: від того, що їсть на сніданок королева Великобританії до того, як розрахувати орбіту Землі та штучних супутників. Система проаналізує запитання, та, використовуючи величезну базу даних про все на світі, дасть відповідь на основі інформації, яка в неї є.

**Інформаційний аналіз та пошук.** У багатьох ПО (бібліографія, юриспруденція), де переважають текстові знання, існує широкий клас задач інформаційного пошуку та аналізу у великих обсягах інформації. Нові пошукові системи, такі як NorthernLight.com та AskJarvis.com ввібрали в себе останні розробки в області штучного інтелекту та автоматичного аналізу інформації. Вони пропонують не тільки звичайний пошук за ключовими словами, а й шукають документи, які містять інформацію з потрібної тематики, аналізуючи інформацію, яка міститься в документах.

## 4.2. Проблемні аспекти розроблення баз знань

У багатьох предметних областях діяльності суспільства: в науці, техніці, соціальній області існує і постійно поповнюється клас задач, де знання накопичуються і подаються в різноманітних формах. У межах проблеми інформатизації суспільства існує нагальна потреба в автоматизації розв'язання задач подання та опрацювання знань. Саме для розв'язання цих задач є перспективними технології експертних систем. Ці технології охоплюють широке коло методів та алгоритмів опрацювання знань, проектування баз знань та експертних систем.

Бази знань, або системи баз знань – це програмні засоби, які забезпечують подання і використання знань для автоматизації розв'язування задач, коли часто необхідні непроцедурні типи знань: правила, міркування, евристики, пошукові запити. В зарубіжній науковій літературі є поширеним термін KBS-knowledge based system – системи, засновані на знаннях.

Структура бази знань (БЗ) залежить від форм представлення знань. Ці форми визначаються особливостями задач предметної області. Основні питання під час проектування БЗ :

1. Які знання будуть в ній представлені?
2. Яка форма представлення?

***Представлення знань*** – це методологія моделювання і формалізації концептуальних знань, орієнтована на комп'ютерну обробку.

***Використання знань*** – це технологія виведення розв'язків відповідно до представлення знань. Опрацювання структур потребує розроблення відповідних процедур пошуку. Тут і проявляється слабкість комп'ютера порівняно з мозком людини.

Основний "стиль" людського мозку під час опрацювання знань: прийшов, побачив, почув, відчув, запам'ятав (можливо, на все життя). Нас не хвилює, як запам'ятовуються ці знання, як вони опрацьовуються. Існує високий рівень автоматизації опрацювання знань.



Застосування експертних технологій до проектування БЗ відповідає концепції: **запити+моделі знань=алгоритми+дані.** В цій концепції основною частиною є модель знань. В ній розміщують типи запитів (уніфіковані для цієї бази і запрограмовані певною мовою – мовою запитів, або логічною мовою) та алгоритм використання (обробки) запитів. Запити можуть програмно реалізуватись: у вигляді фреймів-форм обробки та введення даних, програмних кодів мовою запитів, у вигляді макросів [2, 3]. Алгоритм обробки запитів часто називають в літературі механізмом виведення.

На рис. 4.1. подано узагальнену архітектуру бази знань.

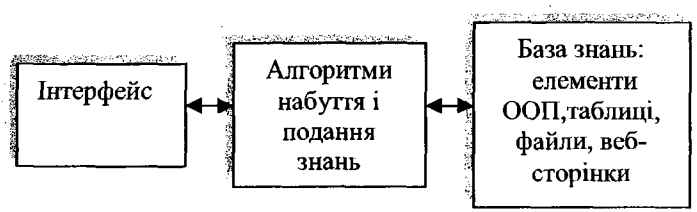



*Рис. 4.1. Основні елементи бази знань*

Аналізом сучасного стану програмних засобів розроблення баз знань виявлено дві групи проблем:

1. Методологічні проблеми;
2. Технологічні проблеми.

**Методологічні проблеми.** Основна проблема – відсутність теоретичного базису для процесу набуття і структурування знань, породжує вужчі питання на всіх етапах створення інтелектуальних систем [1]. Навіть ретельно пророблена методологія KADS характеризується громіздкістю та явною надмірністю. Нижче перераховано найзагальніші питання:

- розмитість критеріїв вибору експертної задачі;
- слабка пропрацьованість теоретичних аспектів процесів видобування знань, відсутність методів видобування знань і розбіжності в термінології;

- обмеженість моделей представлення знань, що змушує спрощувати реальні знання експертів;
- недосконаліть математичного базису моделей представлення знань (описовий, а не конструктивний характер більшості наявних математичних моделей).

**Технологічні проблеми.** Найсерйознішими з технологічних проблем є:

- відсутність концептуальної цілісності і погодженості між окремими прийомами і методами інженерії знань;
- відсутність кваліфікованих фахівців в області інженерії знань;
- явна неповнота і недостатність наявних методів структурування знань, відсутність класифікацій і рекомендацій з вибору придатного методу;
- обмежені графічні можливості програмних засобів, недостатнє врахування когнітивних і ергономічних факторів;
- складність упровадження, обумовлена психологічними проблемами персоналу і неприйняття нової технології розв'язання задач.

## 4.3. Методи набуття знань

Проблема набуття знань є багатогранною. В зв'язку з її актуальністю в 70-ті роки минулого століття з'явився термін "інженерія знань" для позначення напряму, який вивчає процеси і методи отримання, представлення і формалізації знань для розроблення ЕС та інших систем ШІ. Приклад застосування баз знань для певних класів систем наведено на рис. 4.2.

Під час розроблення баз знань ЕС найбільш уживаними є три типи джерел інформації:

1. Знання фахівців-експертів, набуті в процесі тривалої діяльності в конкретній ПО;



2. Матеріальні знання: монографії, статті, підручники, описи, довідки, розміщені в локальних інформаційних середовищах;
3. Матеріальні знання, розміщені в розподілених інформаційних середовищах-мережах, в Інтернет-мережі.

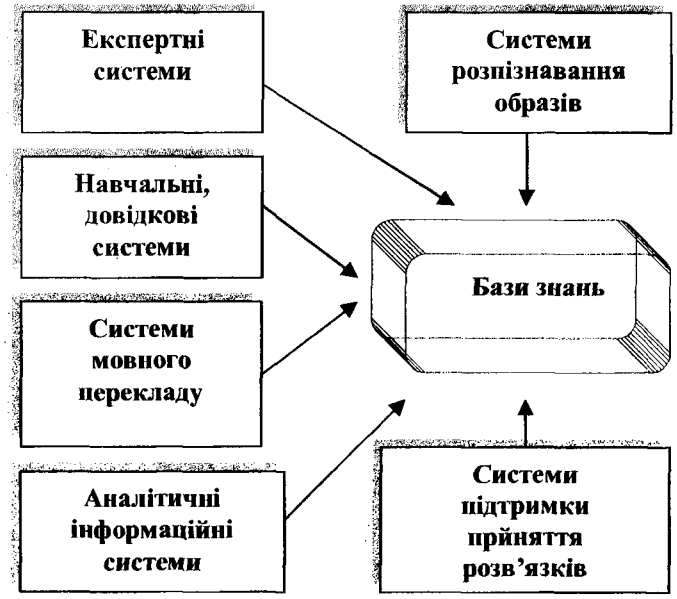



*Рис. 4.2. Застосування баз знань у типових інформаційних системах*

Набуття знань з цих джерел потребує відповідної методології і технології для автоматизації набуття і представлення знань. Найскладнішою проблемою виявилось набуття експертних знань фахівців. Розглянемо деякі аспекти цієї проблеми.

Моделлю ПО в пам'яті людини, згідно з психосемантикою, є семантична (асоціативна) модель (мережа) [1]. Така мережа є складною динамічною структурою, що має сталу (каркас) і динамічну компоненту (наповнення каркаса).

Експериментальна психосемантика як один із загальних методів моделювання знань дає змогу частково відтворювати

семантичний простір понять з предметної області в певну мережу (модель) за допомогою фактів, понять, відношень [1 ].

Важливим результатом відтворення є виділення категоріальних структур знань експертів. Для побудови опису семантичних просторів великої вимірності використовують:

- статистичні процедури;
- кластерний аналіз;
- багатовимірне шкалювання.

Із застосуванням цих процедур можна перейти до загальніших категорій (зменшити вимірність простору), перейти до вищих рівнів абстракції, згрупувати ряд окремих понять у більш місткі категорії. Це загальні уявлення про реконструкцію експертних знань.

Сучасні методології набуття знань у різних ПО передбачають такі спільні етапи:

- виділення предметної області для ЕС (для відомих – пошук аналогів);
- виділення і декомпозиція експертних задач;
- виділення об'єктів ПО як даних задачі;
- встановлення ієрархії відношень між об'єктами (структурна, причинно-наслідкова, функціональна);
- вибір моделі представлення знань.

Фундаментальний принцип набуття знань – поділ знань на поверхові та глибинні – є застосовним до багатьох ПО. Поверхові знання в ПО – це можлива ієрархія її складових, фізичних компонент. Це видима статична ієрархія. Це переважно евристики, певний досвід, аналогії.

У Європі популярною є методологія набуття знань KADS [1]. В її основу покладено поняття інтерпретаційної моделі, що дозволяє процеси збирання, структурування і формалізації знань розглядати як "інтерпретацію" лінгвістичних знань в інші представлення і структури моделі знань. Для розв'язання реальних задач використовують бібліотеку інтерпретаційних моделей, що описують



відомі експертні задачі, такі як ***діагностика, планування, моніторинг тощо.***

В основу методології KADS покладено розв'язання таких задач:

- декомпозиція проблеми (на основі зібраних знань) з широким використанням моделі гіпертексту на рівнях декомпозиції. На рівнях декомпозиції спеціалізовані (до класу задач) функції гіпертексту дають змогу структурувати знання для переведення у такі відомі структури:
  - реляційна модель із поданням ієрархії об'єктів;
  - предикатна модель із поданням ієрархії відношень-предикатів;
  - об'єктно-орієнтована модель із поданням ієрархії об'єктів;
  - функціональна модель із поданням пріоритетів обробки функцій;
- структуризація рівнів проблеми (підзадач) і специфікація у формі моделей, уніфікація вхідних даних та даних обміну між рівнями;
- синтез і структуризація зв'язків між рівнями та уніфікація даних (процесів) обміну між рівнями. Використовуються моделі : "згори донизу" або "знизу догори";
- класифікація задач на основі інтерпретаційних моделей та синтез для задач процедур пошуку рішення (зразків пошуку);
- інтеграція моделей задач та процедур пошуку розв'язків в обчислювальні процеси за допомогою процедурних правил або фреймових структур (горизонтальна інтеграція на рівнях проблеми);
- інтеграція обчислювальних процесів усіх рівнів декомпозиції (вертикальна інтеграція);
- специфікація операцій інтерфейсу для обчислювальних процесів (окремо для горизонтальної та вертикальної інтеграції);
- кодування, відлагодження та верифікація операцій.

На основі цієї методології реалізовано ряд експертних систем та систем інженерії знань (СІЗ).

## 4.4. Принципи побудови систем інженерії знань

До інструментальних засобів побудови ЕС належать мови інженерії знань та системи інженерії знань (СІЗ). СІЗ призначені для автоматизації робіт з придбання, опрацювання знань, побудови моделей та баз знань. Основні функції СІЗ:

- набуття знань і побудова моделей знань;
- верифікація моделей та підтримка баз знань;
- підтримка інтерфейсу користувача;
- реалізація алгоритмів пошуку розв'язків.

Фактично СІЗ сталим прототипом для створення інтегрованих середовищ-оболонок ЕС та систем проектування ЕС. Мови інженерії знань розвинулись на основі досвіду застосування оболонок ЕС.

Сучасні СІЗ передбачають для автоматизації набуття знань такі засоби:

- засоби підтримки різних форм представлення знань (правил, семантичних мереж, фреймів, гіпертексту, реляційних баз даних);
- різні схеми моделювання нечітких знань: за допомогою коефіцієнтів упевненості, імовірнісних, байєсових мереж, нечіткої логіки;
- різні стратегії керування (оптимізації корисності, імовірнісного проходу в мережі, аналіз правил за допомогою прямого і зворотного ланцюжків міркувань), а також програми індуктивного навчання для висновків правил прийняття рішень і управлінської інформації з прикладів [5].

Розглянемо приклади існуючих СІЗ.

**TEIRESIAS** аналізує правила, дає зауваження відносно їх повноти і несуперечливості, допомагає їх відлагоджувати.



Призначена для наповнення MYCIN правилами в галузі медицини (інфекційних захворювань). Система розпізнає терміни-ключові слова, вибирає із словника (бази даних) значення, формує (розширює) вихідне правило, редагує в процесі діалогу. Для ефективної роботи потрібні словники і бази даних.

**SALT** – система придбання знань для задач конструювання. Система SALT розроблялася в припущенні, що ця задача розв'язується методом покрокового поширення обмежень. Для розв'язання задач конструювання методом покрокового поширення обмежень необхідні знання таких типів:

- процедури встановлення значень параметрів;
- процедури перевірки обмежень;
- процедури корекції значень параметрів із зазначенням "ціни" кожної коригувальної дії.

Важливо, щоб усі ці знання становили цілісну і несуперечливу БЗ. Найскладніше для експерта послідовно, крок за кроком описати усі свої дії під час розроблення проекту. Працюючи із системою, експерт уникає цього.

SALT аналізує поточний стан БЗ і пропонує експерту-користувачу ввести чи переглянути фрагмент знань. Діалог з користувачем у SALT ведеться або за допомогою питань-підказок, або за допомогою меню. Ініціатива в діалозі належить системі.

Під час проектування баз знань доцільно мати такі засоби, які дають змогу будувати концептуальну модель за інформацією, наявною в текстових документах. Необхідну інструментальну підтримку було реалізовано в підсистемі AQUIST системи KEAT-2[1].

У цій підсистемі реалізовано ефективну технологію виділення гіпертексту з семантичною фільтрацією і побудовою карт-мереж. Розглянемо функціональні можливості AQUIST докладніше. Виділяють фрагменти за допомогою позначки (за допомогою миші) на області тексту і задання імені поняття, релевантного позначеному тексту. Декільком фрагментам може бути відповідати те саме поняття. Поняття можуть бути заздалегідь перераховані

інженером знань чи генеруватися безпосередньо в процесі аналізу тексту.

В AQUIST лексичний аналіз можна виконувати над безліччю зазначених користувачем текстів. Існує можливість задати фільтр для вилучення слів / *не представляють інтересу*, що реалізується як звичайний текстовий файл, де перераховані такі "нецікаві" слова. На поняттях можна задавати ієрархічну структуру. Користувач AQUIST може об'єднати в одну групу близькі поняття і назвати цю групу.

Групи понять, своєю чергою, можна ще більше узагальнити. Для того, щоб інженер знань отримав цілісне уявлення про ту структуру, для якої він установлює зв'язок, в AQUIST є можливість побудувати карту поточної структури. Для однієї множини понять, груп і фрагментів можна побудувати багато карт, кожна з яких відповідає деякому "погляду" на відношення між елементами. Користувач має можливість переглянути графічне представлення карти і безпосередньо маніпулювати елементами. Зокрема, можна визначати зв'язки, переміщати графічні об'єкти і підструктури тощо.

Збережені версії результатів аналізу називаються тут теоріями. AQUIST дає змогу одночасно завантажити кілька теорій і за необхідності переходити від однієї теорії до іншої. За бажанням користувача різні теорії можна об'єднати. AQUIST створено за методологією ООП: фрагменти, поняття і групи реалізовані як об'єкти.

***Систему придбання знань для медичної діагностики – ОРА*** було створено на початку 80-х років у Станфордському університеті [5]. Ця система забезпечує формування і нарощування бази знань для ЕС ONCOCIN, що дає поради з лікування онкологічних хворих. Система придбання знань ОРА заснована на детально проробленій моделі медичних знань, використовуваних лікарями-онкологами для лікування.

Система містить дев'ять типів знань:

- схема лікування (порядок і тривалість режимів лікування);



- критерії вибору протоколу;
- хіміотерапія (опис комбінацій ліків, призначуваних у тому чи іншому режимі, їхнє дозування);
- радіотерапія (локалізація і дозування радіотерапії);
- зміни в складі крові, що вимагають модифікації дозування;
- негативні реакції на лікування, виявлені шляхом лабораторних досліджень;
- інші негативні наслідки лікування, які потребують модифікації дозування ліків;
- перерва чи припинення лікування;
- лабораторні дослідження, необхідні для виявлення токсичності лікування і для збереження історії перебігу хвороби.

Ці типи медичних знань об'єднані в ієрархічну структуру. Для уведення кожного типу знань розроблено спеціальний графічний інтерфейс, що враховує те, як прийнято фіксувати відповідні знання. Так, наприклад, для запису схеми лікування онкологи використовують діаграми переходів з умовами на дугах. У системі ОРА уводять такі знання за допомогою графічної мови. Схема лікування створюється як програма цією самою мовою.

Схеми протоколів і заповнених форм транслюються системою ОРА у внутрішнє представлення БЗ ЕС ONCOCIN. Цей процес проходить без участі користувача. За схемами протоколів породжуються діаграми переходів, називані *генераторами;* за формами-бланками породжуються правила продукцій, що приєднуються до відповідного стану на діаграмі.

**ACQUIRE** – система виявлення знань для розроблення і підтримки інтелектуальних прикладних програм. Система містить методологію покрокового представлення знань, що дає змогу фахівцям у проблемній області безпосередньо брати участь у процесі придбання, структурування і кодування знання. Пряма участь фахівця в проблемній області поліпшує якість, закін-

ченість і точність придбаного знання, знижує час розроблення й експлуатаційні витрати.

Особливістю оболонки є структурований підхід до придбання знань. Модель придбання знань заснована на розпізнаванні образів. Знання представлені як об'єкти, продукційні правила і системи правил у табличній формі. Оболонка дає змогу обробляти невизначені якісні знання; містить засоби виведення і документацію баз знань у середовищі гіпертексту. Web-сторінка знаходиться за адресою http://vvv.com/ai/

Технологія **Semp-T** поєднує комплекс засобів і методів подання й обробки знань, який містить:

- високорівневі засоби завдання семантики об'єктів предметної області шляхом специфікації обмежень на значення їхніх параметрів і локальних правил висновку;
- ієрархічну семантичну мережу з обумовленими властивостями відносин;
- апарат для роботи з неточно заданими (непевними) значеннями числових, символьних і множинних типів;
- динамічні типи даних;
- розвинутий апарат продукційних правил із двома рівнями засобів динамічного керування;
- засоби генерації й перевірки гіпотез;
- об'єктну графіку й високорівневі засоби створення користувальницьких інтерфейсів;
- візуальний інтерфейс розроблювача.

**Semp-T** має універсальний характер і може використовуватися у різних сферах, де потрібен опис складних за структурою й семантикою предметних областей [35]. **Semp-T** природно поєднує логічний висновок й обчислення над неточно заданими параметрами, що забезпечує її ефективне застосування в таких галузях:

- експертні системи та їхні проблемно-орієнтовані оболонки;
- інтелектуальні бази даних і знань;



- складні діагностичні системи;
- системи планування й прийняття рішень;
- моделювання процесів у техніці, економіці, біології й соціології;
- інтелектуальні системи керування складними об'єктами, зокрема роботами;
- комп'ютерна підтримка навчальних курсів: штучний інтелект, інженерія знань тощо.

Технологія **Semp-T** орієнтована на конструктора інтелектуальних систем. Забезпечує значне підвищення якості й багаторазове скорочення трудозатрат під час створення складних систем обробки знань.

## 4.5. Мови представлення знань

NETL. Підтримує можливість створення й обробки віртуальних копій довільно великого і складного фрагмента семантичної мережі. Ці копії успадковують повні структури скопійованих описів, зокрема всі частини, елементи і внутрішні зв'язки.

OWL. Її основною властивістю є підхід до семантичних мереж, що підтримує таксономію понять з конкретизацією понять і гнучким механізмом спадкування властивостей. Усі знання містяться в єдиній, великій, уніфікованій базі знань, розширеній наборами вбудованих програм машинним кодом і мовою Ліспі пов'язаними з ними структурами даних.

RLL. Засіб дає користувачу змогу задавати свою мову представлення предметних знань, вибирати конкретний набір способів представлення, стратегії спадкування і схеми керування. У RLL також допускається прикріплення процедур і структур мови Ліспі загального виду до слотів фреймів. Засоби підтримки містять удосконалений редактор, здатний контролювати і синтаксис, і семантику.

SRL. Основні властивості мови містять: автоматичні і визначені користувачем відносини спадкування і множинні контекстні зв'язки між фреймами. Надає набір примітивів для визначення відносин та їхньої семантики спадкування, враховуючи параметри, що визначають процедуру пошуку (для її модифікації) керованого структурою спадкування. З кожним фреймом чи об'єктом-*схемою* в SRL можна зв'язати знання метарівня, тобто знання про те, як SRL використовує предметні знання. Множинні контексти забезпечують підтримку керування переглядом моделей і можливість міркувань в альтернативних світах. Діалект SRL, названий SRL/1.5.

UNIT PACKAGE. Її основна властивість полягає в організації фреймів у розділені семантичні мережі. Убудований механізм відносин узагальнення підтримує ієрархічні структури з декількома способами спадкування. UNIT PACKAGE також забезпечує механізми зіставлення зі зразком і підтримує процедури.

KRYPTON є мовою представлення знань з використанням фреймів; він також підтримує методи представлення, засновані на логіці. До її основних властивостей входять термінологічний блок, що допомагає визначити фрейми і мережі фреймів, і блок тверджень, що використовує доказ теореми шляхом резолюції для графів з непредикатними зв'язками для ведення бази даних логічних тверджень про об'єкти, визначених за допомогою фреймів. KRYPTON реалізований на INTERLISP-D.

## 4.6. Інжиніринг баз знань для цілей менеджменту на основі знань

Формування нових суспільних знань (*суспільної думки*, попиту, пропозицій) все тісніше пов'язується з використанням інформаційних технологій. Нові технології приводять до вдосконалення форм представлення знань. З'являється динамічна компонента, яка розширює вербальні та візуальні форми представлення



статичних знань. Знання стають структурованими, легко інтегрованими в інформаційні процеси для комп'ютерного опрацювання.

Ці процеси формування знань доповнюють класичні – набуття знань на основі власного досвіду, особистих контактів, з літературних та інформаційних джерел.

Розвиток інфокомунікаційних технологій створює добрі передумови розширення суспільних сфер впливу знань. Серед передумов можна назвати:

- диференціацію та структурування знань в предметних областях (ПО);
- глобалізацію поширення електронних джерел знань;
- інтенсифікацію використання в сфері науки, виробництва та послуг.

Означені передумови стимулювали появу таких нових факторів впливу знань на інформатизацію суспільства:

1. Інтеграція технологій локальної і розподіленої обробки знань;
2. Швидкість передавання знань у форми моделей знань для комп'ютерного подання (велика кількість комунікаційних каналів, бази знань, сховища даних);
3. Нагромадження і використання фахових знань у виробничих середовищах;
4. Посилення соціальної ролі знань як чинника кар'єрного росту;
5. Посилення комерційної ролі знань;
6. Зміна орієнтації навчальної тріади: знання –вміння – знання на знання – вміння – використання.

Фактори 1–3 є технологічними, фактори 4–6 – соціальними факторами впливу знань на інформатизацію суспільства [22].

**Менеджмент на основі знань** – це методологія керування процесами нагромадження і використання спеціалізованих знань за допомогою інфокомунікаційних технологій. Менеджмент знань спрямований на ефективне використання знань у різних сферах людської діяльності [1].

**Нагромадження знань** про складні системи у формі різноманітних моделей (математичні, імітаційні, ігрові, сценарні) стимулює розвиток менеджменту на основі знань. Наявність та подальший розвиток систем баз даних та знань (сховища даних та інтегровані системи) дає змогу розширити використання моделей для задач обробки даних: інформаційного аналізу, розвідки та пошуку, моделювання, сприяє розробленню нових інформаційних технологій і стандартів.

**Основна проблема сучасних технологій подання знань** – подання знань про проекти, проблеми, задачі у формі, зручній для користувача. Це знання не експерта в традиційному сенсі, а знання для менеджера. Управління знаннями необхідне для задач проектування нових систем (операцій, послуг) у нових середовищах та умовах застосування. І тут знову приходять на допомогу експертні технології та технології ШІ.

Через обмеженість людських можливостей вичерпне видобування знань на великих масивах об'єктів є неможливим без застосування спеціальних інформаційних технологій. Практично знання видобувають документуванням (формуванням і записуванням інформації про окремі поняття і проблемні ситуації). Таке документування дає змогу цілком автоматизувати (із застосуванням комп'ютерних систем пошуку в текстах) процес виявлення відомих понять і ситуацій з масивів і потоків об'єктів (процес класифікації) і перетворити неупорядковану колекцію записів на базу корпоративних знань.

## 4.7. Нові напрями і технології розроблення баз знань

Один із нових підходів до створення баз знань експертних систем одержав назву "виявлення знань у базах даних" (knowledge discovery in databases – KDD). Сьогодні цей підхід вважають найактуальнішим. Методам виявлення знань, орієнтованим на



пошук закономірностей у структурах експериментальних даних, присвячено велику кількість наукових праць, і їхня кількість постійно зростає [19].

Процес KDD передбачає кілька етапів. Це нагромадження сирих даних, добір, підготовка, перетворення даних, пошук закономірностей у даних, оцінювання, узагальнення і структурування знайдених закономірностей.

Специфіка сучасних вимог до обробки даних з метою виявлення знань є такою:

- Дані мають необмежений обсяг;
- Дані є різнорідними (кількісними, якісними, категоріальними);
- Результати повинні бути конкретні і зрозумілі;
- Інструменти для обробки "сирих" даних повинні бути простими у використанні.

Основні засоби, що задовольняють перераховані вимоги, належать до області технологій Data Mining (розкопки даних).

Система інтелектуального аналізу даних CONFOR (CONcept FORmation), що здатна працювати з великими обсягами вхідних даних в умовах загрози "інформаційного вибуху", призначена для розв'язання задач виведення закономірностей, класифікації, діагностики та прогнозування. У співробітництві з російськими та американськими вченими з використанням цієї системи було зроблено декілька тисяч високоточних прогнозів у хімії та матеріалознавстві [9].

**Агентні технології**. Перспективним сьогодні вважається застосування інтелектуальних помічників для виконання операцій пошуку й збирання інформації в Інтернеті. Нові покоління інтелектуальних автономних агентів здатні самонавчатися, ефективно взаємодіяти один з одним і проявляти певну самостійність під час спілкування із клієнтом.

Цікавий напрям в області автономних агентів народився з розвитком мови XML. На його основі сьогодні створюються XML-агенти, здатні за запитом надавати інформацію з довільних джерел даних.

Агенти можуть застосовувати будь-яку доступну мову сценаріїв (або спеціально розроблений XMLScript), а також CGI-програми – залежно від можливостей сервера, на якому вони виконуються. XML-агенти лише обмінюються повідомленнями й даними з іншими агентами за допомогою HTTP-запитів, що дає змогу створювати як завгодно складні структури автоматичного зберігання й керування даними.

Японський проект ICOT був заснований як центральна науково-дослідницька лабораторія проекту п'ятого покоління обчислювальних систем [25]. ICOT сконцентрувався на розробленні мов і апаратного забезпечення для паралельного логічного програмування.

Згідно із раннім баченням ICOT, обчислювальна система п'ятого покоління відзначається зосередженістю на вирішенні проблем і виведення; управлінні баз знань і інтелектуальних інтерфейсах. Ранні ICOT – документи передбачають трирівневу систему. В основу покладено рівень для систем баз знань, що містять апаратне забезпечення для паралельного управління базами даних і програмне забезпечення для управління базами знань. Цю систему уявляли як машину бази даних з місткістю від 100 до 1000 ГБ, здатною знайти бази знань, що вимагаються для відповіді на питання за декілька секунд.

## Контрольні запитання

1. Проаналізуйте класифікацію експертних задач.
2. Проаналізуйте зміст задачі прогнозу.
3. Поясніть зміст задачі діагностики.
4. Поясніть зміст задачі навчання.
5. Проаналізуйте зміст задачі проектування конфігурації.
6. Назвіть фактори впливу знань на інформатизацію суспільства.
7. Наведіть приклади систем інженерії знань.
8. Наведіть приклади мов інженерії знань.
9. Проаналізуйте нові підходи до набуття знань.
10. Поясніть принцип менеджменту на основі знань.
11. Проаналізуйте цілі проекту ICOT.



# Розділ 5
# МЕТОДИ І АЛГОРИТМИ ПОШУКУ РОЗВ'ЯЗКІВ ДЛЯ ЕКСПЕРТНИХ ЗАДАЧ

## 5.1. Класифікація методів і алгоритмів

Існуючі методи розв'язання експертних задач можна умовно класифікувати за видами задач так:

- методи пошуку для задач невеликої вимірності; модель задачі містить точні і повні дані;
- методи пошуку для задач великої вимірності, які допускають ієрархічну структуру за структурними чи функціональними ознаками;
- методи пошуку за наявності неточних і неповних даних в моделі;
- методи пошуку з комбінуванням моделей, призначені для роботи з областями, для адекватного опису яких недостатньо однієї моделі.

Застосування методів може бути комбінованим для розв'язання задач, складність яких зростає одночасно за декількома параметрами.

Для задач з нечіткими даними методи пошуку розрізняють за схемами урахування невизначеності та нечіткості в моделях знань:

- методи пошуку з обчисленнями на основі коефіцієнтів визначеності;
- методи пошуку з нечітким виведенням на основі нечіткої логіки;
- комбіновані методи.

Найпоширенішим для пошукових методів є локальні пошукові процедури. Вони розроблені для поширених структур даних, таких як списки, масиви, дерева, файли із спеціальною організацією знань. Багато локальних процедур мають добре

теоретичне обґрунтування, тому їх використання є закономірним. До таких процедур належать:

- цикли з логічними умовами;
- рекурсивний пошук (списки, дерева);
- пошук з поверненням – бектрекінг;
- пошук за допомогою ключових слів;
- пошук в гіпертексті.

Більшість цих процедур реалізовано в об'єктно-орієнтованих мовах програмування, мовах логічного та функційного програмування.

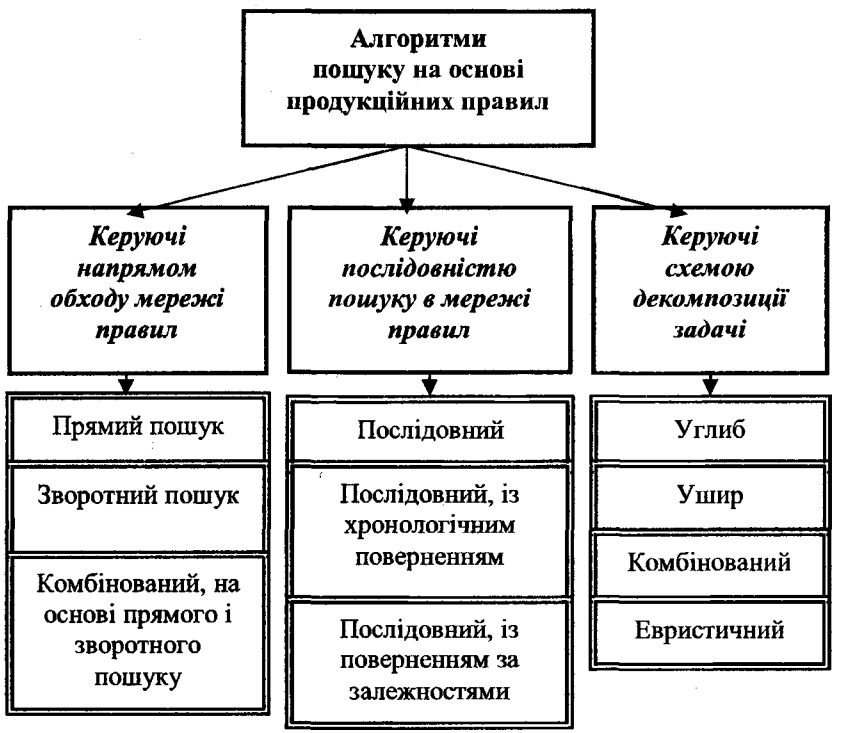



*Рис. 5.1. Класифікація алгоритмів пошуку розв'язків для експертних задач*



Для широкого класу експертних задач допустимою є декомпозиція задачі на проміжні стани-підзадачі. Це задачі діагностики, планування, проектування. Процес декомпозиції може породжувати різні форми підзадач. У разі незалежних підзадач після декомпозиції модель експертної задачі має форму дерева (дерев). Для зв'язаних – форму мережі. Для таких задач найпоширенішим є застосування моделей знань на основі продукційних правил.

В ЕС використовуються такі методи пошуку, в яких початковий стан задачі перетворюється на кінцевий за допомогою ланцюга розв'язків, що будуються на основі аналізу відповідної моделі знань. Методи розв'язування задач зводяться переважно до таких дій, як пошук за зразком, логічне виведення, пошук з вибором альтернативних шляхів та поверненням.

Загальну класифікацію алгоритмів пошуку для ЕС з моделями знань на основі правил подано на рис. 5.1. Класифікацію виконано на основі даних з [7, 11]. Подані на рис. 5.1 перша і друга група алгоритмів виконують локальний пошук у мережі правил.

Третя група призначена для загальної організації пошуку розв'язків експертної задачі.

## 5.2. Алгоритми локального пошуку в мережі правил

Продукційні системи працюють за принципом верифікації гіпотези, яка повинна підтверджуватись, уточнюватись або відкидатись сукупністю правил. Процес доведення гіпотези може породжувати шляхи проміжних висновків. В ЕС часто використовують два типи шляхів побудови висновків: прямий і зворотний. Розглянемо послідовно їх побудову.

*Прямий шлях* означає пошук від фактів до гіпотези, яку вони підтверджують. Це пошук, керований даними.

*Зворотний шлях* означає висунення гіпотези, а потім пошук підтверджувальних фактів (умов), починаючи з найближчих

рівнів і закінчуючи найнижчими. Це пошук, керований причинами або умовами виникнення події. Проілюструємо обидва пошуки з погляду можливого діалогу на прикладі моделі із трьох правил для доведення цілі (гіпотези) Z. Схему об'єднання правил в мережу для пошуку показано на рис. 5.2.

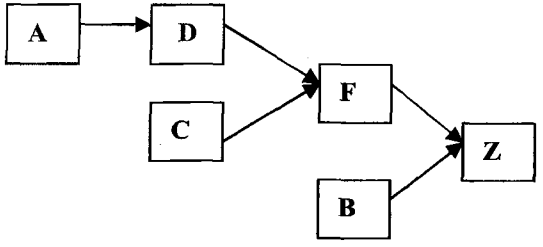



*Рис. 5.2. Мережа даних для ілюстрації пошуку*

*Прямий шлях* пошуку вибирається у такий спосіб:

- факт A реалізує подію D;
- істинність фактів (подій) D і C реалізує подію F;
- істинність подій F і B підтверджує гіпотезу Z.

*Зворотний шлях* пошуку вибирається так:

- висловлюється наявність гіпотези Z;
- початок її підтвердження крім факту B вимагає підтвердження F;
- перевіряється наявність факту C і необхідність підтвердження D, що виконується наявністю факту A.

Відмінність між обома шляхами полягає у зміні послідовності обробки гіпотез і даних. В програмній реалізації мережі правил задаються різні точки входу у відповідний алгоритм пошуку. В принципі висновок може підтверджуватись не всіма, а частиною фактів. Комбінації фактів також можуть бути різними.

***Комбінований пошук*** об'єднує зворотній пошук від вихідного стану і прямий пошук від кінцевого стану. Такий пошук застосовують для локальної мережі, яка містить правила з декількома проміжними висновками для підтвердження гіпотези.



***Області застосування***. Вибір алгоритму залежить від задачі. Якщо початкових станів (гіпотез) мало, а даних є багато, то використовують прямий пошук. Такий пошук застосовують в задачах діагностики.

Якщо є багато початкових гіпотез, то використання зворотного пошуку може бути більш ефективним. Його часто застосовують в задачах планування. Комбінований пошук може об'єднувати переваги обох методів.

Розглянуті алгоритми повинні забезпечувати виконання таких умов:

- алгоритми повинні працювати в умовах неповної інформації;
- послідовність і кількість запитів під час пошуку має бути оптимальною за критеріями швидкості одержання результату.

Однією з можливих стратегій для оптимізації запитів є стратегія одержання насамперед контекстної інформації, що підтверджує або спростовує найімовірніший на цей момент пошуку результат.

Попередні алгоритми реалізують послідовний пошук з можливим перебором всіх розв'язків у мережі правил. Ці алгоритми можуть бути вдосконалені за рахунок керування послідовністю пошуку в мережі правил. Модифікованим варіантом є метод спроб і помилок з поверненням до проміжних станів. Така модифікація називається хронологічним бектрекінгом-поверненням до останньої за часом вершини розгалуження дерева пошуку. Після повернення вибирають новий шлях.

Іншою модифікацією є метод з поверненням за залежностями. В ньому аналізується інформація: про застосовані на шляху пошуку правила; про їхні висновки. Після повернення на новому шляху відкидаються помилкові дії. Бектрекінг за залежностями може доповнюватись відсіканням альтернатив для звуження простору пошуку. Процедура бектрекінгу є реалізована у мовах програмування Пролог, LISP.

## 5.3. Алгоритми керування пошуком на основі декомпозиції задачі

Ці алгоритми належать до керуючих загальною стратегією пошуку розв'язку задачі.

Вони визначають правила обходу всіх станів декомпозиції експертної задачі. В ролі локальних процедур можуть використовуватись алгоритми першої групи. Приклади схем пошуку подано на рис. 5.3.

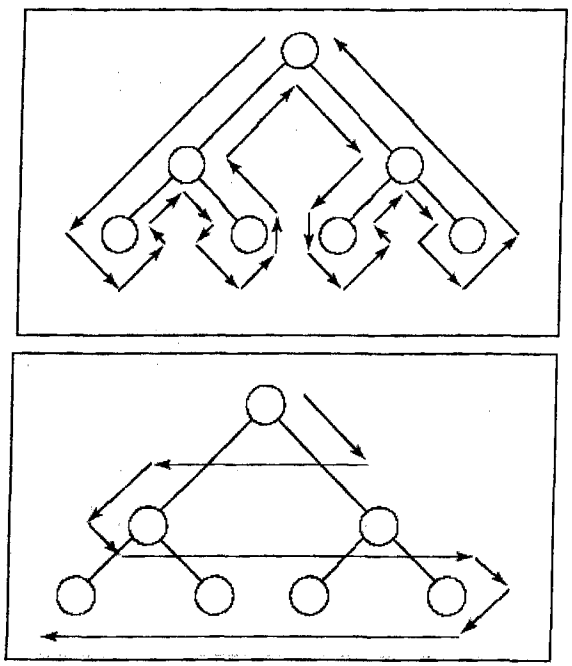



*Рис. 5.3. Схеми пошуку в мережі даних експертної задачі: а – пошук углиб; б – пошук ушир*

У реальних ЕС, що містять тисячі правил, число продукцій для підтримки алгоритму декомпозиції є відносно мале. Однак воно визначає ефективність роботи ЕС загалом.



## 5.4. Алгоритми пошуку на основі планування обчислювальних процесів

***Алгоритм пошуку на основі черги задач***. Це алгоритм процедурного типу. В алгоритмі існує нехронологічний бектрекінг, що відсікає не тільки розглядувану альтернативу, але і частину виведення або шляху в моделі **І/АБО** графу.

Основною логічною одиницею виконуваної дії в системі є задача. Задачі породжуються під час активізації фрейму або розв'язуваною в поточний момент задачею. Одночасно може бути ініційовано декілька задач. Для керування процесом використовується черга задач на основі стека. Генерація нових задач із поточної задачі супроводжується заміною її в черзі новими задачами згідно із розподілом проблеми на підзадачі.

Можливо і відкласти активізовані (генеровані) задачі. Вони стають в чергу перед активною на певний момент задачею. Алгоритм пошуку зупиняє поточну задачу і запускає додані задачі. (Це своєрідна вставка альтернатив). Розв'язані задачі видаляються з черги.

***Алгоритм керування.*** Полягає у підтримці дисципліни обробки черги задач і подаванні на обробку конкретних задач. Задачі розв'язують за відповідними правилами або процедурами, що приєднані до активного фрейму. Процедури фрейму можуть активізувати нові задачі-фрейми. Розв'язування закінчується, коли черга задач є порожньою.

***Правила обробки черги***

П1. Якщо черга порожня і починається аналіз, то активізувати початковий фрейм.

П2. Якщо черга непорожня і додано нову задачу, то активізувати її розв'язок.

П3. Якщо черга непорожня і розв'язано поточну задачу, то видалити її з черги.

П4 Якщо розв'язання задачі перерване, то ініціювати її розв'язання знову.

П5. Якщо згенеровані нові задачі поточним фреймом, то помістити їх у початок черги.

П6. Якщо черга порожня, то закінчити рахунок.

Можливі корекції правил. Правило 5; Якщо згенеровані нові задачі, то зупинити поточну, оцінити за певним критерієм нові задачі, помістити їх у чергу згідно з оцінками. Критеріальні оцінки дають змогу впорядковувати активізацію задач з черги або впорядкувати застосування правил. Отже, алгоритм пошуку на основі черги є алгоритмом, побудованим на основі правил, і належить до 2-ї групи згідно із схемою класифікації.

***Алгоритм пошуку з динамічним плануванням дій.*** Такий алгоритм реалізовано в системі **HEARASY**. Для цього алгоритму застосовується архітектура керування на основі дошки оголошень (ВСА). Це може бути таблиця, динамічний масив. Програмно дошка оголошень містить: структуроване поле (область) пам'яті і множину джерел знань (процедур). Ці джерела є декларативними відносно алгоритму пошуку. Дошку оголошень поділяють на дві частини: проблемну і керівну. Проблемна частина містить проміжні та остаточні результати розв'язання прикладних задач. У керівній частині міститься інформація про послідовність розв'язання задач та засоби їх розв'язання. На дошці виділяють п'ять областей:

**Проблеми** – описи прикладних і керівних задач, які потрібно розв'язати. З кожною задачею пов'язуються необхідні джерела знань. Серед цих задач одна є поточною на певний момент.

**Стратегії** – це плани розв'язання задач з попереднього рівня. Для розв'язання одної задачі можна використовувати різні стратегії згідно з пріоритетом та важливістю задач. Цю особливість використовує алгоритм управління.

**Фокуси** – містить локальні цілі розв'язання задачі. На основі цих цілей активізуються потрібні джерела знань. В них записують інформацію про згадані цілі.



**Поліси** – містять інформацію про планування застосування джерел знань на основі записів попереднього рівня.

**Списки** – містять два списки на основі записів фокусу.

Алгоритм з динамічним плануванням застосовується для планування дій робототехнічних систем та в системах з використанням програмних агентів.

Алгоритми пошуку розв'язків в ЕС, інтегрованих в системи проектування складних об'єктів, докорінно відрізняються від вищеописаних. Вони можуть планувати обчислювальні процедури для розв'язання локальних задач, керувати процесами обробки даних.

## Контрольні запитання

1. Проаналізуйте класифікацію методів пошуку за типами задач.
2. Проаналізуйте класифікацію методів пошуку для задач з неточними даними.
3. Поясніть прямий метод пошуку.
4. Поясніть зворотний метод пошуку.
5. Проаналізуйте локальні пошукові процедури.
6. Проаналізуйте алгоритм пошуку на основі черги задач.
7. Проаналізуйте алгоритм пошуку з динамічним плануванням дій.
8. Поясніть зміст декомпозиції експертної задачі.
9. Проаналізуйте області застосування прямого і зворотного пошуку.
10. Поясніть призначення локальних алгоритмів на основі правил.

**Вправи**

1. Розробити програму для реалізації алгоритму прямого пошуку для задачі класифікації типу транспорту.
2. Розробити програму для реалізації алгоритму зворотного пошуку для задачі класифікації типу комп'ютера.
3. Розробити програму для реалізації алгоритму на основі черги задач.
4. Розробити програму для реалізації алгоритму на основі алгоритму пошуку з динамічним плануванням подій.

# Розділ 6
# ПРОЕКТУВАННЯ ЕКСПЕРТНИХ СИСТЕМ

## 6.1. Аналіз цілей і задач проектування

Проектування ЕС має свою специфіку порівняно з проектуванням інших типів інформаційних систем. Це пов'язано насамперед із складною природою експертних задач, які важко піддаються формалізації, наявністю чітких і нечітких даних у моделі задачі, необхідністю створення баз знань, які можуть мати розподілену структуру. Для вирішення цих проблем доводиться залучати фахівців різних галузей.

У галузі медицини та інших прикладних галузях було розроблено зразки перших експертних систем, здатних моделювати процес розв'язання багатьох задач медичної діагностики, прогнозування. Це такі системи, як MYCIN, PROSPECTOR та ін. [1].

Розглянемо архітектуру та функції компонент класичної ЕС. На рис. 6.1 подано узагальнену архітектуру та функції експертної системи (ЕС). Складові підсистеми призначені для розв'язання поточних задач, пов'язаних із пошуком розв'язку експертної задачі. Їхнє призначення та функції опрацьовувались протягом тривалого часу на основі досліджень у галузі штучного інтелекту. Вони входять до складу як експериментальних зразків ЕС, так і до складу оболонок, систем навчального типу, інтегрованих засобів проектування ЕС.

**Підсистема набуття знань** розробляється на основі розглянутої вище методології для систем інженерії знань. Призначена для наповнення бази знань відповідними до задачі моделями знань.

**Підсистема пояснення** призначена для надання довідки про процес виведення пошуку розв'язків для локальних підзадач, структуру і типи правил, застосованих для операції виведення, інформації про розв'язувані задачі у певній ЕС. Використовуються компоненти графіки для показу дерев розв'язків для правил виведення, мовні



синтезатори для пояснення висновків. ПЗ повинне налаштовуватись на тип моделі знань. Користувачу надається графічний редактор правил, використовуваний для початкового введення продукцій і корекції їх у процесі налагодження, і засоби графічного трасування висновків розв'язків, що дадуть інженеру знань змогу орієнтуватися у взаємодії сотень і тисяч правил.

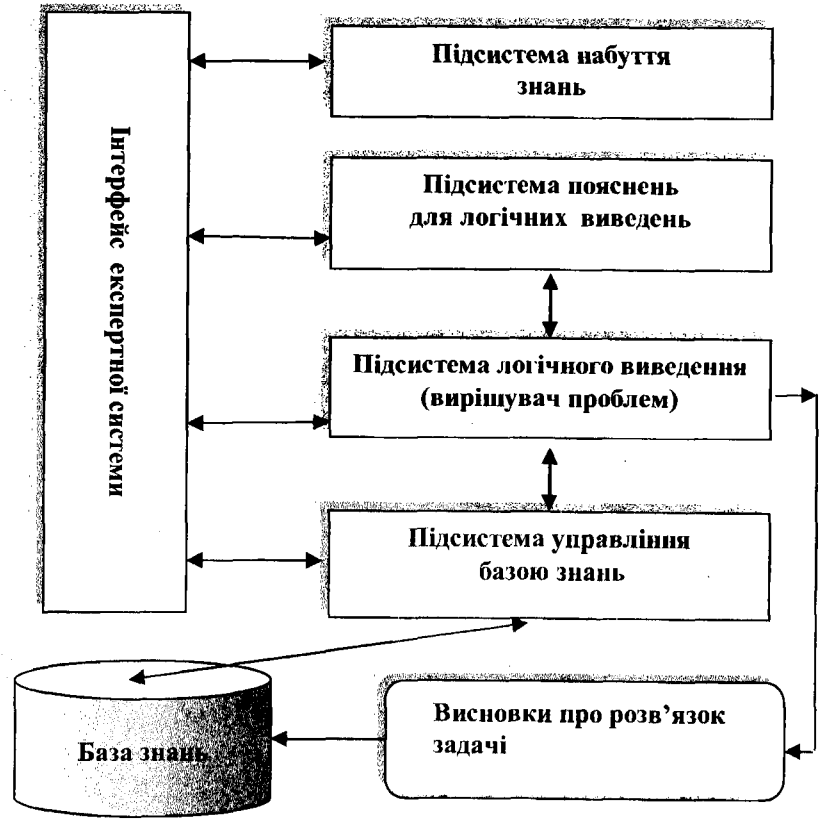



*Рис. 6.1. Загальна архітектура ЕС*

**Підсистема управління БЗ** розробляється для двох типів БЗ: компілювального та інтерпретувального типів. Для першого

типу компілятор перетворює опис моделі, виконаний за допомогою спеціальних шаблонів або мови, на внутрішні структури даних ЕС.

Для другого типу компоненти БЗ формуються як типові структури баз даних (таблиці, файли), орієнтовані на відповідну технологію доступу за допомогою запитів.

Експертна система (ЕС) – це система з базою знань, що охоплює різні аспекти процесу розв'язання експертних задач. Етапи розробки такої системи подано на рис. 6.2. Ця схема розробки є класичною для більшості систем, які використовують моделі і бази знань.

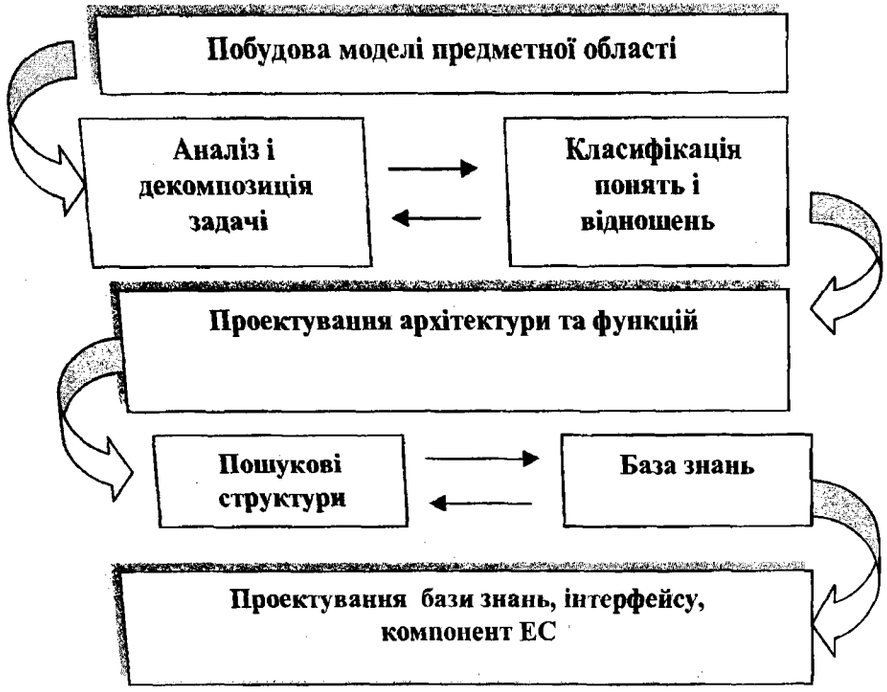



*Рис. 6.2. Етапи розроблення експертної системи*

**Проблема проектування ЕС.** Завдання проектування ПЗ для ЕС є таким: розробити структури даних моделі знань та



керівні структури-алгоритми, що визначають архітектуру ЕС, найдоцільніші для розв'язання певної прикладної задачі.

Системний аналіз об'єкта проектування – експертної системи – дає змогу побудувати дерево цілей, яке містить головну ціль та поточні цілі задачі проектування. Дерево цілей подано на рис. 6.3.

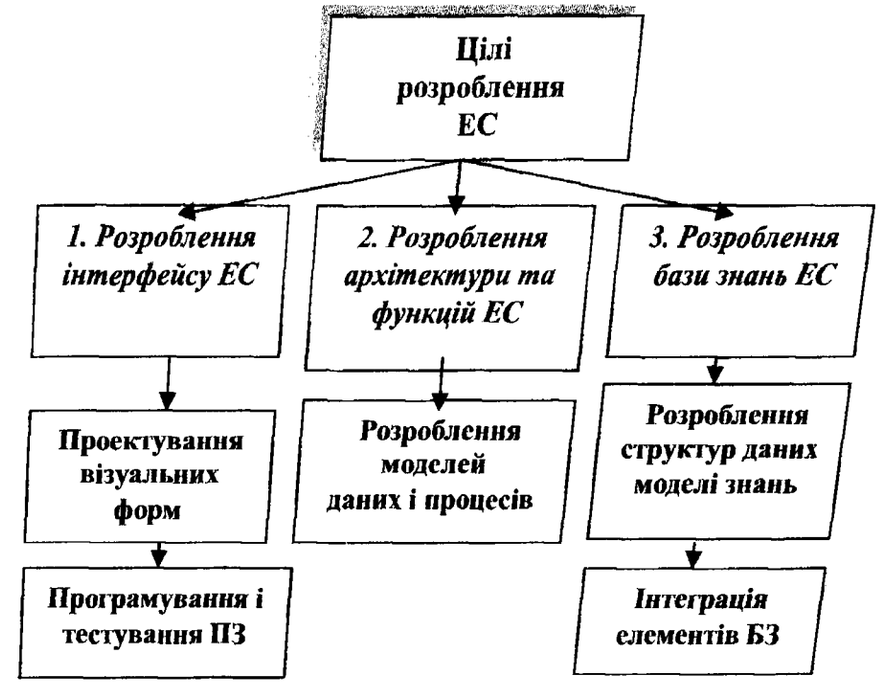



*Рис. 6.3. Дерево цілей під час проектування ЕС*

Поточні цілі містять назви локальних задач проектування ЕС та вказують послідовність їх виконання. Кожну задачу розв'язують у межах вибраної технології та середовища програмування.

Під час розроблення ЕС, як правило, використовують концепцію "швидкого прототипу" [5]. Суть цієї концепції полягає в тому, що розроблювачі не намагаються відразу побудувати кінцевий продукт. На початковому етапі вони створюють прототип ЕС.

Прототипи повинні задовольняти дві суперечливі вимоги: з одного боку, вони повинні розв'язувати типові задачі, а з іншого – час

і трудомісткість їхнього розроблення повинні бути достатньо незначні. Це потрібно для синхронізації процесу збирання і налагодження знань (здійснюваного експертом) із процесом вибору програмних засобів, здійснюваним інженером за знаннями і програмістом. Для задоволення зазначених вимог під час створення прототипу використовують різноманітні засоби, що прискорюють процес проектування.

Вирішальну роль під час практичного розроблення ЕС відіграє вибір як загалом засобів проектування ЕС, так і їхніх компонент. Залежно від призначення і від класу задач ЕС може взаємодіяти з різними типами баз даних і знань, з різними джерелами первинних сигналів. Ці питання потребують участі у розробленні ЕС фахівців як мінімум двох рівнів: експертів ПО та програмістів.

Задачі розроблення таких компонент ЕС, як бази знань і алгоритму пошуку розв'язків розглянуто в 4-му і 5-му розділах.

## 6.2. Проектування інтерфейсу

Добре спроектована інтерфейсна компонента, реалізована за допомогою спеціалізованої мови опису об'єктів і явищ предметної області, підвищує ефективність спілкування як в режимі консультації з ЕС, так і в режимі наповнення баз знань і налагодження моделей знань.

Розглянемо основні вимоги до систем подання знань і до інтерфейсу:

1. Для систематизованого керування складними знаннями великого обсягу бажаною є організація знань на основі концептуальних об'єктів.
2. З метою збільшення гнучкості системи доцільним є комбінування декларативних і процедурних знань для опису пов'язаних з ними концептуальних об'єктів.
3. Оскільки об'єкт зазвичай має ієрархічну структуру, пов'язану з деяким ступенем абстракції, то для подання знань варто застосовувати ієрархічні структури даних.



4. Під час розв'язання складних задач уважають, що різні стани виведення висновків застосовуються в комбінаціях відповідно до ситуації. Тому для подання знань необхідні функції, враховуючі цю обставину.

Створення складних систем, заснованих на знаннях, є багато в чому емпіричним процесом. Питома вага процесів налагодження й доведення системи до потрібного рівня вірогідності висновків перевищує витрати на опис і уведення моделі (принаймні в тій частині, що взаємодіє із середовищем розроблення). Налагодження складних моделей є шляхом проб і помилок, тому спосіб організації знань повинен надавати можливість вільного проектування, що полягає в застосуванні й випробуванні різних способів керування виведенням на єдиному просторі понять і розв'язків предметної області.

Визначимо вимоги до засобів проектування інтерфейсу ЕС:

1. Можливість генерації кадрів діалогу, що містять різнотипні інтерфейсні елементи.
2. Реалізація запитів на уведення текстової інформації.
3. Реалізація меню-орієнтованих запитів текстового й графічного типу.
4. Наявність засобів підтримки створення динамічних сценаріїв відображення графічної і звукової інформації з можливістю взаємної синхронізації.
5. Наявність зручних засобів компонування й редагування кадрів діалогу.
6. Наявність системи архівування і навігації у сховищі кадрів діалогу.

**Формалізація завдання проектування інтерфейсної моделі.** На кожному кроці взаємодії в користувача запитується черговий блок інформації про поточну ситуацію. Для цього надаються інформаційні структури, які, своєю чергою, мають широкий спектр форм подання. Це різні структурні форми: тексти, зображення, звукові фрагменти; з іншого боку, вони мають ще і якісну характеристику,

що визначає їхню роль і значимість у контексті розглянутого кадру діалогу. Зрозуміло, що проектувати і програмувати інтерфейс необхідно за допомогою об'єктно-орієнтованого середовища, в якому ефективно підтримуються означені вище структури даних.

Кадр складається з обумовлених ознак (які потрібно ввести) і пропонованих фактів. Ці факти можуть бути як простими фактами предметної області, означеними до певного моменту діалогу, так і повідомленнями попереджувального характеру. Це привертає увагу користувача до особливих характеристик ситуації, які необхідно або підтвердити, або швидко на них зреагувати, тому що фактори, що означили саме цю ситуацію, можуть призвести до серйозних наслідків.

Такою може бути пояснювальна, або навідна інформація. До особливого класу належать кадри, які описують факт, що є підмножиною простору розв'язків. До супровідних належать динамічні кадри, де демонструється процес, що проходить у часі. У процесі консультації, тобто такої взаємодії користувача з інтелектуальною системою, коли діалог будується за сценарієм, на певному етапі конструювання системи активним модулем стає компонента логічного виведення, яка запитує кадри для означування параметрів.

У режимі пояснення, коли користувач сам проявляє ініціативу для доступу до інформації, що визначає стан внутрішніх змінних системи логічного виведення (для користувача це ступінь розуміння інтелектуальної системи розглянутої проблеми на цьому кроці діалогу), то інтерфейсна компонента за запитами користувача видобуває з моделі шаблони, що містять фактичну інформацію з опису ситуації, наповнюючи їх відомостями з модуля логічного виведення.

Під час роботи в режимі підтримки довідкового компонента інтерфейсний модуль виконує як роль ініціатора, так і роль виконавця запитів учасника діалогу. Процес взаємодії користувача з



інтелектуальною системою має дискретний характер. Позначимо множину об'єктів, використовуваних для одержання інформації про ситуації, через **X;** об'єкти, означувані в процесі аналізу ситуації, через **F.**

Кожний етап взаємодії описується структурою даних, якими обмінюються система і користувач. Визначимо цю структуру як кадр діалогу:

$$\mathbf{Ki = \{ x, f, s \}, x \in X, f \in F, s \in S,}$$

де **S** – множина шаблонів, що описують структуру кадрів діалогу. Систему підтримки діалогу **D** можна подати парою: **D = { h, v }**, де **h ∈ H** – процедури організації уведення значень параметрів об'єктів, **v ∈ V** – процедури підтримки виведення інформаційних повідомлень.

Виведення повідомлень можна реалізувати різними способами: **M = { Vt, Vi, Va }**, де

- **Vt** – виведення текстових повідомлень-шаблонів з використанням виділень різними шрифтами й кольорами;
- **Vi** – вибір зображення, заданого як параметр або описаного в шаблоні;
- **Va** – вибір звукового повідомлення.

Своєю чергою, процедуру уведення можна організувати декількома різними способами: **N = { Hm, Hl, Hi, Ht, Hn, Hd, Hb, Hs }**, де елементи множини є:

- **Hm** – вибір значень із текстового меню, що являє собою статичний (тобто певний у шаблоні **s**) список текстових альтернатив;
- **Hl** – вибір атрибута з динамічно формованого списку значень;
- **Hi** – уведення параметрів шляхом позначення на умовному або схематичному зображенні, збереженому в шаблоні **s**, стану певних фрагментів;
- **Ht** – уведення текстового рядка;
- **Hn** – уведення числового значення із заданого діапазону із клавіатури;

- **Hd** – визначення параметра шляхом опитування датчиків;
- **Hb** – зчитування значення параметра з бази даних;
- **Hs** – одержання значення за допомогою виклику зовнішньої процедури.

Пропонована структура інтерфейсної компоненти ЕС дає змогу визначати на множині уведених об'єктів широкий набір функціоналів, які якісно й кількісно характеризують певну реалізацію моделі спілкування. Це питання пов'язане з ергономічними дослідженнями і потребує окремого висвітлення, яке виходить за межі посібника.

## 6.3. Технології проектування експертних систем реального часу

**Sachem** – французька експертна система для управління в реальному часі режимами доменних печей [29]. Sachem в 1999 році управляла шістьма з доменними печами фірми Usinor Group (2 в Сюр-Мер; 2 у Дюнкерку; і 2 у Лотарингії). На початках проекту в 1991 р. вирахована вигода була приблизно 1 євро на тонну гарячого металу. В 1999 р. повернення інвестицій на одній печі фірми Usinor Group становило приблизно 1,7 євро на тонну.

Схему процесів в системі подано на рис. 6.4. Система у режимі реального часу використовує доступну експертизу виявлення якомога швидше будь-якої аномалії, що виникає. Sachem визначає стан керованих печей, переносячи дані з давачів, обробляючи їх на місці. У кожній частині області система описує існуючі фізичні явища і поділяє їх відповідно до поточних умов експлуатаційного процесу за рівнями попереджень. Мета попереджень – убезпечити операторів від аномалій або інцидентів, які могли відбутися за короткий термін. Послідовність висновків також оцінюється за величиною й місцерозташуванням фізичних явищ.



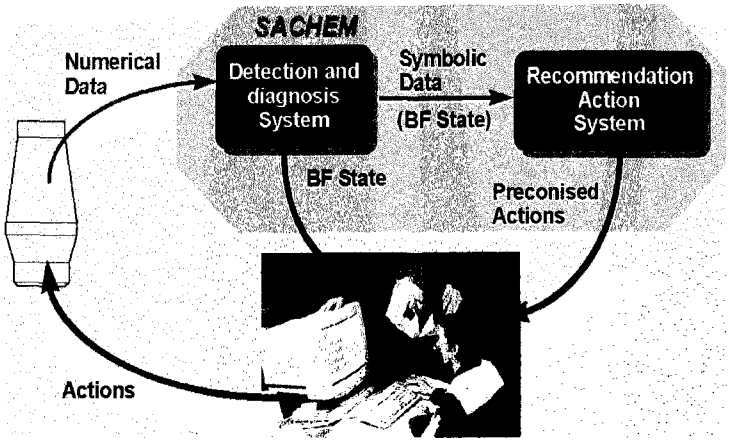



*Рис. 6.4. Схема процесів SACHEM*

Архітектуру системи налаштовано так, щоб опрацьовувати понад 11 000 пунктів даних за хвилину окремої доменної печі.

Відповідно до цієї моделі, система підтримує такі функції:

- отримання даних, синхронізація та перевірка;
- вибірка даних, обробка і підтведження (використовуючи фізичні й хімічні моделі);
- аналіз сигналу, включаючи нейронні мережі, необхідний для виявлення модифікацій (раптових змін) часових сигналів за часом і місцем;
- виявлення явищ, що зустрічаються протягом дії, інтерпретація поточної ситуації і генерація попередження за необхідності;
- рекомендація дії для оператора, коли ситуацію необхідно виправити.

Для побудови Sachem системи використано KADS-методологію. Специфікації всієї системи випливають із аналізу концептуальної моделі знання. Цю концептуальну модель побудовано в OpenKads – середовищі. Вона охоплює 25 000 об'єктів для 33 цілей, 27 завдань, 75 структур побудови висновків, 3200 понять і

2000 відношень. Складність моделі представляє 14 людино/років. Вона була реалізована командою з 6 інженерів знань й 12 експертів протягом 3-х років. Розподіл коду за модулями системи подано у табл. 6.1.

*Таблиця 6.1*

**Розміри коду для модулів системи**

| № з/п | Назва модуля | Розмір коду, % |
|---|---|---|
| 1 | Інтерфейс | 21 |
| 2 | Надбання даних, синхронізація і верифікація | 5 |
| 3 | Процес замовлення даних і підтвердження | 15 |
| 4 | Аналіз сигналу, діагностика явища, генерація попереджень | 33 |
| 5 | Повідомлення про ситуації | 3 |
| 6 | Контроль | 4 |
| 7 | Менеджер баз даних | 10 |
| 8 | Рекомендація дій | 9 |

**Концептуальна модель** складається з двох частин: загальна модель інтерпретації і лінгвістична модель, що подані природною мовою для знань. Лінгвістична модель містить 2000 сторінок тексту й графіки, розподілених у понад 70 документах.

У 1997 р. повний обсяг програмного забезпечення був приблизно 500 000 рядків коду. На рис. 6.6 показано розподіл обсягу програмного забезпечення у різних компонентах Sachem-архітектури й вплив функціональної моделі на компоненти. Близько 415 000 рядків коду було написано, щоб впровадити функціональну модель Sachem. 40 % цих рядків належать до бази знань. Цей код складається приблизно із 165 000 рядків, розподілених між 21 базою знань, що використовуються одночасно.

У 1997 р. ці бази знань містили понад 1060 класів об'єктів, 1100 перших правил логіки і 140 тимчасових ситуацій (тимчасова ситуація – набір тимчасових обмежень, що зв'язують набір подій



разом). Всі знання записані на об'єктно-орієнтованій мові у середовищі KOOL-94.

**Інструментальне середовище G2** – розробка фірми Gensym Corp – ґрунтується на відомій експертній системі реального часу PICO. G2 є однією з кращих для систем реального часу серед всіх інструментів цього класу. Система G2 реалізована на всіх основних обчислювальних платформах, враховуючи робочі станції Sun, HP9000, RS/6000. Основні функціональні можливості G2:

- підтримка процесів спостереження за безліччю (порядку тисяч) одночасно змінних параметрів і обробкою змін у режимі реального часу;
- перевірка позаштатних ситуацій на керованих об'єктах і прийняття рішень як у режимі асистування оператору, так і в автоматичному режимі.

Функціональні можливості системи забезпечуються швидким виконанням розпаралелюваних операцій, доступними в режимі on-line даними, блоками темпорального висновку (враховуючи посилання на минуле поводження і поводження керованого об'єкта в часі).

Інтеграція з підсистемами динамічного моделювання і процедурними знаннями про час, спеціальною технікою висновків для розв'язків у режимі реального часу (враховуючи стандартні forward і backward-міркування, а також event-driven висновки, сканування давачів для визначення ситуацій, що вимагають негайного втручання в процес керування, механізми фокусування на визначеній підмножині знань з використанням метазнань і потужною підсистемою real-time-trues maintenance).

Усе це дає можливість прикладним системам, розробленим з використанням G2, підтримувати на RISC-архітектурах обробку 1000 правил реального рівня складності. Зазначені параметри справлять велике враження, хоча доступної технічної інформації для системи G2 явно недостатньо для остаточних висновків про її застосовність у всьому спектрі заявлених областей.

**Система Сиріус.** Програмне забезпечення дає змогу створювати широкий спектр комп'ютерних систем для моніторингу і

керування процесами. Серед переліку є основні види систем, інтегрованих з експертними системами, розроблення яких доведено до рівня технологій:

- експертні системи підтримки оператора;
- системи контролю з експертною системою підтримки оператора;
- системи контролю/керування з експертною системою підтримки оператора;
- системи контролю/керування із засобами технічної діагностики й експертною системою підтримки оператора;
- тренажери для самонавчання операторів з експертною системою.

## 6.4. Засоби проектування експертних систем

Оболонка експертної системи PEXPERT призначена для розв'язування задач класифікації біологічних і технічних об'єктів, задач діагностики технічних об'єктів [10]. Система працює з моделлю знань на основі продукційних правил, які об'єднуються в ієрархічну мережу. Алгоритм розв'язування задачі побудований за зворотною стратегією виведення (рис. 6.5), від гіпотези до її підтвердження сукупністю фактів. Особливістю моделі знань є врахування імовірнісного характеру даних задачі та оцінювання остаточного розв'язання за допомогою коефіцієнтів упевненості. ПЗ розроблене мовою PDC-Пролог.

Як приклади використовуються три файлові бази знань: JORNEY.RBS, WEATHER.KBS і ANIMAL.KBS.

JORNEY.KBS дає змогу визначити, який вид транспорту краще використовувати для подорожі: автомобіль, поїзд чи літак.

WEATHERS.KBS намагається передбачити погоду на завтра за інформацією про погоду вчора і сьогодні. Її правила сконструйовані на основі системи правил прогнозу для погоди в Лондоні.

ANIMALS.KBS є розширеною версією ідентифікації тварин. Модель знань будується для трирівневої схеми-мережі виведення згідно з рис. 6.7.



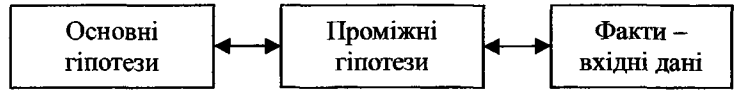



*Рис. 6.5. Елементи мережі виведення*

Систему PEXPERT побудовано у вигляді інтегрованої оболонки, яка містить віконний інтерфейс, компілятор моделі знань у внутрішнє представлення правил мовою Пролог. Діалоговий режим роботи із системою PEXPERT забезпечує такі операції:

- уведення бази знань;
- вибір з бази знань екпертної задачі;
- вибір гіпотез про стан задачі;
- уведення та редагування числових і символьних даних моделі задачі, коефіцієнтів імовірності та впевненості;
- показ правил, використаних для розв'язування задачі.

Після уведення бази знань відбувається її компіляція. Наступним кроком є вибір гіпотези, яку необхідно підтвердити за допомогою системи. Це здійснюють через вікно вибору, що виводить всі доступні гіпотези. Після уведення гіпотези починається її опрацювання. Запитання і відповіді подано на рис. 6.6.

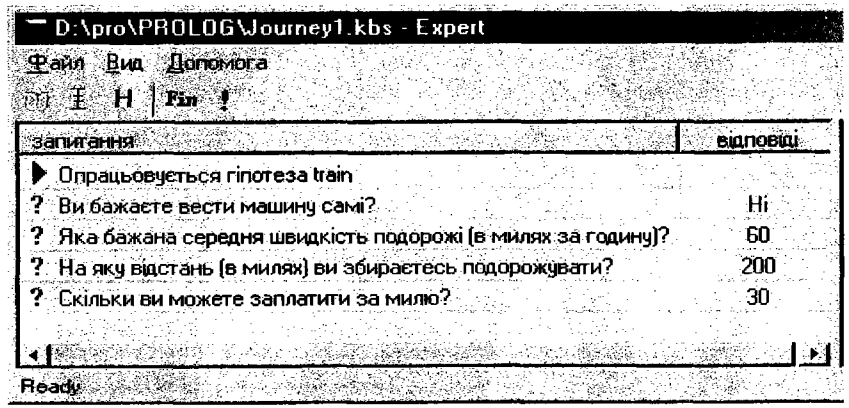
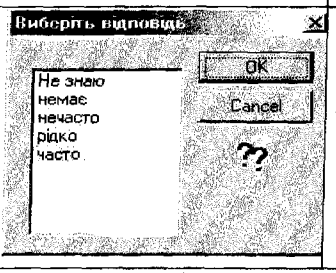
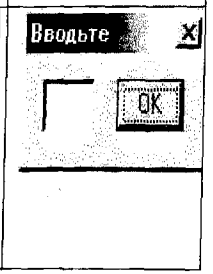



*Рис. 6.6. Варіанти запитань*

У табл. 6.2 подано можливі варіанти вибору відповідей на запитання.

*Таблиця 6.2*

**Типи відповідей і форми для їх уведення**

| № з/п | Загальні відповіді | Точніші відповіді | Числові дані |
|---|---|---|---|
| 1 |  |  |  |

На рис. 6.7 подано результати порівняння варіантів вибору.

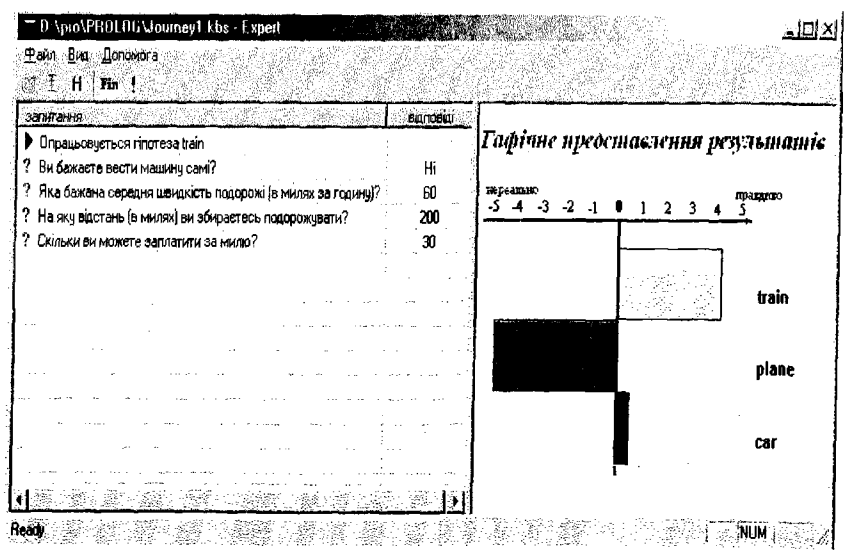



***Рис. 6.7. Графічне порівняння коефіцієнтів упевненості усіх гіпотез: поїзд, літак, автомобіль***



**XpertRule** – Архітектор Знань є середовищем для розроблення і створення розв'язків, основаних на знаннях [15]. Система складається з розробницької та виконавчої систем. Розробницька система використовується для компонування програми і збирання та обслуговування знань. Виконавча система підтримується Архітектором Знань у різних базах даних. Механізм виконання використовується для запуску програм і є доступним як EXE-програма чи Windows COM-об'єкт для вбудовування в інші програми. Механізм виконання може бути викликаний з іншого середовища розробки, що дає змогу швидко створювати прототипи та тестувати програми. Середовище розробки є графічно насиченим інтелектуальним інтерфейсом користувача із вичерпною он-лайн допомогою.

Програми, створені в Архітекторі Знань, є структуровані як Проект. Проект складається з Модулів. Кожний модуль складається з об'єктів: атрибутів, процедур, діалогів і форм звітності. **Knowledge Explorer** показує всі об'єкти проекту.

Архітектор Знань підтримує дуже широкий діапазон представлень знань, які рідко знаходяться в єдиному середовищі. Це дає можливість розробникам підтримувати набір прикладних програм знань. Архітектор Знань містить стандартні графічні блоки для подання знань. Сумісність представлення знання гарантує легкість технічного обслуговування і здатність розробляти гібридні прикладні програми.

Архітектор Знань представляє знання, використовуючи такі структури для візуального і діалогового опрацювання, як дерева розв'язків і таблиці варіантів.

Дерево розв'язків пов'язує результат або поточне розв'язання до ряду атрибутів-даних з опису задачі. На рис. 6.8 зображене дерево розв'язків у формі мозаїчної структури для задачі діагностики проблем у паливній системі автомобіля, які виникають під час старту автомобіля.

Дерево будують у покроковому режимі, факти для окремих вузлів дерева вибирають з таблиці розв'язків.

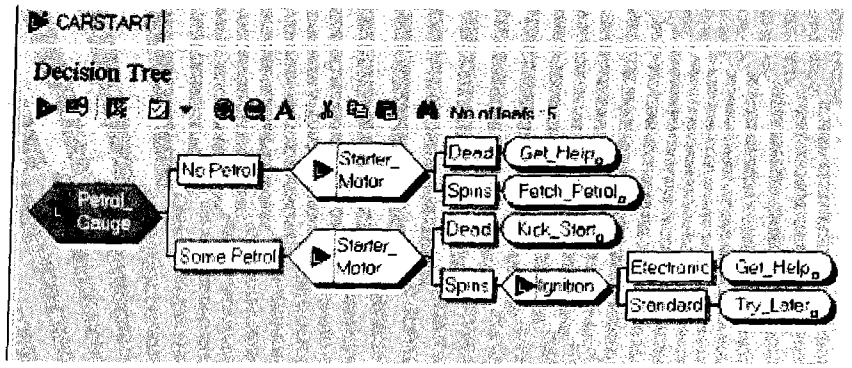



*Рис. 6.8. Мозаїчне дерево розв'язків для задачі діагностики*

Таблиця варіантів містить список прикладів або правил, кожний з яких показує, як результат або розв'язок стосуються комбінації значень атрибутів. Дані табл. 6.3 можна використовувати для діагностики проблем, які виникають під час старту автомобіля.

*Таблиця 6.3*

**Варіанти розв'язання задачі діагностики**

CARSTART

Cases

| | Starter_Motor | Engine_Fires | Petrol_Gauge | Ignition | CARSTART |
|---|---|---|---|---|---|
| 1 | Dead | * | No Petrol | * | Get_Help |
| 2 | Spins | No | Some Petrol | Electronic | Get_Help |
| 3 | Spins | No | No Petrol | * | Fetch_Petrol |
| 4 | Spins | Yes | No Petrol | Electronic | Fetch_Petrol |
| 5 | Spins | Yes | Some Petrol | Electronic | Get_Help |
| 6 | Dead | * | Some Petrol | * | Kick_Start |
| 7 | Spins | No | Some Petrol | Standerd | Try_Later |



Атрибут у дереві розв'язків або в таблиці варіантів може самостійно бути представлений іншим деревом розв'язання або таблицею варіантів. Це називається "формуванням ланцюжка" знань. Такі ланцюжки утворюють мережу правил для розв'язання задачі діагностики.

Коли знання набуті і протестовані, вони можуть бути впроваджені на великій кількості платформ і конфігурацій.

Механізм інтерфейсу використовується для запуску програми на клієнтському комп'ютері. Він є доступним як виконавчий exe-файл. Типова конфігурація використовується для запуску програми на окремому ПК чи мережевій машині.

COM+ інтерфейс використовується для запуску програми під Windows NT / 2000 server. Використовується XML для обміну даними з іншими програмами (отримання даних і повернення розв'язків). COM+ інтерфейс може бути використаний як вбудований сервер знань для інших програм.

Знання, набуті Архітектором Знань, можуть бути згенеровані як Java клас у вигляді аплету. Це дає змогу знанням бути вбудованими в HTML-сторінки для виконання клієнтом без інтерфейсу сервера. Згенерований Java-аплет є малим (приблизно 100 KB) і може використовуватися навіть на Інтернет-каналах з низькою пропускною здатністю. Це є ідеальна конфігурація для додавання інтелекту до веб-сторінки без вимоги наявності сервера, надаючи можливість запуску аплету у вікні браузера.

Основні характеристики середовища **XpertRule:**

- інтеграція з XML-даними;
- використання шаблонів для об'єктів, налаштовування іконок, імен і груп об'єктів, враховуючи термінологію користувача;
- надає розробникам контроль прав доступу і "ролі користувачів" для всіх об'єктів і методів підтримки знань;
- зберігає розроблену програму в ODBC-базі даних для

стандартного системного управління (перевірки-вводу/виводу, резервування і відкат версій);

- всі об'єкти словника і властивості налаштовуються для розробника, навіть зміна кольору кнопки діалогу під час виконання програми;
- додавання спеціальних властивостей до об'єктів. Наприклад, додавання властивостей до кожного значення атрибуту списку, що дасть можливість поширювати ці властивості на дерева варіантів, діалоги користувача і процедурні команди;
- відображення словникових об'єктів для зовнішніх баз даних. Це необхідно для використання великих баз списків значень атрибутів і коли потрібно мати динамічні значення під час виконання програми;
- уведення з таблиць варіантів, які можуть бути також відображені в зовнішніх базах даних;
- наскрізна drag- and drop-технологія редагування, з багатьма вікнами і вкладками під повним контролем розробника;
- помічник розробника – набір складних властивостей знань із покроковою допомогою.

**KEAT-Knowledge Engineers Assistant**. Ця система розроблялась у межах проекту VITAL згідно з європейською науковою програмою ESPRIT (1990–2000). Система підтримує не тільки окремі методи моделювання, але і забезпечує інтегровану програмну підтримку взаємозалежних моделей знань. Основні компоненти KEAT:

- **ACQUIST** – засіб фрагментації текстових джерел знань, що дає змогу розбити текст чи протокол бесіди з експертом на безліч взаємозалежних, анотованих фрагментів (гіпертекст) і утворити концепти (поняття);
- **FLIK** – фреймово-орієнтована мова представлення знань;



- **GIS** – графічний інтерфейс, використовуваний як для створення гіпертекстів і концептуальних моделей за допомогою ACQUIST, так і для проектування фреймових систем на основі мови FLIK;
- **ERI** – базисний інтерпретатор правил, що забезпечує прямий і зворотний висновки для продукційних правил;
- **TRI** – візуалізуючий інтерпретатор правил, що демонструє трасу виконання продукцій у формі мозаїчної таблиці, а також графічно показує активні правила, фрейми, конфліктні множини;
- **Tables** – інтерфейс маніпулювання таблицями, що може використовуватися разом із усіма моделями знань, підтримуваними в KEAT;
- **CS** – мова опису і поширення обмежень;
- **TMS** – немонотонна система супроводу істинності, тісно пов'язана з ERI, FLIK і CЗ;
- **TMV** – графічний інтерфейс підсистеми TMS, що забезпечує візуалізацію на І/АБО дереві значення істинності чи хибності висновків залежно від посилань.

У системі KEAT концептуальні моделі можуть створюватися за допомогою методів "донизу" і "догори". Перший підхід використовується за чітко визначеної задачі і наявності концептуальної моделі в ПО, наприклад у задачах діагностики. Таку модель можна відразу відобразити у вигляді таблиць і концептуальних моделей і ввести в майбутню ЕС за допомогою GIS і Tables.

Коли придбання знань побудоване не на чітко визначеній моделі ПО, наприклад, на протоколі опитування експерта, використовується другий підхід. Текстові дані аналізуються і фрагментуються за допомогою ACQUIST та виділяються концепти. Створена модель потім візуалізується за допомогою GIS.

Аналіз існуючих інструментальних систем показує, що спочатку в області ШІ активніше велися роботи зі створення інтелектуальних систем автоматизованого синтезу виконавчих програм [1].

І це природно, якщо вважати, що інструментарій ШІ є, власне кажучи, еволюційним розвитком систем автоматизації програмування. При цьому основна частка потужності й інтелектуальності такого інструментарію пов'язувалася не з його архітектурою, а з функціональними можливостями окремих компонентів. Великого значення надавали і зручності сполучення окремих компонент. Саме тут було отримано вражаючих результатів і саме тут найширше використовувалися останні досягнення теорії і практики програмування, – такі, як синтаксично-орієнтоване редагування, об'єктно-орієнтоване програмування та інкрементна компіляція.

## Контрольні запитання

1. Проаналізуйте призначення підсистеми пояснення в ЕС.
2. Проаналізуйте цілі проектування ЕС.
3. Поясніть основні етапи проектування ЕС.
4. Поясніть концепцію прототипу під час проектування ЕС.
5. Проаналізуйте основні вимоги до засобів проектування інтерфейсу.
6. Назвіть можливі процедури уведення для інтерфейсу ЕС.
7. Назвіть функціональні характеристики системи реального часу Gensym.
8. Проаналізуйте основні функції системи реального часу Sachem.
9. Проаналізуйте характеристики XpertRule-Архітектора Знань.
10. Поясніть функції середовища KEAT.



# Розділ 7
# ПРИКЛАДИ ПРОЕКТУВАННЯ І ПРОГРАМУВАННЯ ЕКСПЕРТНИХ СИСТЕМ

## 7.1. Розроблення консультаційної системи для задачі кредитування

У діяльності банків кредитний ризик відіграє важливу роль. Він пов'язаний із можливістю неповернення кредиту у визначені терміни або взагалі неповернення. Статистичні методи обчислення кредитного ризику потребують значних масивів даних, які зібрати доволі складно. Тому найчастіше використовують експертні методи, які ґрунтуються на судженнях фахівців банківської справи. Найпоширенішим способом оцінювання ризику під час надання кредиту є рейтингові оцінки кредитоспроможності позичальника. Оцінки ґрунтуються на рекомендаціях Національного банку України, які стосуються як фізичних, так і юридичних осіб. Стосовно фізичних осіб враховують такі фактори:

- Наявність реальної застави;
- Загальний матеріальний стан клієнта (прибутки, витрати);
- Боргові зобов'язання стосовно банківських позик.

Для програмної реалізації експертних оцінок нами вибрано приклад, поданий в [11]. Цей приклад містить 4 фактори знань про клієнта: кредитна історія, порука, борг, дохід. Основні комбінації факторів-даних зібрано у табл. 7.1.

Програмна реалізація експертних оцінок передбачає два етапи: проектування моделі знань (збирання даних, вибір структур даних, вибір алгоритмів розв'язання) і компіляцію логічних правил розв'язання.

Об'єктно-орієнтовані засоби програмування дають змогу розробляти моделі знань інтерпретувального типу.

Подамо базу знань задачі у вигляді таблиці розв'язків.

*Таблиця 7.1*

**Таблиця розв'язків**

| № з/п | Кредитна історія | | | Порука | | Борг | | Дохід | | | Ризик кредиту |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Невідома* | *Добра* | *Погана* | *Немає* | *є* | *Низький* | *Високий* | *Від 0 до 15* | *Від 15до 35* | *> 35* | |
| 1 | є | | | | | | є | | | | Високий |
| 2 | є | | | є | | є | | є | | | Високий |
| 3 | є | | | є | | є | | | є | | Середній |
| 4 | є | | | є | | є | | | | є | Низький |
| 5 | є | | | | є | є | | | | | Низький |
| 6 | | | є | є | | | | | | | Високий |
| 7 | | | є | | є | | | | | | Середній |
| 8 | | є | | є | | | є | є | | | Високий |
| 9 | | є | | є | | | є | | є | | Середній |
| 10 | | є | | є | | | є | | | є | Низький |
| 11 | | є | | | є | | є | | | | Низький |
| 12 | | є | | | | є | | | | | Низький |

У цьому прикладі існує три логічні висновки, які оцінюють ризик під час надання кредиту. Це такі висновки:

- Високий ризик;
- Середній ризик;
- Низький ризик.

Висновки одержують на основі аналізу таблиці розв'язків. У таблиці розміщено знання: умови надання кредиту і висновки про ризик під час надання кредиту. Аналіз різних комбінацій умов дає змогу побудувати логічні висновки. Їх є 12.

Первинною умовою під час формування комбінацій є кредитна історія. Така історія – це результат аналізу певних характеристик клієнта. Аналізують за відповідними методиками, які є різними для різних банків та становлять комерційну таємницю.

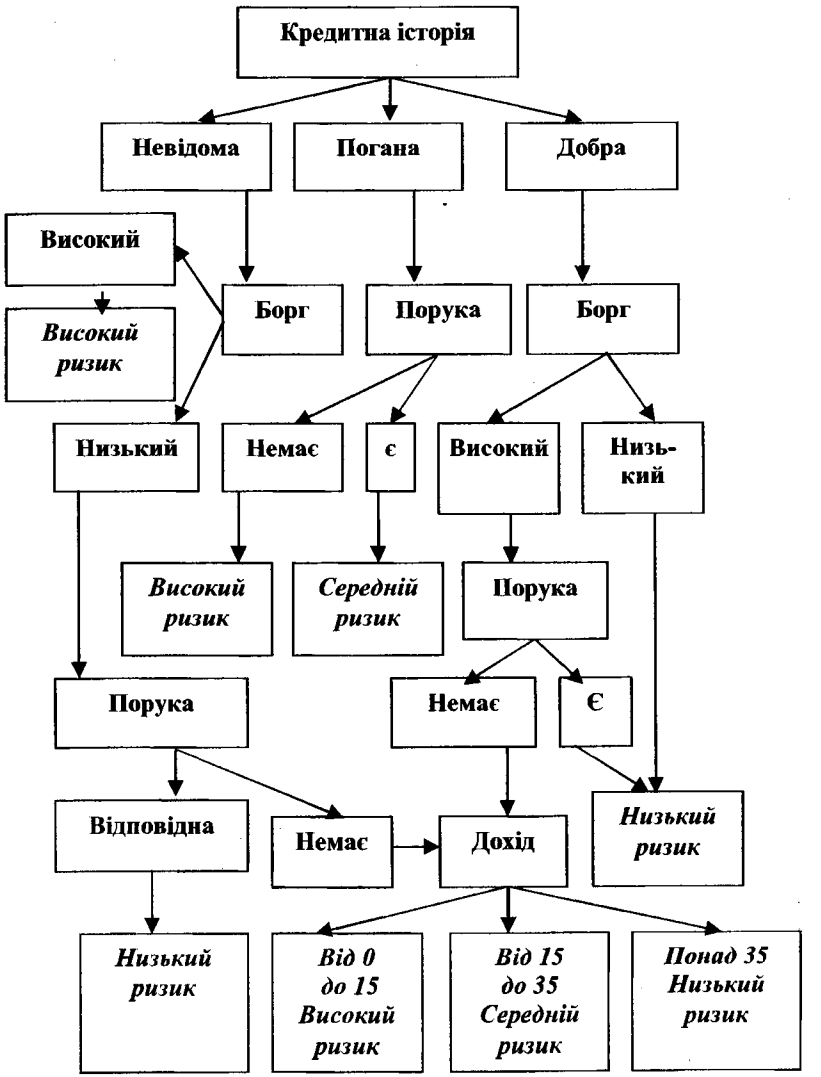



*Рис. 7.1. Дерево розв'язків для системи оцінювання ризиків під час надання кредитів*

На основі таблиці побудовано повне дерево пошуку розв'язків для задачі кредитування. Графічне зображення дерева розв'язків показано на рис. 7.1.

Для пошуку конкретного логічного висновку у дереві існує 12 шляхів. У ролі кореня дерева вибрано умову: **кредитну історію**. Для кожного шляху можна записати логічне правило отримання висновку у формі IF ( ) THEN ( ).

Кожне правило аналізує окрему комбінацію фактів, які описують кредитну ситуацію, спостережувану для конкретного позичальника кредиту. Сукупність усіх правил становить базу – модель правил пошуку розв'язків.

База правил становить інтерпретувальний алгоритм для аналізу кредитної ситуації. Цей алгоритм реалізує пряму стратегію пошуку – від фактів до гіпотези. Спочатку аналізують дані позичальника і на основі відповідного правила формують висновок про доцільність надання кредиту.

## 7.2. Програмування елементів інтерфейсу та бази знань

Програмну реалізацію бази знань виконано в середовищі VBA, поєднаному з оболонкою Excel. Структурну схему програмного забезпечення подано на рис. 7.2.

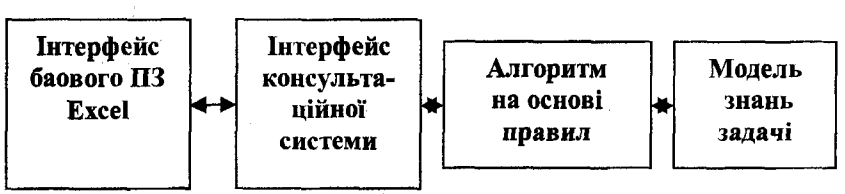



*Рис. 7.2. Структурна схема програмного забезпечення*

**Програмування деревоподібної структури даних.** Під час розроблення ПЗ підтримку деревоподібної структури розглядають як основну операцію. У середовищі програмування VBA для її



реалізації використовується елемент Treeview з набору елементів ActiveX. Цей елемент забезпечує засоби зберігання ієрархічно структурованих даних. Він ховає (або інкапсулює) деталі реалізації і дає змогу реалізувати деревоподібні структури даних за допомогою мінімальної кількості кодів.

**Принцип організації візуального відображення деревоподібної структури.** Дерево складається із вузлів (node). Верхній вузол – це кореневий вузол (root node). З кореневим вузлом пов'язують родинні вузли (root node). З родинними вузлами пов'язують дочірні вузли (parent node). Кожен рівень вузла – це сімейство. Дочірні вузли можуть утворювати нові сімейства тощо.

Елемент Treeview використовують спільно з елементом Imagelist. Цей елемент є невидимим під час виконання програми. Він містить список зображень, які можна використати для позначення вузлів дерева.

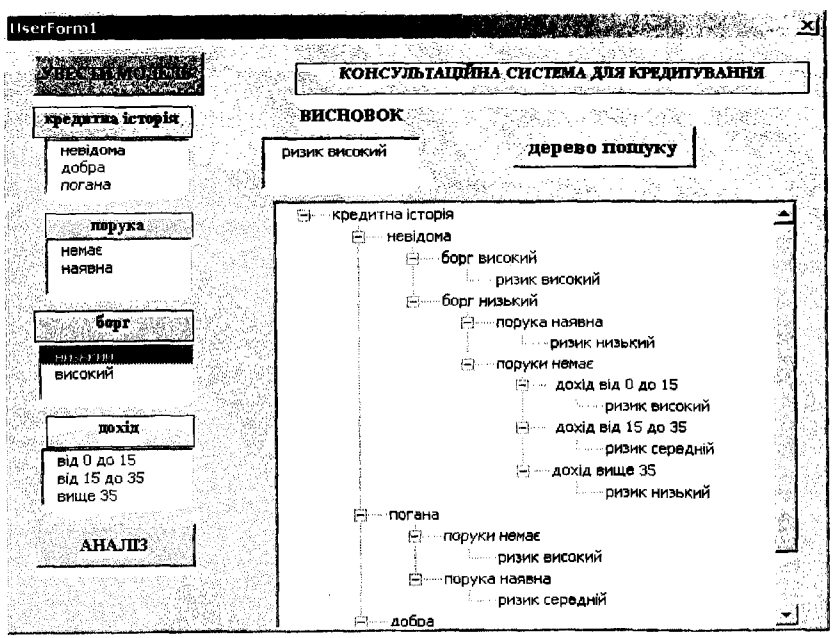



*Рис. 7.3. Вигляд головної форми консультаційної системи*

```
'Аналіз вибраної у списках комбінації умов задачі кредитування
Private Sub CommandButton1_Click()
Set ls1 = ListBox1
Set ls2 = ListBox2
Set ls3 = ListBox3
Set ls4 = ListBox4
'12 Логічних правил аналізу комбінацій умов
1 If ls1.Selected(0) And ls3.Selected(1) Then
TextBox1.Text = "РИЗИК ВИСОКИЙ,ЯКЩО кр.історія -невідома, і борг-високий"
GoTo 13
End If
2 If ls1.Selected(0) And ls2.Selected(0) And ls3.Selected(0) And ls4.Selected(0) Then
TextBox1.Text = " РИЗИК ВИСОКИЙ,ЯКЩО кр.історія -невідома,і поруки немає,і борг-низький,і дохід від 0 до 15 "
GoTo 13
End If
3 If ls1.Selected(0) And ls2.Selected(0) And ls3.Selected(0) And ls4.Selected(1) Then
TextBox1.Text = " РИЗИК СЕРЕДНІЙ,ЯКЩО кр.історія -невідома,і поруки немає,і борг-низький,і дохід від 15 до 35 "
GoTo 13
End If
4 If ls1.Selected(0) And ls2.Selected(0) And ls3.Selected(0) And ls4.Selected(2) Then
TextBox1.Text = " РИЗИК НИЗЬКИЙ,ЯКЩО кр.історія -невідома,і поруки немає,і борг-низький,і дохід вище 35 "
GoTo 13
End If
5 If ls1.Selected(0) And ls2.Selected(1) And ls3.Selected(0) Then
TextBox1.Text = " РИЗИК НИЗЬКИЙ,ЯКЩО кр.історія -невідома,і порука наявна,і борг-низький "
```



```
GoTo 13
End If
6 If ls1.Selected(2) And ls2.Selected(0) Then
TextBox1.Text = "РИЗИК ВИСОКИЙ,ЯКЩО кр.історія -погана,і поруки-немає"
GoTo 13
End If
7 If ls1.Selected(2) And ls2.Selected(1) Then
TextBox1.Text = "РИЗИК СЕРЕДНІЙ,ЯКЩО кр.історія -погана,і порука-наявна"
GoTo 13
End If
8 If ls1.Selected(1) And ls2.Selected(0) And ls3.Selected(1) And ls4.Selected(0) Then
TextBox1.Text = " РИЗИК ВИСОКИЙ,ЯКЩО кр.історія -добра,і поруки немає,і борг-високий,і дохід від0 до 15"
GoTo 13
End If
9 If ls1.Selected(1) And ls2.Selected(0) And ls3.Selected(1) And ls4.Selected(1) Then
TextBox1.Text = " РИЗИК СЕРЕДНІЙ,ЯКЩО кр.історія -добра,і поруки немає,і борг-ВИСОкий,і дохід від 15 до 35 "
GoTo 13
End If
10 If ls1.Selected(1) And ls2.Selected(0) And ls3.Selected(1) And ls4.Selected(2) Then
TextBox1.Text = " РИЗИК НИЗЬКИЙ,ЯКЩО кр.історія -добра,і поруки немає,і борг-високий,і дохід вище 35 "
GoTo 13
End If
11 If ls1.Selected(1) And ls2.Selected(1) And ls3.Selected(1) Then
TextBox1.Text = " РИЗИК НИЗЬКИЙ,ЯКЩО кр.історія -добра,і порука наявна,і борг-високий "
```

```
GoTo 13
End If
12 If ls1.Selected(1) And ls3.Selected(0) Then
TextBox1.Text = " РИЗИК НИЗЬКИЙ,ЯКЩО кр.історія -добра,і
борг-низький "
GoTo 13
End If
'Очищення списків
13 ls1.Clear
ls2.Clear
ls3.Clear
ls4.Clear
Заповнення списків
ls1.AddItem "невідома"
ls1.AddItem "добра"
ls1.AddItem "погана"
ls2.AddItem "немає"
ls2.AddItem "наявна"
ls3.AddItem "низький"
ls3.AddItem "високий"
ls4.AddItem "від 0 до 15"
ls4.AddItem "від 15 до 35"
ls4.AddItem "понад 35"
End Sub
' завантаження моделі задачі
Private Sub CommandButton2_Click()
Set ls1 = ListBox1
Set ls2 = ListBox2
Set ls3 = ListBox3
Set ls4 = ListBox4
TextBox1.Text = " "
ls1.Clear
ls2.Clear
```



```
ls3.Clear
ls4.Clear
ls1.AddItem "невідома"
ls1.AddItem "добра"
ls1.AddItem "погана"
ls2.AddItem "немає"
ls2.AddItem "наявна"
ls3.AddItem "низький"
ls3.AddItem "високий"
ls4.AddItem "від 0 до 15"
ls4.AddItem "від 15 до 35"
ls4.AddItem "понад 35"
End Sub
заповнення дерева пошуку розв'язків – TreeView1
Private Sub CommandButton3_Click()
TreeView1.Nodes.Clear
Запис кореневого вузла- кредитна історія
TreeView1.Nodes.Add , , "мет", "кредитна історія"
Запис дочірніх вузлів дерева
TreeView1.Nodes.Add "мет", tvwChild, "мет11", "невідома"
TreeView1.Nodes.Add "мет11", tvwChild, "мет111", "борг високий"
TreeView1.Nodes.Add "мет111", tvwChild, "мет112", "ризик високий"
TreeView1.Nodes.Add "мет11", tvwChild, "мет113", "борг низький"
TreeView1.Nodes.Add "мет113", tvwChild, "мет114", "порука наявна"
TreeView1.Nodes.Add "мет113", tvwChild, "мет115", "поруки немає"
TreeView1.Nodes.Add "мет115", tvwChild, "дох11", " дохід від 0 до 15"
TreeView1.Nodes.Add "дох11", tvwChild, "дох111", "ризик високий"
TreeView1.Nodes.Add "мет115", tvwChild, "дох12", " дохід від 15 до
35"
TreeView1.Nodes.Add "дох12", tvwChild, "дох112", "ризик середній"
TreeView1.Nodes.Add "мет115", tvwChild, "дох13", "дохід вище 35"
TreeView1.Nodes.Add "дох13", tvwChild, "дох113", "ризик низький"
TreeView1.Nodes.Add "мет114", tvwChild, "мет116", "ризик низький"
```

```
'аналіз поганої кредитної історії
TreeView1.Nodes.Add "мет", tvwChild, "пог1", "погана"
TreeView1.Nodes.Add "пог1", tvwChild, "пог11", "поруки немає"
TreeView1.Nodes.Add "пог11", tvwChild, "пог111", "ризик високий"
TreeView1.Nodes.Add "пог1", tvwChild, "пог12", "порука наявна"
TreeView1.Nodes.Add "пог12", tvwChild, "пог112", "ризик середній"
TreeView1.Nodes.Add "мет", tvwChild, "мет3", "добра"
'аналіз доброї кредитної історії
TreeView1.Nodes.Add "мет3", tvwChild, "доб1", "борг високий"
TreeView1.Nodes.Add "доб1", tvwChild, "доб11", "порука наявна"
TreeView1.Nodes.Add "доб11", tvwChild, "доб111", "ризик низький"
TreeView1.Nodes.Add "мет3", tvwChild, "доб2", "борг низький"
TreeView1.Nodes.Add "доб2", tvwChild, "до112", "ризик низький"
TreeView1.Nodes.Add "доб1", tvwChild, "доб3", "поруки немає"
TreeView1.Nodes.Add "доб3", tvwChild, "доб113", " дохід від 0 до 15"
TreeView1.Nodes.Add "доб113", tvwChild, "доб114", "ризик високий"
TreeView1.Nodes.Add "доб3", tvwChild, "доб115", " дохід від 15 до 35"
TreeView1.Nodes.Add "доб115", tvwChild, "доб116", "ризик середній"
TreeView1.Nodes.Add "доб3", tvwChild, "доб117", "дохід вище 35"
TreeView1.Nodes.Add "доб117", tvwChild, "доб118", "ризик низький"
'TreeView1.Nodes.Add , , "под", "порука"
'TreeView1.Nodes.Add "под", tvwChild, "под1", "немає"
'TreeView1.Nodes.Add "под", tvwChild, "под2", "наявна"
'TreeView1.Nodes.Add , , "власт", "борг"
'TreeView1.Nodes.Add "власт", tvwChild, "вл1", "високий"
'TreeView1.Nodes.Add "власт", tvwChild, "вл2", "низький"
'TreeView1.Nodes.Add , , "дох", "дохід"
'TreeView1.Nodes.Add "дох", tvwChild, "дох1", "від 0 до 15"
'TreeView1.Nodes.Add "дох", tvwChild, "дох2","від 15 до 35"
'TreeView1.Nodes.Add "дох", tvwChild, "дох3", "понад 35"
End Sub
```



## 7.3. Проектування експертної системи для задачі консалтингу під час вибору професії

Експертна система належить до класу консультаційних систем. Успіх вибору професії значною мірою залежить від вдалого збігу можливостей пошукача професії і характеристик професії та пов'язаної з нею відповідної роботи. Стосунки між пошукачем і роботодавцем – основний фактор вільного ринку. Перед пошукачем і роботодавцем стоять різні задачі. Дуже важливо знати індивідуальні особливості потреб і вимог пошукача та роботодавця. Очевидно, що всі типи вимог в експертній системі врахувати неможливо.

Розглянемо задачу вибору професій на основі моделювання психофізіологічних характеристик, описану в [9]. Модель знань містить два нечіткі відношення, які формуються на основі двох множин X і Y. Множина X описує множину спеціальностей, з яких проводиться відбір. Y описує множину психофізіологічних характеристик кандидатів. Результуюча множина Z описує множину кандидатів на навчання.

Розглядається 5 професій: x1 – менеджер; x2 – програміст; x3 – шофер; x4 – секретар-референт; x5 – перекладач.

Обрано 10 спільних характеристик для професій: y1 – швидкість та гнучкість мислення; y2 – вміння швидко приймати рішення; y3 – стійкість та концентрація уваги; y4 – зорова пам'ять; y5 – швидкість реакції; y6 – рухлива пам'ять; y7 – фізична витривалість; y8 – координація рухів; y9 – емоційно-вольова стійкість; y10 – відповідальність. Є п'ять кандидатів, які хочуть набути вищеописані професії.

Побудуємо таблиці опису нечітких відношень. Значення функції належності в нечітких бінарних відношеннях необхідно визначати за методиками експертного оцінювання за участі фахівців.

У табл. 7.2 описано нечітке бінарне відношення професії – характеристики, спільні для обраних професій.

*Таблиця 7.2*

**Бінарне нечітке відношення між професіями і характеристиками**

| | X1 | X2 | X3 | X4 | X5 |
|---|---|---|---|---|---|
| Менеджер | 0,9 | 0,9 | 0.8 | 0,4 | 0,5 |
| Програміст | 0,8 | 0,5 | 0,9 | 0,3 | 0,1 |
| Шофер | 0,3 | 0,9 | 0,6 | 0,5 | 0,9 |
| Референт | 0,5 | 0,4 | 0,5 | 0,5 | 0,2 |
| Перекладач | 0,7 | 0,8 | 0,8 | 0,2 | 0,6 |
| | X6 | X7 | X8 | X9 | X10 |
| Менеджер | 0,3 | 0,6 | 0.2 | 0,9 | 0,8 |
| Програміст | 0,2 | 0,2 | 0,2 | 0,5 | 0,5 |
| Шофер | 0,8 | 0,9 | 0,8 | 0,6 | 0,3 |
| Референт | 0,2 | 0,3 | 0,3 | 0,9 | 0,8 |
| Перекладач | 0,2 | 0,2 | 0,3 | 0,3 | 0,2 |

Нечітке бінарне відношення для можливих кандидатів на професії подано у табл. 7.3. Воно містить оцінки психофізіологічних характеристик для можливих кандидатів на обрані професії. Оцінки подано значеннями функцій належності.

Табл. 7.2 та 7.3 є вхідними даними для алгоритму max-min.

*Таблиця 7.3*

**Бінарне нечітке відношення між кандидатами і характеристиками**

| | K1 | K2 | K3 | K4 | K5 |
|---|---|---|---|---|---|
| Швидкість та гнучкість мислення – X1 | 0,9 | 0,8 | 0.7 | 0,9 | 1 |
| Вміння швидко приймати рішення – X2 | 0,6 | 0,4 | 0,8 | 0,5 | 0,6 |
| Стійкість та концентрація уваги – X3 | 0,5 | 0,2 | 0,3 | 0,5 | 0,9 |
| Зорова пам'ять – X4 | 0,5 | 0,9 | 0,5 | 0,5 | 0,2 |
| Швидкість реакції– X5 | 1 | 0,6 | 0,5 | 0,7 | 0,4 |
| Рухлива пам'ять | 0,4 | 0,5 | 1 | 0,7 | 0,8 |
| Фізична витривалість – X6 | 0,5 | 0,8 | 0,9 | 0,5 | 0,4 |
| Координація рухів – X7 | 0,5 | 0,6 | 0,7 | 0,6 | 0,5 |
| Емоційно-вольова стійкість – X8 | 0,8 | 1 | 0,2 | 0,5 | 0,6 |
| Відповідальність – X9 | 0,3 | 0,5 | 0,9 | 0,6 | 0,8 |



Застосувавши формулу композиції **max-min**, отримаємо матрицю-результат нечіткого відношення "професія –особистість", подану в табл. 7.4. На основі цієї таблиці можна подавати рекомендації щодо вибору професії.

*Таблиця 7.4*

**Нечітке бінарне відношення між кандидатами і професіями**

| | K1 | K2 | K3 | K 4 | K5 |
|---|---|---|---|---|---|
| Менеджер | 0,9 | 0,9 | 0.8 | 0,9 | 0,9 |
| Програміст | 0,8 | 0,8 | 0,7 | 0,8 | 0,8 |
| Шофер | 0,9 | 0,8 | 0,9 | 0,7 | 0,8 |
| Референт | 0,8 | 0,9 | 0,8 | 0,6 | 0,8 |
| Перекладач | 0,7 | 0,7 | 0,8 | 0,8 | 0,7 |

Подана матриця відповідає характеристикам кандидатів на здобуття певної професії. Рекомендації можна подавати за максимальними значеннями функції належності. Результати з табл. 7.4 подано в графічному вигляді на рис. 7.4

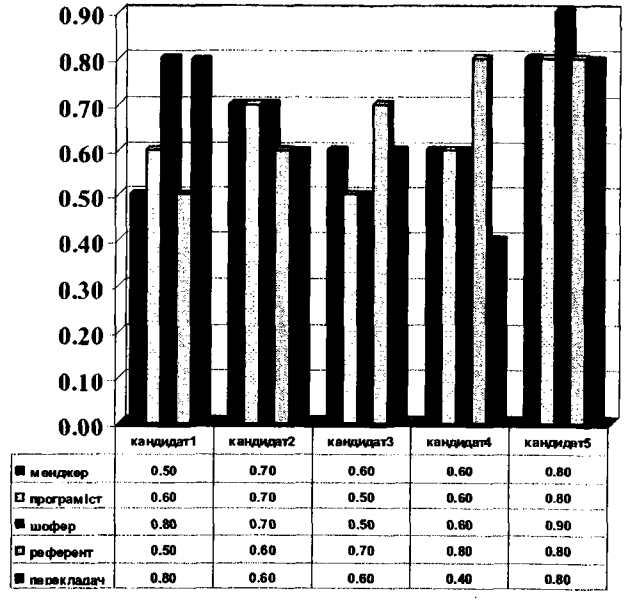



*Рис. 7.4. Графічне подання результатів*

Алгоритм розв'язання задачі оцінювання вибору професій і прийняття рішень містить таку послідовність даних і операцій:

1. Формування таблиці оцінок для професій;
2. Формування таблиці оцінок для кандидатів;
3. Обчислення нечітких оцінок для відношень професія-кандидат;
4. Побудова графіків.

Модель задачі містить матрично-векторні структури даних, які отримують на основі експертних оцінювань для характеристик професій та пошукачів. Ці структури необхідні для алгоритму обчислення нечітких оцінок на основі формул **max-min**-композиції.

Результатом роботи експертної системи є таблиця оцінок, що рекомендуються для окремого пошукача професії. Ця інформація може допомогти приймати остаточне рішення щодо рекомендації певному пошукачеві.

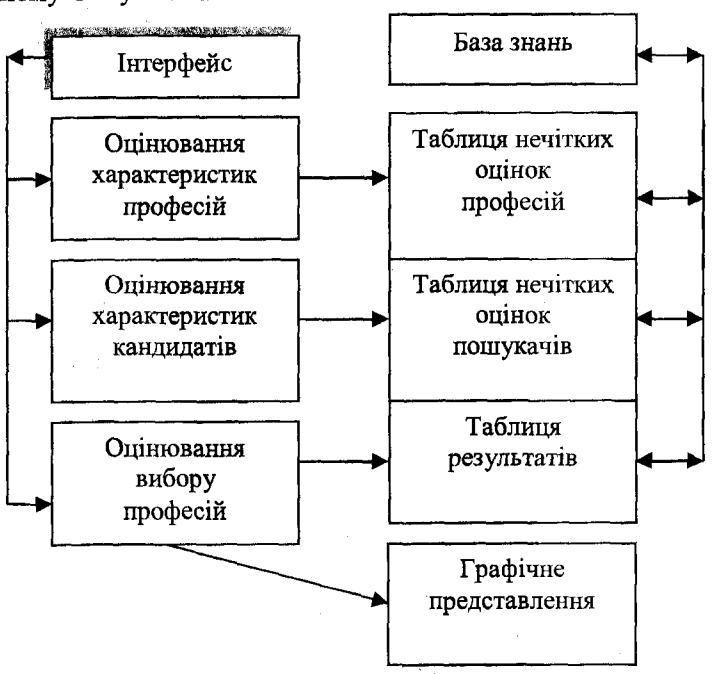



*Рис. 7.5. Архітектура та функції консультаційної експертної системи*



На рис. 7.6 подано приклад форми інтерфейсу для оцінювання характеристик професій та кандидатів. Застосування ЕС може бути корисним для кадрових агентств, під час вивчення дисциплін, пов'язаних з технологіями експертних систем.

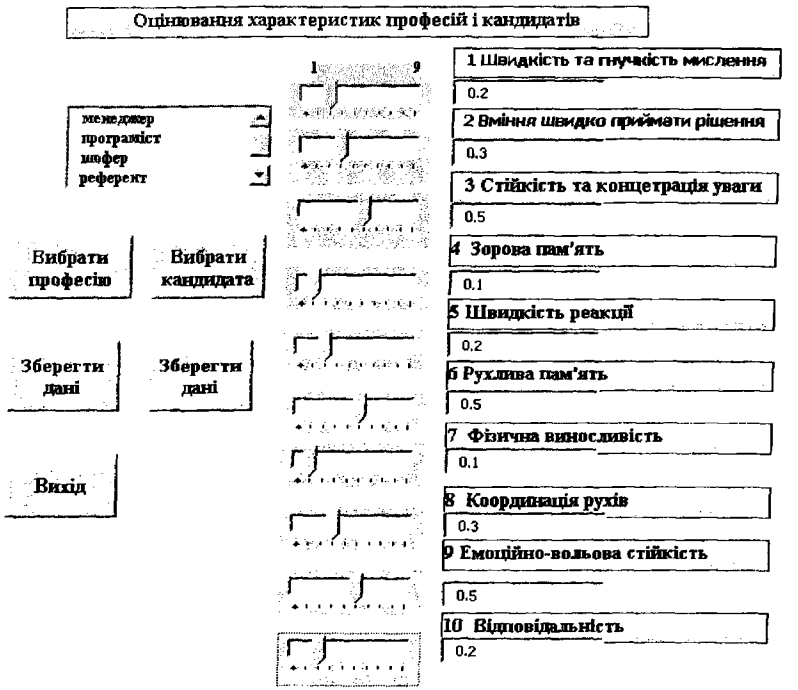



*Рис. 7.6. Форма інтерфейсу для оцінювання характеристик професій та кандидатів*

Програмне забезпечення ЕС повинне підтримувати такі функції:

- забезпечувати оцінювання характеристик професій та пошукачів;
- забезпечувати правильність обчислень за алгоритмом **max-min**;

- показувати виведені в таблицю і на графік результати оцінювання;
- забезпечувати рестарт і поновлення оцінювання.

Для розширення функціональних можливостей ЕС можна запропонувати додаткові функції:

- власний варіант набору професій та їхніх характеристик;
- власну базу знань, яка міститиме декілька різних наборів професій;
- варіант оцінювання за алгоритмом **max-prod**.

На рис. 7.6 подано приклад форми для експертного оцінювання даних, необхідних для заповнення табл. 7.2 та 7.3.

Коди програми, яка реалізує всі необхідні операції збору даних та обчислення, подано нижче.

**Програма 2**

```
Private Sub UserForm_Click()
End Sub
Private Sub CommandButton1_Click()
'підготовка до заповнення матриці
Set tb1 = TextBox1
 Set tb2 = TextBox2
Set tb3 = TextBox3
Set tb4 = TextBox4
Set tb5 = TextBox5
Set tb6 = TextBox6
Set tb7 = TextBox7
Set tb8 = TextBox8
Set tb9 = TextBox9
Set tb10 = TextBox10
'УВЕДЕННЯ СПИСКУ КАНДИДАТІВ
ListBox1.AddItem "кандидат1"
ListBox1.AddItem "кандидат2"
ListBox1.AddItem "кандидат3"
ListBox1.AddItem "кандидат4"
```



```
ListBox1.AddItem "кандидат5"
tb1.Value = 0
tb2.Value = 0
tb3.Value = 0
tb4.Value = 0
tb5.Value = 0
tb6.Value = 0
tb7.Value = 0
tb8.Value = 0
tb9.Value = 0
tb10.Value = 0
Slider1.Value = 0
Slider2.Value = 0
Slider3.Value = 0
Slider4.Value = 0
Slider5.Value = 0
Slider6.Value = 0
Slider7.Value = 0
Slider8.Value = 0
Slider9.Value = 0
Slider10.Value = 0
End Sub

Private Sub CommandButton2_Click()
'ЗАПОВНЕННЯ МАТРИЦІ КАНДИДАТ-ХАРАКТЕРИСТИКИ
Dim mas1(9) As Single
 mas1(0) = TextBox1.Value
 mas1(1) = TextBox2.Value
 mas1(2) = TextBox3.Value
 mas1(3) = TextBox4.Value
 mas1(4) = TextBox5.Value
 mas1(5) = TextBox6.Value
 mas1(6) = TextBox7.Value
 mas1(7) = TextBox8.Value
```

```
 mas1(8) = TextBox9.Value
 mas1(9) = TextBox10.Value
Set tb1 = TextBox1
Set tb2 = TextBox2
Set tb3 = TextBox3
Set tb4 = TextBox4
Set tb5 = TextBox5
Set tb6 = TextBox6
Set tb7 = TextBox7
Set tb8 = TextBox8
Set tb9 = TextBox9
Set tb10 = TextBox10
'ЗАПОВНЕННЯ МАТРИЦІ КАНДИДАТ-ХАРАКТЕРИСТИКИ
ЗА СТОВПЦЯМИ
Select Case ListBox1.ListIndex
Case 0
Worksheets(3).Range("c9:C18").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 1
Worksheets(3).Range("D9:D18").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 2
Worksheets(3).Range("E9:E18").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
```



```
Case 3
Worksheets(3).Range("F9:F18").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 4
Worksheets(3).Range("G9:G18").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
End Select
End Sub
Private Sub CommandButton3_Click()
End Sub
Private Sub Label1_Click()
End Sub
Private Sub Label10_Click()
End Sub
Private Sub Label12_Click()
End Sub
Private Sub Slider1_Click()
TextBox1.Value = Slider1.Value / 10
End Sub
Private Sub Slider10_Click()
TextBox10.Value = Slider10.Value / 10
End Sub
Private Sub Slider2_Click()
TextBox2.Value = Slider2.Value / 10
End Sub
Private Sub Slider3_Click()
TextBox3.Value = Slider3.Value / 10
```

```
End Sub
Private Sub Slider4_Click()
TextBox4.Value = Slider4.Value / 10
End Sub
Private Sub Slider5_Click()
TextBox5.Value = Slider5.Value / 10
End Sub

Private Sub Slider6_Click()
TextBox6.Value = Slider6.Value / 10
End Sub
Private Sub Slider7_Click()
TextBox7.Value = Slider7.Value / 10
End Sub
Private Sub Slider8_Click()
TextBox8.Value = Slider8.Value / 10
End Sub
Private Sub Slider9_Click()
TextBox9.Value = Slider9.Value / 10
End Sub
Private Sub CommandButton1_Click()
'ПІДГОТОВКА
Set tb1 = TextBox1
Set tb2 = TextBox2
Set tb3 = TextBox3
Set tb4 = TextBox4
Set tb5 = TextBox5
Set tb6 = TextBox6
Set tb7 = TextBox7
Set tb8 = TextBox8
Set tb9 = TextBox9
Set tb10 = TextBox10
'УВЕДЕННЯ СПИСКУ ПРОФЕСІЙ
ListBox1.AddItem "менеджер"
```



```
ListBox1.AddItem "програміст"
ListBox1.AddItem "шофер"
ListBox1.AddItem "референт"
ListBox1.AddItem "перекладач"
tb1.Value = 0
tb2.Value = 0
tb3.Value = 0
tb4.Value = 0
tb5.Value = 0
tb6.Value = 0
tb7.Value = 0
tb8.Value = 0
tb9.Value = 0
tb10.Value = 0
Slider1.Value = 0
Slider2.Value = 0
Slider3.Value = 0
Slider4.Value = 0
Slider5.Value = 0
Slider6.Value = 0
Slider7.Value = 0
Slider8.Value = 0
Slider9.Value = 0
Slider10.Value = 0
End Sub

Private Sub CommandButton2_Click()
'ЗАПОВНЕННЯ МАТРИЦІ НЕЧІТКИХ ВІДНОШЕНЬ
ПРОФЕСІЯ- ХАРАКТЕРИСТИКИ
Dim mas1(9) As Single
 mas1(0) = TextBox1.Value
 mas1(1) = TextBox2.Value
 mas1(2) = TextBox3.Value
 mas1(3) = TextBox4.Value
```

```
 mas1(4) = TextBox5.Value
 mas1(5) = TextBox6.Value
 mas1(6) = TextBox7.Value
 mas1(7) = TextBox8.Value
 mas1(8) = TextBox9.Value
 mas1(9) = TextBox10.Value
'ОПЕРАЦІЇ ЗАПОВНЕННЯ РЯДКІВ МАТРИЦІ
Select Case ListBox1.ListIndex
Case 0
Worksheets(3).Range("c3:l3").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 1
Worksheets(3).Range("c4:l4").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 2
Worksheets(3).Range("c5:l5").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 3
Worksheets(3).Range("c6:l6").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
Case 4
```



```
Worksheets(3).Range("c7:l7").Select
Set mas = Application.Selection
For i = 0 To 9
mas(i + 1) = mas1(i)
Next i
End Select
End Sub
Private Sub CommandButton3_Click()
End Sub
Private Sub Label1_Click()
End Sub
Private Sub Label10_Click()
End Sub
Private Sub Label12_Click()
End Sub
Private Sub Slider1_Click()
TextBox1.Value = Slider1.Value / 10
End Sub
Private Sub Slider10_Click()
TextBox10.Value = Slider10.Value / 10
End Sub
Private Sub Slider2_Click()
TextBox2.Value = Slider2.Value / 10
End Sub
Private Sub Slider3_Click()
TextBox3.Value = Slider3.Value / 10
End Sub
Private Sub Slider4_Click()
TextBox4.Value = Slider4.Value / 10
End Sub
Private Sub Slider5_Click()
TextBox5.Value = Slider5.Value / 10
End Sub
```

```
Private Sub Slider6_Click()
TextBox6.Value = Slider6.Value / 10
End Sub
Private Sub Slider7_Click()
TextBox7.Value = Slider7.Value / 10
End Sub
Private Sub Slider8_Click()
TextBox8.Value = Slider8.Value / 10
End Sub
Private Sub Slider9_Click()
TextBox9.Value = Slider9.Value / 10
End Sub
```

**Алгоритм композиції max-min**

```
Private Sub CommandButton1_Click()
Dim irow, icol, r1, c1, c2 As Integer
Dim mp(10) As Single
Set spree1 = Spreadsheet1
Set spree2 = Spreadsheet2
Set spree3 = Spreadsheet3
'Коди кнопки
'вибір(відмічення) масиву даних
spree1.Range("c3:l7").Select
spree2.Range("c3:g12").Select
spree3.Range("c3:g7").Select
' Set Mdata – об'ктна змінна
'Set Mdata = Application.Selection
Set mdata1 = spree1.Selection
Set mdata2 = spree2.Selection
Set Mdata3 = spree3.Selection
For r1 = 1 To 5
 For c1 = 1 To 5
 For c2 = 1 To 10
 mp(c2) = mdata2(c2, c1)
```



```
If mdata1(r1, c2) < mdata2(c2, c1) Then mp(c2) = mdata1(r1, c2)
Next c2
max1 = mp(1)
For c2 = 2 To 10
If mp(c2) > max1 Then max1 = mp(c2)
Next c2
Mdata3(r1, c1) = max1
Next c1
Next r1
End Sub
Private Sub Spreadsheet1_BeforeContextMenu(ByVal x As Long,
ByVal y As Long, ByVal Menu As OWC10.ByRef, ByVal Cancel As
OWC10.ByRef)
End Sub
Private Sub Spreadsheet2_BeforeContextMenu(ByVal x As Long,
ByVal y As Long, ByVal Menu As OWC10.ByRef, ByVal Cancel As
OWC10.ByRef)
End Sub
Private Sub Spreadsheet3_BeforeContextMenu(ByVal x As Long,
ByVal y As Long, ByVal Menu As OWC10.ByRef, ByVal Cancel As
OWC10.ByRef)
End Sub
Private Sub UserForm_Click()
End Sub
```

## Контрольні запитання

1. Поясніть зміст алгоритму **max-min.**
2. Поясніть зміст нечітких бінарних відношень.
3. Проаналізуйте зміст алгоритму **max-prod.**
4. Поясніть принципи побудови консультаційної системи.
5. Поясніть недоліки розглянутого способу оцінювання вибору професії.

## 7.4. Розроблення експертної системи для задачі медичної діагностики

Задача медичної діагностики полягає у визначенні можливих діагнозів хворого на основі знань предметної області і даних його обстеження, до яких належать значення ознак (в моменти їх огляду), значення анатомо-фізіологічних особливостей (постійні в часі) і значення подій, що відбулися (в моменти, коли вони відбувалися).

У загальному вигляді задачу медичної діагностики можна сформулювати так. Є наявні медичні знання:

- знання про огляди (ознаки, події, анатомо-фізіологічні особливості пацієнта);
- знання про захворювання, причинно-наслідкові зв'язки між захворюваннями і оглядами;
- результати оглядів хворого – анатомо-фізіологічні особливості і результати їх огляду (значення);
- ознаки, дати їх огляду і значення ознак, що спостерігалися в ці моменти;
- події, що відбулися, моменти, в які вони відбувалися, і значення подій в ці моменти.

Необхідно знайти діагноз пацієнта (множину захворювань для пацієнта), а також побудувати пояснення діагнозу – назвати причину кожного захворювання (етіологію або ускладнення) і пояснити всі спостережувані значення ознак (нормальними реакціями, реакціями на дію подій, клінічними проявами захворювань тощо [12]). Зв'язки між тим, що дано, і тим, що вимагається знайти, визначаються в моделі знань.

Така загальна постановка задачі вимагає наявності складної моделі знань, яка повинна охоплювати більшість систем людського організму. Зрозуміло, що побудова такої моделі є дуже складною задачею. Тому на практиці під час розроблення програмних засобів обмежуються моделями знань для окремих видів хвороб.



Як прикладну задачу вибрано задачу медичної діагностики для восьми типів найпоширеніших захворювань. Модель знань для задачі діагностики потребує формування таких даних:

- списку вибраних захворювань;
- списків симптомів для кожного захворювання та значень вагових коефіцієнтів для кожного симптому;
- наборів правил для підтвердження гіпотези захворювання.

Алгоритм побудови моделі знань подано на рис. 7.7.

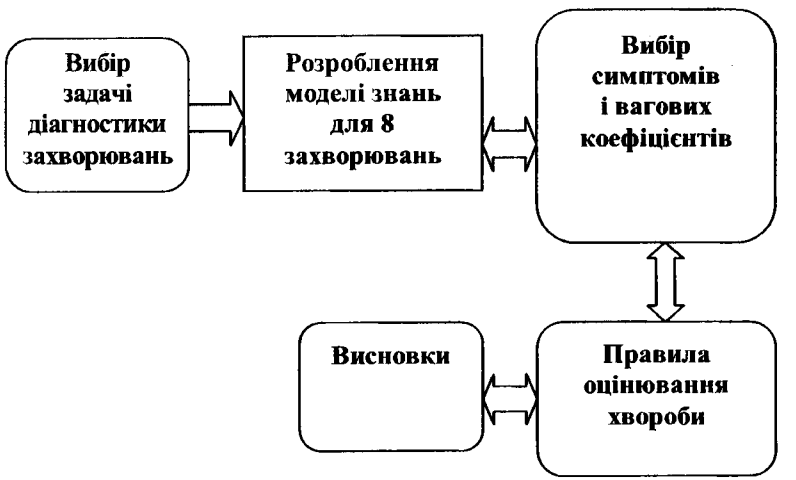



*Рис. 7.7. Схема алгоритму побудови моделі знань*

Для реалізації означених структур вибрано матричну форму – таблицю розв'язків. Її зміст для вибраної вище задачі медичної діагностики подано в табл. 7.5. Ця матриця має дві цілі: по-перше, вона визначає формат даних ; по-друге, полегшує об"єднання симптомів та їх опрацювання системою правил.

Таблиця містить назви восьми вірусних захворювань і відповідних їм симптомів та вагових коефіцієнтів.

Методику оцінювання висновків побудовано на використанні вагових коефіцієнтів для кожного симптому. Вагові коефіцієнти підсумуються тільки для підтверджених у процесі діалогу

симптомів. Сума вагових коефіцієнтів визначає впевненість у відсотках для вибраного діагнозу. Така модель оцінювання використовується в багатьох системах медичної діагностики.

*Таблиця 7.5*

**Табличне подання моделі знань для задачі медичної діагностики**

| | Хвороби → Симптоми ↓ | X1 | X2 | X3 | X4 | X5 | X6 | X7 | X8 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Температура | 20 | 20 | 20 | 20 | 15 | 30 | 20 | |
| 2 | Нежить | 10 | 10 | | | | | | |
| 3 | Кашель | 10 | 10 | 20 | | | | | |
| 4 | Біль у горлі | | | | 20 | 15 | | | |
| 5 | Збільшення підчелюсних залоз | | | | 30 | 10 | 30 | | |
| 6 | Ядуха | | | 30 | | | | | |
| 7 | Висипання | 30 | | | | 30 | | | |
| 8 | Блювота, нудота | | | | | 30 | | | |
| 9 | Опухлість вушних залоз | | | | | | 40 | | |
| 10 | Болі у животі | | | | | | | 40 | 40 |
| 11 | Розлади кишківника | | | | | | | 40 | |
| 12 | Пожовтіння шкіри | | | | | | | | 60 |
| 13 | Головний біль | | 30 | 15 | 15 | | | | |
| 14 | Озноб | | 30 | 15 | 15 | | | | |
| 15 | Світлобоязнь | 30 | | | | | | | |
| 16 | Сума вагових коефіцієнтів | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

Формати правил для діагностики захворювань:

1. **Якщо** (Температура І нежить І кашель І висипання І світлобоязнь),
   **То** є захворювання на кір.



2. **Якщо** (Температура І нежить І кашель І головний біль І озноб),
   **То** є захворювання на грип.
3. **Якщо** (Температура І ядуха І кашель І головний біль І озноб),
   **То** є запалення легенів.
4. **Якщо** (Температура І біль у горлі І збільшення підчелюсних залоз І головний біль І озноб),
   **То** є захворювання на ангіну.
5. **Якщо** (Температура І біль у горлі І збільшення підчелюсних залоз І висипання І нудота І блювота),
   **То** є захворювання на скарлатину.
6. **Якщо** (Температура І збільшення підчелюсних залоз І опухлість вушних залоз),
   **То** є захворювання на паротит.
7. **Якщо** (Температура І болі в животі І розлади кишківника),
   **То** є захворювання на дизентерію.
8. **Якщо** (болі в животі І пожовтіння шкіри),
   **То** є захворювання на гепатит.

До цієї моделі знань у формі продукційних правил буде застосовано зворотний алгоритм виведення, опис якого подано в розділі 5.

На рис. 7.8. подано архітектуру та функції програмного забезпечення експертної системи для задачі медичної діагностики.

В архітектурі ПЗ реалізовано основний принцип – діалоговий режим роботи користувача.

Програмну реалізацію експертної системи для діагностики медичних захворювань виконано в об'єктно-орієнтованому середовищі VBA. Інтерфейс будується на основі візуальних компонент. Основою для їх розміщення є форми. Тому проектування інтерфейсу ґрунтується на моделюванні форм. Вигляд форм подано на рис. 7.9, 7.10. Головна форма викликається за допомогою макросу.

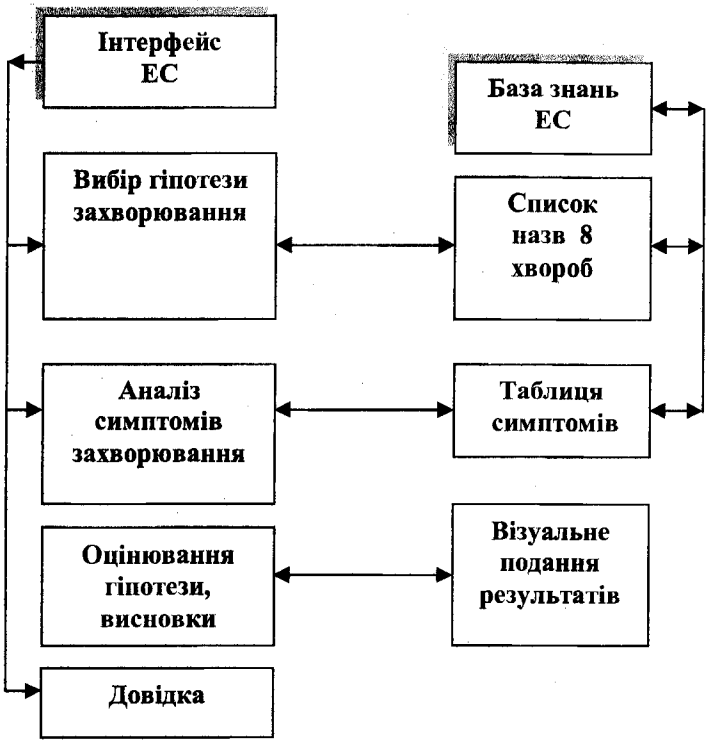



*Рис. 7.8. Архітектура та функції програмного забезпечення експертної системи*

За допомогою елементів управління даними вибирається можлива гіпотеза захворювання з 2-х груп хвороб: вірусних та інфекційних. Вибір гіпотези активізує алгоритм зворотного пошуку розв'язку задачі медичної діагностики.

Ця форма реалізує такі функції:

1. Уведення з бази знань для вибраної гіпотези хвороби її симптомів;
2. Підтримка вибору необхідних симптомів;
3. Підтвердження і візуалізація результатів аналізу гіпотези;
4. Повернення до першої форми інтерфейсу;



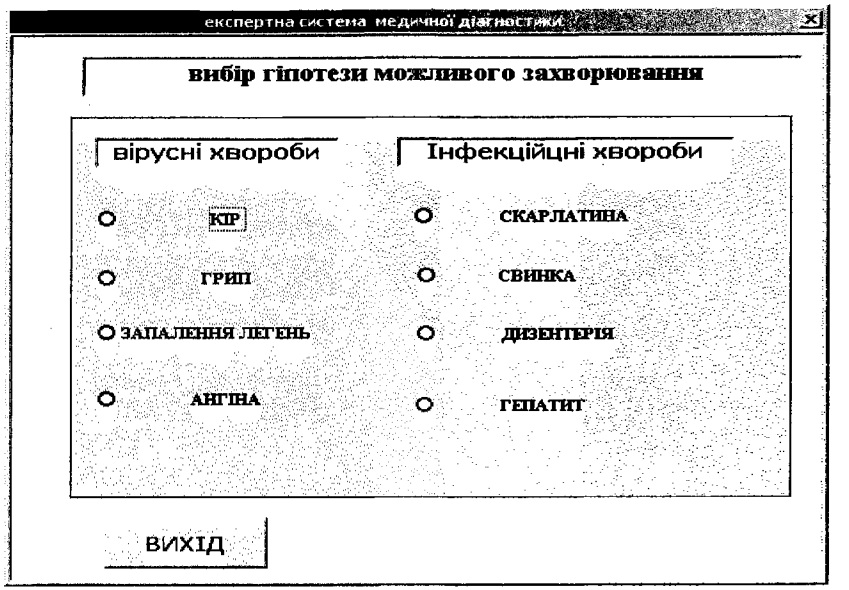



*Рис. 7.9. Головна форма інтерфейсу ЕС*

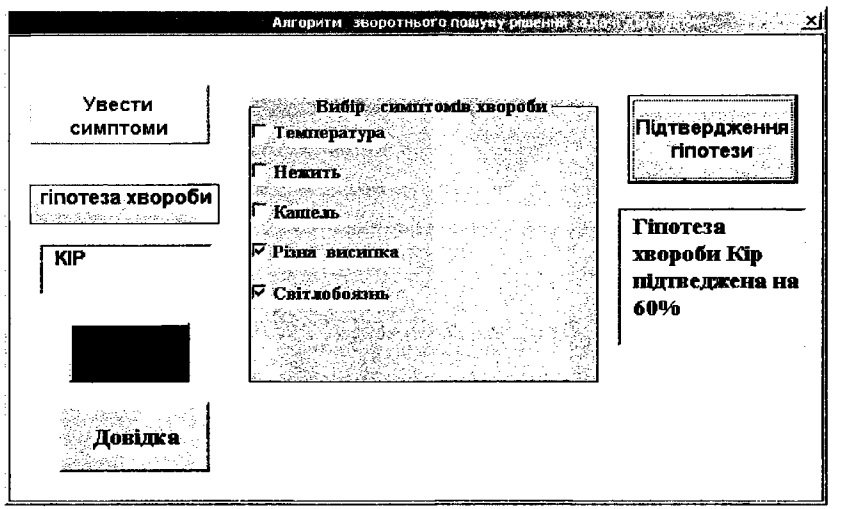



*Рис. 7.10. Форма інтерфейсу для аналізу симптомів і оцінювання гіпотези захворювання*

5. Виклик довідки про описи захворювань та можливе лікування.

Програмне забезпечення експертної системи може бути використано в навчальних цілях для лабораторних робіт з дисципліни "Системи штучного інтелекту" та дисциплін, пов'язаних з медичною інформатикою.

**Програма 3**

Коди головної форми

```
Private Sub CommandButton1_Click()
Unload UserForm1
Unload UserForm2
Unload UserForm2
End Sub

Private Sub Frame1_Click()
'UserForm2.ListBox1.Clear
UserForm1.Hide
'Load UserForm2
UserForm2.ListBox1.Clear
End Sub
```

**Елементи вибору назви захворювання**

```
Private Sub OptionButton1_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton1.TabIndex
UserForm2.ListBox1.AddItem OptionButton1.Caption
UserForm2.Show
End Sub

Private Sub OptionButton2_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton2.TabIndex
UserForm2.ListBox1.AddItem OptionButton2.Caption
```



```
UserForm2.Show
End Sub

Private Sub OptionButton3_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton3.TabIndex
UserForm2.ListBox1.AddItem OptionButton3.Caption
UserForm2.Show
End Sub

Private Sub OptionButton4_Click()
Set shet1 = Sheets(1)
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton4.TabIndex
UserForm2.ListBox1.AddItem OptionButton4.Caption
UserForm2.Show
End Sub

Private Sub OptionButton5_Click()
Set shet1 = Sheets(1)
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton5.TabIndex
UserForm2.ListBox1.AddItem OptionButton5.Caption
UserForm2.Show
End Sub

Private Sub OptionButton6_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton6.TabIndex
UserForm2.ListBox1.AddItem OptionButton6.Caption
UserForm2.Show
End Sub
```

```
Private Sub OptionButton7_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton7.TabIndex
UserForm2.ListBox1.AddItem OptionButton7.Caption
UserForm2.Show
End Sub

Private Sub OptionButton8_Click()
Set shet1 = Sheets(1)
shet1.Range("b19").Select
shet1.Range("b19").Value = OptionButton8.TabIndex
UserForm2.ListBox1.AddItem OptionButton8.Caption
UserForm2.Show
End Sub
```

Коди другої форми

```
Private Sub UserForm_Click()

End Sub

Private Sub CheckBox1_Click()
End Sub

Private Sub CheckBox2_Click()
End Sub

Private Sub CheckBox3_Click()
End Sub

Private Sub CheckBox4_Click()
End Sub

Private Sub CheckBox5_Click()
End Sub
```



```
Private Sub CommandButton1_Click()
'Коди кнопки уведення симптомів
Dim r1 As Integer
Dim st1 As String
Set shet1 = Sheets(1)
Set ls1 = ListBox1
CommandButton2.Enabled = False
TextBox1.Text = " "
Dim ind As Integer
Set ls1 = ListBox1
TextBox1.Text = " "
CheckBox1.Caption = " "
CheckBox2.Caption = " "
CheckBox3.Caption = " "
CheckBox4.Caption = " "
CheckBox5.Caption = " "
CheckBox1.Value = False
CheckBox2.Value = False
CheckBox3.Value = False
CheckBox4.Value = False
CheckBox5.Value = False
CommandButton2.Enabled = True
Set shet1 = Sheets(1)
ind = shet1.Range("b19").Value
Select Case ind
Case 0
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b4").Value
CheckBox3.Caption = shet1.Range("b5").Value
CheckBox4.Caption = shet1.Range("b9").Value
CheckBox5.Caption = shet1.Range("b17").Value
shet1.Range("c18").Value = 0
Case 1
```

```
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b4").Value
CheckBox3.Caption = shet1.Range("b5").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("d18").Value = 0
Case 2
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b5").Value
CheckBox3.Caption = shet1.Range("b8").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("e18").Value = 0
Case 3
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b6").Value
CheckBox3.Caption = shet1.Range("b7").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("f18").Value = 0
Case 4
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b6").Value
CheckBox3.Caption = shet1.Range("b7").Value
CheckBox4.Caption = shet1.Range("b9").Value
CheckBox5.Caption = shet1.Range("b10").Value
shet1.Range("g18").Value = 0
Case 5
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b7").Value
CheckBox3.Caption = shet1.Range("b11").Value
shet1.Range("h18").Value = 0
Case 6
```



```
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b12").Value
CheckBox3.Caption = shet1.Range("b13").Value
shet1.Range("i18").Value = 0
Case 7
CheckBox1.Caption = shet1.Range("b12").Value
CheckBox2.Caption = shet1.Range("b14").Value
shet1.Range("j18").Value = 0
End Select
End Sub
```

**Коди кнопки аналізу симптомів**

```
Private Sub CommandButton2_Click()
'Коди кнопки
'Public ind As Integer
Dim ind As Integer
Dim irow, icol, r1, c1, c2 As Integer
Dim st1 As String
Dim mp(10) As Single
Set shet1 = Sheets(1)
Set ls1 = ListBox1
TextBox1.Text = " "
'Set ls2 = ListBox2
'Set ls3 = ListBox3
shet1.Range("c18").Value = 0
klu = shet1.Range("b19").Value
Select Case klu
Case 0
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("c3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("c4").Value
```

```
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("c5").Value
End If
If CheckBox4.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("c9").Value
End If
If CheckBox5.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("c17").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("c2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 1
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("d3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("d4").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("d5").Value
End If
If CheckBox4.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("d15").Value
End If
```



```
If CheckBox5.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("d16").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("d2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 2
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("e3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("e5").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("e8").Value
End If
If CheckBox4.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("e15").Value
End If
If CheckBox5.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("e16").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("e2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 3
If CheckBox1.Value Then
```

```
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("f3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("f6").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("f7").Value
End If
If CheckBox4.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("f15").Value
End If
If CheckBox5.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("f16").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("f2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 4
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("g3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("g6").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("g7").Value
```



```
End If
If CheckBox4.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("g9").Value
End If
If CheckBox5.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("g10").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("g2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 5
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("h3").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("h7").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("h11").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("h2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 6
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("i3").Value
End If
```

```
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("i12").Value
End If
If CheckBox3.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("i13").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("i2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
Case 7
If CheckBox1.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("j12").Value
End If
If CheckBox2.Value Then
shet1.Range("c18").Value = shet1.Range("c18").Value +
shet1.Range("j14").Value
End If
TextBox1.Text = "Гіпотеза хвороби " & shet1.Range("j2").Value &
" підтверджена на " & shet1.Range("c18").Value _
 & "%"
End Select
End Sub
'список гіпотез

Private Sub CommandButton3_Click()
'UserForm1.Show
UserForm2.Hide
UserForm2.ListBox1.Clear
TextBox1.Text = " "
CheckBox1.Caption = " "
CheckBox2.Caption = " "
```



```
CheckBox3.Caption = " "
CheckBox4.Caption = " "
CheckBox5.Caption = " "
CheckBox1.Value = False
CheckBox2.Value = False
CheckBox3.Value = False
CheckBox4.Value = False
CheckBox5.Value = False
'Unload UserForm2
'UserForm1.Show
End Sub
Private Sub CommandButton4_Click()
UserForm2.Hide
'Load UserForm3
UserForm3.Show
End Sub
Private Sub ListBox1_Click()
Dim ind As Integer
Set ls1 = ListBox1
TextBox1.Text = " "
CheckBox1.Caption = " "
CheckBox2.Caption = " "
CheckBox3.Caption = " "
CheckBox4.Caption = " "
CheckBox5.Caption = " "
CheckBox1.Value = False
CheckBox2.Value = False
CheckBox3.Value = False
CheckBox4.Value = False
CheckBox5.Value = False
CommandButton2.Enabled = True
Set shet1 = Sheets(1)
ind = shet1.Range("b19").Value
```

```
Select Case ind
Case 0
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b4").Value
CheckBox3.Caption = shet1.Range("b5").Value
CheckBox4.Caption = shet1.Range("b9").Value
CheckBox5.Caption = shet1.Range("b17").Value
shet1.Range("c18").Value = 0
Case 1
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b4").Value
CheckBox3.Caption = shet1.Range("b5").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("d18").Value = 0
Case 2
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b5").Value
CheckBox3.Caption = shet1.Range("b8").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("e18").Value = 0
Case 3
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b6").Value
CheckBox3.Caption = shet1.Range("b7").Value
CheckBox4.Caption = shet1.Range("b15").Value
CheckBox5.Caption = shet1.Range("b16").Value
shet1.Range("f18").Value = 0
Case 4
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b6").Value
CheckBox3.Caption = shet1.Range("b7").Value
```



```
CheckBox4.Caption = shet1.Range("b9").Value
CheckBox5.Caption = shet1.Range("b10").Value
shet1.Range("g18").Value = 0
Case 5
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b7").Value
CheckBox3.Caption = shet1.Range("b11").Value
shet1.Range("h18").Value = 0
Case 6
CheckBox1.Caption = shet1.Range("b3").Value
CheckBox2.Caption = shet1.Range("b12").Value
CheckBox3.Caption = shet1.Range("b13").Value
shet1.Range("i18").Value = 0
Case 7
CheckBox1.Caption = shet1.Range("b12").Value
CheckBox2.Caption = shet1.Range("b14").Value
shet1.Range("j18").Value = 0
End Select
End Sub
End Sub
```

# ТЕРМІНОЛОГІЧНИЙ СЛОВНИК

*Знання про найближче майбутнє*
*не можна отримати від богів і демонів,*
*не можна отримати його і шляхом роздумів,*
*не можна отримати і шляхом будь-яких обчислень.*
*Знання... можна отримати тільки від людей*
**Сун Цзи**

**Алгоритми пошуку розв'язків** – алгоритми розв'язування експертних задач з моделями знань на основі правил. Розрізняють два типи алгоритмів: локальні та керівні. За локальними алгоритмами розв'язують окремі задачі в просторі станів (декомпозиції). Алгоритми управління організовують процеси вибору локальних процедур і прийняття висновків про закінчення пошуку.

**База знань** – це програмний засіб, який забезпечує подання і використання знань для автоматизації розв'язування задач, коли часто необхідні непроцедурні типи знань: правила, міркування, евристики, пошукові запити. У зарубіжній науковій літературі є поширеним термін KBS-knowledge based system – системи, засновані на знаннях.

**Бектрекінг** – алгоритм локального пошуку для моделі знань на основі продукційних правил. Передбачає циклічне уточнення задачі пошуку в поступово розширюваному просторі опису задачі (за рахунок повернення до вибору альтернативних правил Прологу). Бектрекінг застосовується в логічних мовах програмування.

**Бектрекінг хронологічний** – алгоритм повернення до останнього за часом розгалуження шляху пошуку. Після повернення вибирають новий шлях.

**Бектрекінг за залежностями** – алгоритм з поверненням за залежностями, коли аналізують інформацію: про застосовані на шляху пошуку правила; про їхні висновки. Після повернення на новому шляху відкидаються помилкові дії.

**Використання знань** – технологія отримання розв'язків відповідно до представлення моделі знань для експертної задачі. Задача



набуття знань найуспішніше розв'язана в експертних технологіях. Розв'язується на етапі пошуку інженером знань. Часткову автоматизацію для цієї задачі здійснюють за допомогою систем інженерії знань (СІЗ).

**Гіпертекстова модель знань** – різновид моделі семантичної мережі, у вузлах якої розміщують фрагменти тексту, зображення та інші інформаційні одиниці. Забезпечує візуалізацію мережі вузлів гіпертексту. Дає змогу надавати доступ до великих обсягів різнорідної інформації: тексту, графіки, відео. Існує багато засобів для побудови гіпертексту: HTML, PHP, XML. Широко використовується в мережних засобах програмування та в Інтернет.

**Експерт** – це фахівець, який може професійно розв'язувати важко формалізовані задачі, використовуючи різні прийоми та евристики. Експерт має великий запас знань і може швидко розпізнавати типові задачі, які постають перед ним.

**Експертні системи (ЕС)** – програми, призначені моделювати міркування експерта під час розв'язання експертної задачі. Експертні системи являють собою обчислювальну структуру, що самостійно формує алгоритм розв'язання з можливого набору сконструйованих експертами підсистем логічного вибору (логічних правил) та обчислювальних операцій. Підсистеми вибирають відповідно до оцінок і порівнянь, сформульованих раніше експертами.

**Експертні інтегровані системи** – це ЕС, введені до архітектури систем проектування для автоматизації процесів обробки даних та процесів аналізу. Використовуються для аналізу вихідних текстів задач і розуміння їхнього змісту, керування вимогами та обмеженнями проектів, виробленням специфікацій, проектуванням, кодогенерацією, верифікацією.

**Експертні технології** – технології проектування ІС, зокрема ЕС, НС, ДЕС, КС, СППР, КСП. Застосовують до задач, де є можливим використання моделей знань та методів їх обчислення.

**Знання** – це основні закономірності предметної області, за допомогою яких можна розв'язувати наукові, виробничі та інші задачі. Це добре структуровані дані. Розрізняють алгоритмічні та неалгоритмічні знання. Алгоритмічні знання: алгоритми, процедури,

програми. Неалгоритмічні знання – це факти, правила, оцінки, поняття (математичні та нематематичні), евристики. Термін ***знання*** для програміста означає структури даних, які необхідні програмі, щоб вона могла робити "інтелектуальні висновки".

**Інжиніринг знань** – технічна дисципліна, яка вивчає методи інтегрування моделей знань у комп'ютерних системах для того, щоб вирішувати складні проблеми, які зазвичай потребують високого рівня знань. Відповідних фахівців називають інженерами знань.

**Індуктивне виведення** – метод узагальнення, який передбачає виведення загальних наслідків із даних-фактів. Використовується у формально-логічних системах як структурно-логічний метод. Індуктивне моделювання є одним із напрямів ШІ, зокрема в машинному навчанні та робототехнічних системах.

**Контекстне виведення** – сукупність умов (фактів, висновків), для якої є логічно обґрунтованим застосування правила на шляху виведення. Це може бути сукупність ознак, симптомів, фактів. Застосовується в продукційних моделях. Часто передбачає діалоговий режим взаємодії з користувачем.

**Логічна модель даних** – формалізм опису структур даних, зв'язків між ними, обмежень цілісності. Опис у такому формалізмі називається схемою.

**Метод дошки оголошень** – спосіб організації логічного пошуку, за якого базу знань поділяють на незалежні модулі. Кожен модуль "спостерігає" за своєю областю пам'яті-класної дошки і вводиться в роботу тільки тоді, коли там з'являється нова інформація. У процесі роботи модулі залишають отримані ними нові знання в своїх областях.

**Механізм логічного висновку** – частина експертної системи, що забезпечує одержання висновків-розв'язків на основі моделі знань, що містяться в БЗ. Цей механізм має забезпечувати подібність міркувань людини і може генерувати в процесі висновку нові знання – висновки, рішення. Для систем, заснованих на продукційному представленні правил, використовуються пошукові алгоритми.



**Модель знань** – сукупність структур даних (числово-символьних, табличних, графічних та ін.), спеціально організованих для конкретного алгоритму пошуку розв'язків в експертній задачі. Технологія побудови моделі знань містить такі елементи: методи і програмні засоби набуття і представлення знань.

**Нечітке виведення** – механізм логічного висновку, використовуваний за наявності нечітких даних у моделях знань. У продукційних системах він найчастіше заснований на використанні так званих ФАКТОРІВ УПЕВНЕНОСТІ (коефіцієнти визначеності). Перерахування факторів упевненості для правил і фактів здійснюють за допомогою формул нечіткої логіки Заде. Ці обчислення приєднують до мережі продукційних правил моделі знань. Сукупність формул нечіткої логіки враховує типові комбінації об'єднання правил.

**Нечіткі знання** – числово-символьні оцінки фактів і висновків правил у моделях знань для підтвердження правдоподібності (вірогідності) висновків. Нечіткість знань (умов, фактів, висновків правил) задається спеціальними елементами:

- коефіцієнтами імовірності;
- коефіцієнтами визначеності, довіри;
- нечіткими множинами;
- нечіткими відношеннями;
- лінгвістичними змінними.

**Онтологія** – формалізована специфікація ПО. Призначена для представлення та інтерпретації знань у формі, зручній для комп'ютерних технологій під час розв'язання окремих задач. Використовується в ШІ поряд із базами знань. Запис у формі множин

$$O = \{ X, R, F\},$$

X – множина (об'єктів) понять; Р – множина відношень; Ф – множина функцій (процедур інтерпретації). Існують засоби і технології побудови онтологій. Приклади – набори індексних файлів в системах ІПС, словники, тезауруси.

**Підсистема пояснень** – компонента експертної системи, яка проводить трасування логічних правил в моделі знань для пояснення шляхів отримання висновків з правил.

**Предметна область** – це множина об'єктів разом із знаннями про них, вираженими у вигляді правил (відношень). Відношення описують властивості об'єктів та зв'язки між ними.

**Представлення знань** – це методологія моделювання і формалізації концептуальних знань, орієнтована на комп'ютерну обробку. Формалізація цієї задачі і відповідні комп'ютерні технології є предметом інженерії знань. Для автоматизації представлення знань використовують програмні підсистеми ЕС-системи інженерії знань.

**Продукційні правила** – найпоширеніший формалізм представлення знань у ЕС. Продукція, чи продукційне правило, являє собою пари "умова" → "дія". Зміст її полягає в тому, що у разі виконання умови система повинна виконати дію, зазначену в першій частині продукції. Умова повинна використовувати інформацію про поточні стани бази знань, а дії можуть змінювати цей стан. Основні вимоги до системи продукційних правил моделі: несуперечність, розширюваність, адаптивність. Це загальні вимоги до правил моделі експертної задачі.

**Прямий пошук** – це локальний алгоритм пошуку у мережі правил. Пошук розпочинається з аналізу первинних фактів підтвердження проміжних гіпотез до закінчення – визначення кінцевої гіпотези у локальній мережі правил.

**Семантичні мережі** – формалізм подання знань у ЕС. Відповідно до цього методу знання подають у вигляді деяких об'єктів, пов'язаних різними відносинами типу "є частиною", "працює в...", "аналогічний", "є одруженим з..." тощо. Виписування всіх істотних для розглянутих об'єктів відносин і являє собою семантичну мережу. Об'єкти слугують вузлами мережі, відносини представляються дугами.

**Таксономія** – теорія класифікації і систематизації складно організованих природних, технічних, соціальних систем реальності, що мають зазвичай ієрархічну будову: органічний світ, об'єкти географії, геології, астрономії, космічні апарати тощо.

**Фрейм** – структура даних, яка може містити системно-структурний опис ПО. Така структура є зручною для моделювання



аналогій знань, опису областей з родовидовими, просторовими, причинними, функціональними зв'язками понять. Реалізують фрейм у сучасних засобах ООП за допомогою об'єктів-форм. На формі розміщують декларативні, функціональні та процедурні елементи, необхідні для моделювання знань експертної задачі.

**Фреймова модель знань** – сукупність фреймів, організована за принципом семантичної мережі. Окремі фрейми-форми є вузлами мережі, яка структурує знання про експертну задачу та схему пошуку. Особливості фреймової моделі знань:

- відсутність потреби у компіляції моделі знань. Пошук в системі фреймів проходить за заданими правилами обходу мережі і використовує режим інтерпретації фрейму (його елементів). Зручність візуального аналізу структури фрейму та його ручної обробки;
- легкість приєднання виклику процедур або інших фреймів;
- адаптованість до різного типу структур: фрейми-довідки; фрейми-сценарії; фрейми-події або ситуації;
- жорстка прив'язка моделі до алгоритму пошуку, важко вносити зміни та доповнення.

**Штучний інтелект (ШІ)** – науково-технічний напрям у галузі інформатики. На початку 80-х років у дослідженнях з ШІ сформувався самостійний напрям, що одержав назву "Експертні системи" (ЕС). Мета дослідження у межах цього напряму полягає в розробленні програм, що дають можливість одержувати результати, які за якістю й ефективністю подібні рішенням, одержуваним найбільш кваліфікованими фахівцями – експертами. Синонім словосполучення "експертні системи" – "системи, засновані на знаннях".

**KEE (Knowledge Engineering Enviromet)** – технологія (стандарт) представлення знань в США мовою ЛІСП. Вперше запропонована як мова з використанням фреймів фірмою Intelcorp.

**KADS (Knowledge Asqusition and Documentation Structuring)** – технологія (стандарт) представлення знань та проектування ЕС, поширений у Європі.

## СПИСОК ЛІТЕРАТУРИ


1. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интелектуальных систем. – СПб.: Питер, 2000. –384 с.
2. Глибовець М.М., Олецький О.В. Штучний інтелект. – К.: КМ "Академія", 2002. – 366 с.
3. Глушков В.М. Основи безпаперової інформатики. – М.: Наука, 1982. – 354 с.
4. Гриценко В., Бакаев А., Козлов Д. Экспертные системы и логическое программирование. – К.: Наук. думка, 1992. – 219 с.
5. Евангелос Петрусос. Visual Basic 6. Руководство разработчика: В 2 т. – К.: Изд. группа BHV, 2000.
6. Загорулько Ю.А, Попов И.Г., Щипунов В.В. Интегрированная технологическая среда для создания систем обработки знаний // РАН. Изв. Академии наук. Теория и системы управления. – 1995. – № 5. – С. 210–213.
7. Исскуственный интеллект: Справочник: В 3 т. – М.: Радио и связь,1990.
8. Кокорева Л.В., Перевозчикова О.Л., Ющенко Е.Л. Диалоговые системы и представление знаний. – К.: Наук. думка, 1993. – 220 с.
9. Леоненков А. Нечеткое моделирование в среде MathCad и fuzzyТЕСН. – СПБ: БХВ – Петербург, 2003. – 736 с.
10. Люгер Джордж Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. – 4-е изд. – М.: Вильямс, 2003. – 864 с.
11. Марселлус Д. Программирование экспертных систем на Turbo Prolog. – М.: Финансы и статистика, 1994. – 256 с.
12. Минский М. Фреймы для представления знаний. – М.: Энергия, 1979. – 151 с.
13. Пасічник В.В., Резніченко В.А. Організація баз даних та знань. – К.: BHV, 2006. – 386 с.
14. Джексон П. Введение в экспертные системы. – М.: Вильямс, 2001. – 624 с.
15. Проектування інформаційних систем: посіб. / За ред. В.С. Пономаренка. – К.: Вид. центр "Академія", 2002. – 488 с.
16. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – 2-е изд. – М.: Вильямс, 2006. – 1408 с.
17. Protege. Редактор онтологий и баз знаний для CLIPS. [Електронний ресурс]. – Режим доступу: http://protege. stanford.edu/index.html.
18. Сайт довідково-консультаційної системи EXSYS. [Електронний ресурс]. – Режим доступу: www.EXSYS.COM
19. Ситник В.Ф. Засоби дейтамайнінгу для аналізу бізнесових розв'язків // Науково-технічна інформація. – 2002. – № 3. – С. 60–64.

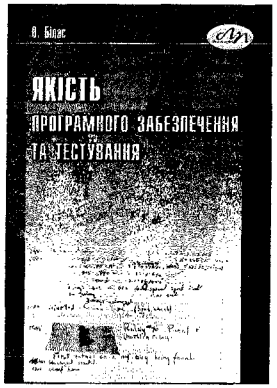




20. Суботін С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навч. посібник. – Запоріжжя: ЗНТУ, 2008. – 341 с.
21. Уотермен Д. Руководство по экспертным системам: Пер. с англ. / Под ред. В.Л. Стефанюка. – М.: Мир, 1989. – 385 с.
22. Федорчук Є. Менеджмент знань на основі інформаційних та телекомунікаційних технологій / Матеріали 7-ї Міжнар. наук.-практ. конф. "Наука і освіта 2004" Сучасні інформаційні технології, 10–25 лютого 2004 р. – Т. 72. – Дніпропетровськ: Наука і освіта, 2004. – С. 41–42.
23. Хамби Э. Программирование таблиц решений. – М.: Мир, 1976. – 86 с.
24. Хорев В.Д. Самоучитель программирования на VBA в Microsoft Office. – К.: Юниор, 2001. – 296 с.
25. Bos C., Botella B. and Vanheeghe P. Modeling and Simulating Human behaviors with Conceptual Graphs,in Proceedings of the Fifth International Conference on Conceptual Structures Seattle, WA., 1997. – P. 275–289.
26. Edward Feigenbaum, Pamela McCorduck. The fifth generation: Japan's computer challenge to the world. CREATIVE COMPUTING. – Vol. 10. – № 8. – 1984. – P. 103–110.
27. Nelson, T. H. A file structure for the complex, the changing, and the indeterminate. Proceedings of the 20 National ACM Conference. – 1965. – P. 84–100.
28. Newell F., Shaw J.C. and Simon H.A. Elements of a theory of human problem solving. Psychological Review, 65. – 1958. – P. 151–166.
29. Norbert Wiener. Cybernetics: Or Control and Communication in the Animal and the Machine. Paris, (Hermann & Cie) & Camb. Mass. (MIT Press) 2nd revised ed. 1961.
30. Marc Le Goc: SACHEM, a Real-Time Intelligent Diagnosis System Based on the Discrete Event Paradigm Computer Simulation International, Nov. – 2004. – Vol. 80, № 11. – P. 591–617.
31. Idiogrid: Idiographic Analysis with Repertory Grids. Електронний ресурс]. – Режим доступу: http://www.idiogrid. com/examine.html
32. Quinlan J. R. Induction of decision trees // Machine Learning. – 1986. – 1. – P. 81–106.
33. Shortliffe E. Computer based medical consultations: MYCIN. American Elsver, 1976.
34. Zadeh L.A. Fuzzy Logic and Approximate Reasoning. Synthese, 30 (1975). – P. 407–428.
35. Zadeh L.A. Fuzzy Sets. Information and Control, 1965. – P. 383–353.
36. Zagorulko Yu.A. A Program System For Efficient Operation With Embedded Semantic Network // Computers and Art. Intelligence. – 1984. – Vol. 3, № 6. – P. 483–494.

# ЗМІСТ

# Книги для навчання і роботи!




**Білас О. Є.**

**ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ**

*Навчальний посібник. – 2011. – 216 с.*
*ISBN 978-617-607-124-2*

Розглядаються способи розв'язання задач забезпечення та контролю якості розроблення програмних продуктів з позицій тестування. Викладено завдання процесу верифікації, описано процедуру формального інспектування різних складових програмного продукту, методи пошуку помилок, побудову середовищ тестування, рівні та види тестування. Посібник містить термінологічний словник з якості та тестування програмних продуктів.

Для студентів напряму "Програмна інженерія" вищих навчальних закладів та фахівців, що займаються розробленням програмних продуктів.


**Тимощук П. В.**

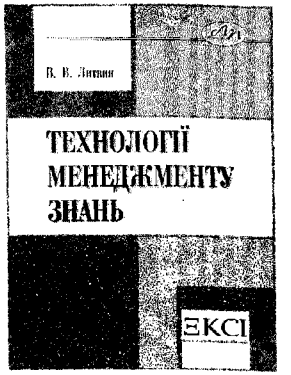
**ШТУЧНІ НЕЙРОННІ МЕРЕЖІ**

*Навчальний посібник. – 2011. – 444 с.*
*ISBN 978-617-607-063-4*

Штучні нейронні мережі, нейрокомп'ютери або паралельно розподілені процесори є спробою хоча б часткового моделювання структури і функцій мозку та нервових систем живих істот. Теорія нейронних мереж, яка інтенсивно розвивається приблизно з середини минулого століття, вивчає методи створення їхніх аналогових і дискретних математичних моделей, а також відповідних структурно-функціональних та принципових схем, призначених для виконання цих завдань. Викладено основи теорії штучних нейронних мереж та деякі приклади їхнього застосування. Зокрема, розглянуто математичні основи штучних нейронних мереж, їхню архітектуру та електронну реалізацію, стабільність функціонування мереж, мережі лінійного, квадратичного програмування та лінійних доповнювальних задач, алгоритми оптимізації без обмежень і навчальні алгоритми, мережі нелінійних задач оптимізації з обмеженнями, мережі задач дискретної та комбінаторної оптимізації, мережі ідентифікації сигналів і систем, методи моделювання нейронних осциляторів.

Для магістрів спеціальності 8.080402 "Інформаційні технології проектування", а також магістрів і студентів інших технічних спеціальностей.





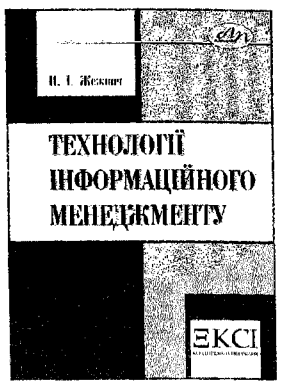

**Литвин В. В.**

**ТЕХНОЛОГІЇ МЕНЕДЖМЕНТУ ЗНАНЬ**

*Навчальний посібник. – 2010. – 260 с.*
*(Серія "Консолідована інформація". Випуск 2).*
*ISBN 978-966-553-975-9, 978-966-553-968-1 (випуск 2)*

Викладено основні технології менеджменту знань, особливу увагу звернуто на технології видобування, нагромадження, структуризації, формування, опрацювання даних та знань. Висвітлено особливості технологій менеджменту та інженерії знань, їх теоретичні та прикладні аспекти. Розглянуто моделі онтологічних систем та методику розроблення онтологій. Детально описано побудову онтології за допомогою програмного засобу Protégé. Наведено завдання для практичних завдань та подано перелік питань та завдання для самостійного розв'язування.

Для магістрів зі спеціальності "Консолідована інформація" специфічних категорій підготовки, а також бакалаврів та магістрів, що навчаються за напрямами галузей знань "Інформатика та обчислювальна техніка", "Системні науки та кібернетика", "Системна інженерія". Посібником можуть скористатися також студенти інших спеціальностей.




**Жежнич П. І.**

**ТЕХНОЛОГІЇ ІНФОРМАЦІЙНОГО МЕНЕДЖМЕНТУ**

*Навчальний посібник. – 2010. – 260 с.*
*(Серія "Консолідована інформація". Випуск 6).*
*ISBN 978-966-553-975-9*
*ISBN 978-617-607-012-2 (випуск 6)*

Окреслено комплекс теоретичних, методичних, технологічних та організаційно-практичних проблем з інформаційного менеджменту. Структура і зміст підручника відповідають програмі курсу для вищих навчальних закладів. Навчальний посібник містить основні методи, моделі та засоби інформаційного менеджменту, поняття інфодинаміки, методи оцінювання якості інформаційних систем та підходи до здійснення інформаційного маркетингу.

**Видавництво Львівської політехніки**
вул. Ф. Колесси, 2, корп. 23А, м. Львів, 79000
тел. +380 32 2582146, факс +380 32 2582136, http://vlp.com.ua, vmr@vlp.com.ua

НАВЧАЛЬНЕ ВИДАННЯ

Федорчук Євдоким Никифорович

# ПРОГРАМУВАННЯ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

## ЕКСПЕРТНІ СИСТЕМИ



Редактор *Ольга Дорошенко*
Коректор *Олена Сеник*
Технічний редактор *Лілія Саламін*
Комп'ютерне верстання *Галини Сукмановської*
Художник-дизайнер *Маріанна Рубель-Кадирова*

Здано у видавництво 05.03.2012. Підписано до друку 16.04.2012.
Формат $60\times84^{1}/_{16}$. Папір офсетний. Друк офсетний.
Умовн. друк. арк. 21,3. Обл.-вид. арк. 19,7.
Наклад 100 прим. Зам. 120192.

Видавець і виготівник: Видавництво Львівської політехніки
*Свідоцтво суб'єкта видавничої справи ДК № 751 від 27.12.2001 р.*
*вул. Ф. Колесси, 2, Львів, 79000*
тел. +380 32 2582146, факс +380 32 2582136
vlp.com.ua, ел. пошта: vmr@vlp.com.ua