ЕКСПЕРТНІ СИСТЕМИ

Лекції 1-2.

Початкові дані. Інженерія знань

Мета навчальної дисципліни.

Засвоєння студентами основних теоретичних відомостей та практичних умінь з курсу. Підготувати студента до ефективного використання сучасних експертних систем у подальшій професійній діяльності; допомогти набути навички практичної роботи з програмними засобами для вирішення задач підтримки прийняття діагностичних та керувальних рішень; навчити студентів вирішенню прикладних задач автоматизації керування, аналізу та прогнозування стану складних об'єктів і процесів на основі експертних систем.

Завдання навчальної дисципліни:

- формувати знання та практичні навички для використання експертних систем для розв’язання задач прийняття рішень;
- забезпечувати єдину методичну базу для взаємодії курсу експертних систем та інших предметних дисциплін;
- давати уяву про стан і перспективу розвитку експертних систем та їхнього програмного забезпечення.

Вимоги до знань та вміння студентів, які вивчили дисципліну повинні

знати:

- основні поняття та визначення теорії експертних систем;
- класифікацію експертних систем;
- математичні моделі подання знань в експертних системах;
- основні принципи побудови експертних систем;
- методи логічного виведення та пошуку рішень в експертних системах;

вміти:

- обґрунтовувати та аналізувати вибір конкретного типу моделі та методу подання знань для вирішення відповідних практичних задач;
- використовувати сучасні програмні засоби для вирішення задач побудови експертних систем;
- створювати бази знань для побудови та використання експертних систем;
- здійснювати підготовку даних для побудови моделей знань;
- представляти результати роботи із експертними системами;
- аналізувати результати побудови та використання моделей й вирішення практичних задач на основі експертних систем.

У нашому курсу висвітлюються питання теорії й практики розробки експертних систем.

Як інструментальне середовище розробки використається **експертна оболонка CLIPS**.

Суть технології CLIPS полягає в тім, що мова й середовище CLIPS надають користувачам можливість швидко створювати ефективні, компактні й легко керовані експертні системи. При цьому користувач застосовує множину вже готових інструментів (убудований механізм керування базою знань, механізм логічного висновку, менеджери різних об'єктів CLIPS і т.д.) і конструкцій (упорядковані факти, шаблони, правила, функції, родові функції, класи, модулі, обмеження, убудована мова COOL і т.д.).

Тема 1. Визначення експертної системи

День сьогодні з ранку не задався. Ви тільки що встановили нову версію текстового редактора, до якого давно звикли, але після клацання на його ярликові комп'ютер реагує зовсім не так, як хотілося б, - виводиться повідомлення начебто цього:

Call to Undefined Link (Виклик невизначеного зв'язку).

Як і більшість повідомлень про помилки, це допомагає не більше, ніж пророкування долі по стану Марса. Ви застосовуєте надзвичайний захід - видаляєте цілий каталог і переінсталюєте програму, але результат від цього не міняється. Ви починаєте міняти настроювання в різних файлах ініціалізації, але це теж не допомагає.

Нарешті, втомившись від безнадійних спроб, ви набираєте номер сервісної служби підтримки користувачів. І тільки після цього фортуна повертається до вас особою - на допомогу приходить людина, яка знає, про що говорить. Він радить вам викинути з півдюжини застарілих dll- модулів у системному каталозі й знову перевстановити програму. Виконавши його пораду, ви вже через десяток хвилин можете нормально працювати, кров'яний тиск, що недавно підскочив, знову повертається до норми.

Який би рівень експертного аналізу не був потрібний у даній області, ясно, що фахівець із сервісної служби здатний його зробити, а ви - ні. Хоча в ящику стола у вас лежить диплом кандидата технічних наук за фахом "Інформатика", і ви, можливо, прекрасно програмуєте завдання у своїй області, але, не маючи певного досвіду й підготовки, проблему усунення несправності розв'язати не змогли. Таким чином, здатність виконати експертний аналіз - це не тільки питання наявності певних знань і рівня кваліфікації. Для цього потрібно мати й дуже специфічні навички й умінням розібратися в конкретній ситуації в даній предметній області. Таким чином, бути експертом і мати загальна освіта - це далеке не те саме .

Едварда Фейгенбаума називають "батьком експертних систем", як це значиться на обкладинці однієї з його книг "Становлення експертної компанії". Він дійсно стояв у джерел експертної індустрії й створив першу експертну систему DENDRAL в області ідентифікації органічних сполук за допомогою аналізу мас-спектрограм.

Далі Фейгенбаум разом із Шортліфом і Бучананом спроектували першу медичну експертну систему MYCIN, при цьому вони зробили відкриття, якому було призначено істотно розширити сферу створення й використання експертних систем. Коли вони видалили із системи MYCIN базу знань (конкретну медичну інформацію), те залишилася частина, називана "машиною логічного висновку". Було показано, що базу знань можна змінювати й заміняти повністю, не порушуючи цілісності системи. Так виникла EMYCIN (Empty MYCIN - порожній MYCIN) або перша експертна оболонка - інструментальне середовище для побудови експертних систем різного призначення. З тих часів (із середини 70-х років XX сторіччя) з'явилося велике число подібних інструментальних систем MicroExpert, GURU, G2, JESS та ін.

1.1. Що таке експертний аналіз

Замисліться над таким питанням: **"При виконанні яких умов комп'ютерну програму можна назвати експертом?”**

Цілком логічно зажадати, щоб така програма мала знання. Просто здатність виконувати деякий алгоритм, наприклад робити аналіз списку елементів на наявність якої-небудь властивості, явно не відповідає цій вимозі. Це однаково , що дати першому випадковому перехожому список питань і відповідей і очікувати від нього успішного виконання пошуку й усунення несправностей у системах певного типу. Раніше або пізніше, але він обов'язково зіштовхнеться із ситуацією, не передбаченою в тому списку, яким його забезпечили.

Знання, якими має програма, повинні бути сконцентровані на певну предметну область. Випадковий набір імен, дат і місць подій, сентенцій із класиків і т.п. - це аж ніяк не ті знання, які можуть послужити основою для програми, що претендує на здатність виконати експертний аналіз. Знання припускають певну організацію й інтеграцію - тобто окремі відомості повинні співвідноситися один з одним і утворювати щось начебто ланцюжка, у якому одна ланка "тягне" за собою наступне.

І, нарешті, із цих знань повинне безпосередньо випливати вирішення проблем. Просто продемонструвати свої знання, що стосуються, наприклад, технічного обслуговування комп'ютерів, - це далеко не те ж саме, що привести комп'ютер в "почуття". Точно так само, отримати доступ до оперативної документації - це зовсім не те ж саме, що роздобути у своє розпорядження фахівця (або програму), здатного впоратися з виниклими проблемами.

Тепер спробуємо підсумувати ці міркування в наступному формальному визначенні експертної системи.

Експертна система - це програма для комп'ютера, яка оперує зі знаннями в певній предметній області з метою вироблення рекомендацій або вирішення проблем.

Експертна система може повністю взяти на себе функції, виконання яких звичайно вимагає залучення досліду людину-фахівця, або відігравати роль асистента для людини, що ухвалює розв'язок. Інакше кажучи, система (технічна або соціальна), що вимагає ухвалення рішення, може одержати його безпосередньо від програми або через проміжну ланку - людини, яка спілкується із програмою. Той, хто ухвалює рішення, може бути експертом зі своїми власними правами, і в цьому випадку програма може "виправдати" своє існування, підвищуючи ефективність його роботи.

Альтернативний варіант - людей, що працює в співробітництві з такою програмою, може добитися з її допомогою результатів більш високої якості. Загалом кажучи, правильний розподіл функцій між людиною й машиною є однією із ключових умов високої ефективності впровадження експертних систем. Технологія експертних систем є одним з напрямків нової області дослідження, яка одержала найменування **штучного інтелекту (Artificial Intelligence - AI)**. Дослідження в цій області сконцентровані на розробці й впровадженні комп'ютерних програм, здатних емулювати (імітувати, відтворювати) ті області діяльності людину, які вимагають мислення, певного майстерності й накопиченого досвіду. До них належать завдання прийняття рішень, розпізнавання образів і розуміння людської мови. Ця технологія вже успішно застосовується в деяких областях техніки й життя суспільства - органічної хімії, пошуку корисних копалин, медичній діагностиці.

Перелік типових завдань, які вирішуються експертними системами, включає:

- добування інформації з первинних даних (таких як сигнали, що надходять від гідролокатора);
- діагностика несправностей (як у технічних системах, так і в людському організмі);
- структурний аналіз складних об'єктів (наприклад, хімічних сполук);
- вибір конфігурації складних багатокомпонентних систем (наприклад, розподілених комп'ютерних систем);
- планування послідовності виконання операцій, що приводять до заданої цілі (наприклад, виконувані промисловими роботами).

Хоча відомі й "звичайні" програми, що спеціалізуються на певних завданнях із представленого переліку (або аналогічних їм у суміжних областях), у наступній главі ми покажемо, у чому полягає суттєва різниця між "звичайним" підходом і запропанованим у сфері штучного інтелекту й чому експертні системи можна виділити в окремий, досить добре помітний клас програм. Чіткого формального визначення експертної системи, яке всіх би задовольнило, не існує - приведене вище теж досить розпливчасте. Але проте існує досить багато важливих ознак, властивих тією чи іншою мірою всім експертним системам.

1.2. Основні характеристики експертних систем

Експертна система відрізняється від інших прикладних програм наявністю наступних ознак.

Моделює не стільки фізичну (або іншу) природу певної проблемної області, скільки **механізм мислення** людину стосовно до розв'язку завдань у цій проблемній області. Це суттєво відрізняє експертні системи від систем математичного моделювання або комп'ютерної анімації. Не можна, звичайно, сказати, що програма повністю відтворює психологічну модель фахівця в цій предметній області (експерта), але важливо, що основна увага все-таки приділяється відтворенню комп'ютерними засобами методики рішення проблем, яка застосовується експертом, -тобто виконанню деякої частини завдань так само (або навіть краще), як це робить експерт.

Система, крім виконання обчислювальних операцій, формує певні міркування й висновки, ґрунтуючись на тих знаннях, якими вона розташовує. Знання в системі представлені, як правило, на деякій спеціальній мові й зберігаються окремо від власне програмного коду, який і формує висновки й міркування. Цей компонент програми прийнято називати **базою знань**.

При розв'язку завдань основними є евристичні й наближені методи, які, на відміну від алгоритмічних, не завжди гарантують успіх. Евристика, по суті, є правилом впливу (rule of thumb), який у машинному виді уявляє деяке знання, придбане людиною в міру накопичення практичного досвіду вирішення аналогічних проблем. Такі методи є приблизними в тому розумінні, що, по-перше, вони не вимагають вичерпної вихідної інформації, і, по-друге, існує певний ступінь упевненості (або непевності) у тому, що пропонований розв'язок є вірним.

Експертні системи відрізняються й від інших видів програм з галузі штучного інтелекту.

Експертні системи мають справу із предметами реального світу, операції з якими звичайно вимагають наявності значного досвіду, накопиченого людиною. Множина програм з галузі штучного інтелекту є сугубо дослідницькими й основна увага в них приділяється абстрактним математичним проблемам або спрощеним варіантам реальних проблем (іноді їх називають "іграшковими" проблемами), а метою виконання такої програми є "підвищення рівня інтуїції" або відпрацьовування методики. Експертні системи мають яскраво виражену практичну спрямованість у науковій або комерційній галузі.

Однією з основних характеристик експертної системи є її продуктивність, тобто швидкість одержання результату і його вірогідність (надійність). Дослідницькі програми штучного інтелекту можуть і не бути дуже швидкими, можна примиритися й з існуванням у них відмов в окремих ситуаціях, оскільки, зрештою , - це інструмент дослідження, а не програмний продукт. А ось експертна система повинна за прийнятний час знайти розв'язок, який був би не гірше, чим те, яке може запропонувати фахівець у цій предметній області.

Експертна система повинна мати здатність пояснити, чому запропонований саме такий розв'язок, і довести його обґрунтованість. Користувач повинен отримати всю інформацію, необхідну йому для того, щоб бути впевненим, що розв'язок прийнятий "не зі стелі". НА відміну від цього, дослідницькі програми "спілкуються" тільки зі своїм творцем, який і так (швидше за все) знає, на чому ґрунтується її результат. Експертна система проектує розраховуючи на взаємодію з різними користувачами, для яких її робота повинна бути, по можливості, прозорою.

Найчастіше термін система, заснована на знаннях (knowledge-based system), використовується в якості синоніма терміна експертна система, хоча, строго кажучи, експертна система - це більш широке поняття. Система, заснована на знаннях, - це будь-яка система, процес роботи якої заснований на застосуванні правил відносин до символічної подання знань, а не на використанні алгоритмічних або статистичних методів. Таким чином, програма, здатна міркувати про погоду, буде системою, заснованої на знаннях, навіть у тому випадку, якщо вона не здатна виконати метеорологічну експертизу. А ось щоб мати право називатися метеорологічною експертною системою, програма повинна бути здатна давати прогноз погоди (інше питання - наскільки він буде достовірний).

Підсумовуючи все сказане, відзначимо - експертна система містить знання в певній предметній області, накопичені в результаті практичної діяльності людини (або людства), і використовує їх для вирішення проблем, специфічних для цієї області. Цим експертні системи відрізняються від інших, "традиційних" систем, у яких перевага віддається більш загальним і менш пов'язаним із предметною областю теоретичним методам, найчастіше математичним. Процес створення експертної системи часто називають інженерією знань (knowledge engineering) і він розглядається в якості "застосування методів штучного інтелекту" (див. [Feigenbaum, 1977]). Далі, у розділах 2 і 3, ми більш пильно розглянемо відмінність між загальноприйнятим у програмуванні підходом до вирішення проблем і тем, який пропонується при проектуванні експертних систем.

У частині, що залишився, цієї лекції ми розглянемо наступні питання.

Спочатку будуть перераховано чотири базові функції, які реалізуються в експертних системах. Ці функції тут будуть розглянуті дуже поверхово, з єдиною метою дати читачеві загальна уявлення про них і познайомити з відповідною термінологією.

1.3. Основні функції експертних систем

Оскільки теорія експертних систем виросла з більш загальної концепції штучного інтелекту, то немає нічого дивного в тому, що проблематика цих областей має багато спільного. На деяких з таких зв'язків акцентується увага в подальших розділах при огляді літератури. Ви також зустрінете в них заслання на подальші глави цієї книги, в яких та або інша тема розгледить детально.

1.3.1. Придбання знань

Бучанан у такий спосіб сформулював функцію придбання знань [Buchanan et al, 1983]: "[**Придбання знань це**] передача потенційного досвіду розв'язку проблеми від деякого джерела знань і перетворення його у вигляд, який дозволяє використовувати ці знання в програмі".

Передача знань виконується в процесі досить тривалих і великих співбесід між фахівцем із проектування експертної системи (будемо надалі називати його інженером по знаннях) і експертом у певній предметній області, здатним досить чітко сформулювати наявний у нього досвід. За існуючими оцінками, таким методом можна сформувати від двох до п'яти "елементів знання" (наприклад, правил впливу) у день. Звичайно, це дуже низька швидкість, а тому багато дослідників розглядають функцію придбання знань у якості одного з головних "вузьких місць" технології експертних систем [Feigenbaum, 1977].

Причин такої низької продуктивності предосить. Нижче перелічені тільки деякі з них.

Фахівці у вузькій області, як правило, користуються власним жаргоном, який важко перевести на звичайний "людський" мову (див. урізання 1.1). Але зміст жаргонного "слівця" аж ніяк не очевидний, а тому потрібно досить багато додаткових питань для уточнення його логічного або математичного значення. Наприклад, фахівці з військової стратегії говорять про "агресивну демонстрацію" іноземної військової моці, але при цьому не можуть пояснити, чому така "агресивна" демонстрація відрізняється від демонстрації, що не несе погрози.

Факти й принципи, що лежать в основі багатьох специфічних галузей знання експерта, не можуть бути чітко сформульовані в термінах математичної теорії або детермінованої моделі, властивості якої добре зрозумілі. Так, експертові у фінансовій галузі може бути відомо, що певні події можуть стати причиною зростання або зниження котирувань на фондовій біржі, але він нічого вам не скаже точно про механізми, які приводять до такого ефекту, або про кількісну оцінку впливу цих факторів. Статистичні моделі можуть допомогти зробити загальний довгочасний прогноз, але, як правило, такі методи не працюють відносно курсів конкретних акцій на коротких часових інтервалах.

Для того щоб розв'язати проблему в певній галузі, експерту недостатньо просто мати суму знань про факти й принципи в цій області. Наприклад, досвідчений фахівець знає, якого роду інформацією потрібно розташовувати для формулювання того або іншого судження, наскільки надійні різні джерела інформації і як можна розчленувати складну проблему на більш прості, які можна вирішувати більш-менш незалежно. Виявити в процесі співбесіди такого роду знання, засновані на особистому досвіді, що й погано піддаються формалізації, значно складніше, чим отримати простий перелік якихось фактів або загальних принципів.

Експертний аналіз навіть у дуже вузькій галузі, виконуваний людиною, дуже часто потрібно помістити в досить великий контекст, який включає й багато речей, що видадуться експерту саме собою, що розуміють, але для стороннього аж ніяк такими, що не являються. Візьмемо для прикладу експерта- юриста, який бере участь у судовому процесі. Дуже важко окреслити кількість і природу знань загального роду, які виявляються залучені в розслідування того або іншої справи.

1.3.2. Подання знань

Подання знань - ще одна функція експертної системи. Теорія подання знань - це окрема область досліджень, тісно пов'язана з філософією формалізму й когнітивною психологією. Предмет дослідження в цій області - методи асоціативного зберігання інформації, подібні тем, які існують у мозку людини. При цьому основна увага, природно, приділяється логічній, а не біологічній боку процесу, вилучаючи подробиці фізичних перетворень.

Приклад 1.1. Синтаксис і семантика представлення знань про сімейні відносини

Основна частина подання знань, на яку часто навіть не звертають особливої уваги, полягає в тому, що подання повинна якимось способом "стандартизувати" семантичну розмаїття людської мови. Ось кілька речень.

"Степан - батько Бориса".

"Степан - Бориса батько".

"Бориса батько - Степан".

"Батьком Бориса є Степан".

Усі ці фрази виражають ту саму думку (семантично ідентичні).

При машинній поданні цієї думки (знання) ми намагаємося знайти більш простий метод зіставлення форми й змісту, чим у звичайній людській мові, тобто добитися того, щоб вирази з однаковим (або схожим) змістом були однаковими й за формою. Наприклад, усі приведені вище фрази можуть бути зведені до виразу в такій формі:

батько (Степан. Борис)

батько (Степан, Василь)

В 70- х роках дослідження в області подання знань розвивалися в напрямках розкриття принципів роботи пам'яті людину, створення теорій добування відомостей з пам'яті, розпізнавання й відновлення. Деякі з досягнутих у теорії результатів привели до створення комп'ютерних програм, які моделювали різні способи зв'язування понять (*концептів*). З'явилися комп'ютерні застосування, які могли деяким чином відшукувати потрібні "елементи" знання на певному етапі розв'язку деякої проблеми. Згодом психологічна вірогідність цих теорій відійшла на другий план, а основне місце, принаймні з погляду проблематики штучного інтелекту, зайняла їхня здатність служити інструментом для роботи з новими інформаційними й керувальними структурами.

Загалом , питання подання знання було й швидше за все залишиться питанням суперечливим. Філософи й психологи найчастіше бувають шоковані безцеремонністю фахівців зі штучного інтелекту, які жваво базікають про людське знання на жаргоні, що уявляє дику суміш термінології, узятої з логіки, логістики, філософії, психології й інформатики. З іншого боку, комп'ютерний формалізм виявився новаторським засобом постановки, а іноді й пошуку відповідей на важкі запитання, над якими сторіччями билися метафізики.

В області експертних систем подання знань цікавить нас в основному як засіб відшукання методів формального опису великих масивів корисної інформації з метою їх наступної обробки за допомогою символічних обчислень. Формальний опис означає впорядкування в рамках якої-небудь мови, що володіє достатньо чітко формалізованим синтаксисом побудови виразів і такого ж рівня семантикою, що погоджує значення виразу з його формою.

Символьні обчислення позначають виконання нечислових операцій, у яких можуть бути сконструйовані символи й символьні структури для подання різних концептів і відносин між ними.

В області штучного інтелекту ведеться інтенсивна робота зі створення мов подання (representation languages). Під цим терміном розуміються комп'ютерні мови, орієнтовані на організацію описів об'єктів і ідей, на противагу статичним послідовностям інструкцій або зберіганню простих елементів даних. Основними критеріями доступу до подання знань є логічна адекватність, евристична потужність і природність, органічність нотації. .Ці терміни, швидше за все, потребують пояснень.

Логічна адекватність означає, що подання повинна мати здатність розпізнавати всі відмінності, які ви закладаєте у вихідну сутність. Наприклад, неможливо уявити ідею, що кожні ліки має який-небудь побічний небажаний ефект, якщо тільки не можна буде провести відмінність між призначенням конкретного лікарського препарату і його побічним ефектом (наприклад, аспірин збільшує виразкову хворобу). У більш загальному вигляді вираз, що передає цей ефект, звучить так: "кожні ліки має небажаний побічний ефект, специфічний для цього препарату".

Евристична потужність означає, що поряд з наявністю виразної мови подання повинна існувати деякий засіб використання вистав, сконструйованих і інтерпретованих таким чином, щоб з їхньою допомогою можна було розв'язати проблему. Часто виявляється, що мова, що володіє більшою виразною здатністю в термінах кількості семантичних відмінностей, виявляється й більше складним у керуванні описом взаємозв'язків у процесі розв'язку проблеми. Здатність до виразу в багатьох зі знайдених формалізмів може виявитися досить обмеженою в порівнянні з англійською мовою або навіть стандартною логікою. Часто рівень евристичної потужності розглядається по результату, тобто по тому, наскільки легко виявляється витягти потрібне знання стосовно до конкретної ситуації. Знати, які знання найбільше підходять для розв'язку конкретної проблеми, - це одне з якостей, яке відрізняє дійсно фахівця, експерта в певній галузі, від новачка або просто начитаний людини.

Природність нотації слід розглядати як якусь чесноту системи, оскільки більшість застосувань, побудованих на базі експертних систем, потребує накопичення великого обсягу знань, а розв'язати таке завдання досить важко, якщо угоди в мові подання занадто складні. Будь-який фахівець скаже вам, що при інших рівних характеристиках краще та система, з якої простіше працювати. Вирази, якими формально описуються знання, повинні бути по можливості простими для написання, а їх зміст повинен бути зрозумілий навіть тому, хто не знає, як же комп'ютер інтерпретує ці вирази. Прикладом може слугувати декларативний програмний код, який сам по собі дає досить чітка подання про процес його виконання навіть тому, хто не має подання про деталі реалізації комп'ютером окремих інструкцій.

За минулі роки було запропоновано чимало угод, придатних для кодування знань на мовному рівні. Серед них відзначимо *правила, що породжують* (production rules) [Davis and King, 1977], *структуровані об'єкти* (structured objects) [Findler, 1979] і *логічні програми* (logic programs) [Kowalski, 1979]. У більшості експертних систем використовується один або трохи з перерахованих формалізмів, а доводи на користь і проти кожного з них дотепер являють собою тему для жвавих дискусій серед теоретиків.

Більшість фрагментів програмного коду, наведених у цьому курсі, написані мовою CLIPS, у якому використовується комбінація правил, що породжують, і структурованих об'єктів. У Додатку ви знайдете досить великий опис основних концепцій і програмних конструкцій мови CLIPS, яке супроводжується безліччю прикладів.

1.3.3. Керування процесом пошуку розв'язку

При проектуванні експертної системи серйозна увага повинна бути приділене й тому, як здійснюється доступ до знань і як вони використовуються при пошуку розв'язку [Davis, 1980, a]. Знання про те, які знання потрібні в тієї або іншій конкретній ситуації, і вміння ними розпорядитися - важлива частина процесу функціювання експертної системи. Такі знання дістали найменування мета-знань - тобто знань про знання. Вирішення нетривіальних проблем вимагає й певного рівня планування й керування при виборі, яке питання потрібно задати, який тест виконати, і т. ін.

Використання різних стратегій перебору наявних знань, як правило, виявляє досить істотний вплив на характеристики ефективності програми. Ці стратегії визначають, яким способом програма відшукує розв'язок проблеми в деякому просторі альтернатив. Як правило, не буває так, щоб дані, якими розташовує програма роботи з базою знань, дозволяли точно "вийти" на ту область у цьому просторі, де має сенс шукати відповідь.

Більшість формалізмів подання знань може бути використане в різних режимах керування, і розробники експертних систем продовжують експериментувати в цій області. У наступних главах будуть описані системи, які спеціально підібрані таким чином, щоб проілюструвати відмінності в існуючих підходах до розв'язку проблеми керування. У кожній із представлених систем є що-небудь корисне для студентів, що спеціалізуються в галузі розробки й дослідження експертних систем.

Приклад. 1.2. Обслуговування автомобіля

Уявіть собі, що ваш автомобіль із труднощами заводиться, а в дорозі явно відчувається зниження потужності. Самі по собі ці симптоми недостатні для того, щоб ухвалити рішення, де ж шукати джерело несправності - у паливній або електросистемі автомобіля.

Пізнання в обладнанні автомобіля підказують - потрібно ще поекспериментувати, перш ніж звати на допомогу механіка. Можливо, погана паливна суміш, тому придивитеся до вихлопу й нагару на свічах. Можливо, збоїть розподільник - подивитеся, не ушкоджена чи його кришка. Ці досить специфічні евристики не гарантують, що віднайдеться дійсна причина, але раптом вам посміхнеться фортуна, і ви знайдете несправність без втомлюючої процедури послідовної перевірки всіх систем.

Швидше за все, ваших знань досить для того, щоб виконати загальну перевірку, перш ніж займатися доскональним вивченням окремих вузлів. Наприклад, подивитися, чи досить потужна іскра у свічі (якщо це так, то підозри з електросистеми можна зняти), перш ніж перевіряти акумулятор. При відсутності спеціальних евристик, чому більш методично ви будете діяти, тим більше шансів швидко знайти причину несправності. Загальне евристичне правило говорить:

"Спочатку перевір увесь вузол, а вже потім приступай до перевірки його компонентів".

Це правило можна вважати частиною режиму керування - систематичної стратегії застосування наявних знань.

Інше евристичне правило можна сформулювати, наприклад, так:

"Спочатку міняй більш дешеві деталі, а вже потім берися за більш дорогі".

У деяких випадках ці дві евристики можуть суперечити один одному, так що потрібно заздалегідь вибрати, яка з них має пріоритет у випадку, якщо обидві включені в той самий режим керування.

1.3.4. Роз'яснення ухваленого рішення

Питання про те, як допомогти користувачу зрозуміти структуру й функції деякого складного компонента програми, зв'язаний із порівняно новою галуззю взаємодії людину й машини, яка з'явилася на перетинанні таких областей, як штучний інтелект, промислова технологія, фізіологія й ергономіка. На сьогодні внесок у цю галузь дослідників, що займаються експертними системами, полягає в розробці методів подання інформації про поведінку програми в процесі формування ланцюжка логічних висновків при пошуку розв'язку.

Подання інформації про поведінку експертної системи важливо з багатьох причин.

Користувачі, що працюють із системою, потребують підтвердження того, що в кожному конкретному випадку висновок, до якого прийшла програма, в основному коректно.

Інженери, що мають справу з формуванням бази знань, повинні переконатися, що сформульовані ними знання застосовані правильно, у тому числі й у випадку, коли існує прототип.

Експертам у предметній області бажане простежити хід міркувань і спосіб використання тих відомостей, які з їхніх слів були введені в базу знань. Це дозволить судити, наскільки коректно вони застосовуються в даній ситуації.

Програмістам, які супроводжують, налагоджують і модернізують систему, потрібно мати у своєму розпорядженні інструмент, що дозволяє заглянути в "її нутро" на рівні більш високому, чим виклик окремих мовних процедур.

Менеджер системи, що використовує експертну технологію, який зрештою відповідає за наслідки розв'язку, прийнятого програмою, також потребує підтвердження, що ці розв'язки досить обґрунтовані.

Здатність системи пояснити методику ухвалення рішення іноді називають *прозорістю системи*. Під цим розуміється, наскільки просто персоналу з'ясувати, що робить програма й чому. Цю характеристику системи слід розглядати в сукупності з режимом керування, про який ішла мова в попередньому розділі, оскільки послідовність етапів ухвалення рішення тісно пов'язана із заданою стратегією поведінки.

Відсутність достатньої прозорості поведінки системи не дозволить експертові вплинути на її продуктивність або дати пораду, як можна її підвищити. Простежування й оцінка поведінки системи - завдання досить складне й для її розв'язку необхідні спільні зусилля експерта й фахівця з комп'ютерних наук.

Тема 2. Подання знань

2.1. Знання й дані

Якщо у вас є проблема або задача, яку не можна вирішити самостійно - ви звертаєтеся до знаючих людей, або до експертів, до тих, хто має ЗНАННЯ. Термін "системи, засновані на знаннях" (knowledge-based systems) з'явився в 1976 році одночасно з першими системами, що акумулюють досвід і знання експертів. Це були експертні системи (expert systems) MYCIN й DENDRAL [Shortliffe, 1976; Shortliffe Feigenbaum, Buchanan, 1978] для медицини й хімії. Вони ставили діагноз при інфекційних захворюваннях крові й розшифровували дані мас-спектрографічного аналізу.

Експертні системи з'явилися в рамках досліджень по штучному інтелекті (ШІ) (artificial intelligence) у той період, коли ця наука переживала серйозну кризу, і був потрібний істотний прорив у розвитку практичних додатків. Цей прорив відбувся, коли на зміну пошукам універсального алгоритму мислення й рішення задач дослідникам прийшла ідея моделювати конкретні знання фахівців-експертів. Так у США з'явилися перші комерційні *системи, засновані на знаннях, або експертні системи (ЕС).* Ці системи по праву стали першими інтелектуальними системами, і дотепер єдиним критерієм інтелектуальності є наявність механізмів роботи зі знаннями.

Так з'явився новий підхід до рішення задач штучного інтелекту — *подання знань.*

При вивченні інтелектуальних систем традиційно виникає питання - що ж таке знання й чим вони відрізняються від звичайних даних, десятиліттями оброблюваних на комп'ютерах. Можна запропонувати кілька робочих визначень, у рамках яких це стає очевидним.

Визначення 2.1

***Дані* — це інформація, отримана в результаті спостережень або вимірів окремих властивостей (атрибутів), що характеризують об'єкти, процеси та явища предметної області.**

Інакше, дані - це конкретні факти, такі як температура повітря, висота будинку, прізвище співробітника, адреса сайту й ін.

При обробці на ЕОМ дані трансформуються, умовно проходячи наступні етапи:

- D1 - дані як результат вимірів і спостережень;
- D2 - дані на матеріальних носіях інформації (таблиці, протоколи, довідники);
- D3 - моделі (структури) даних у вигляді діаграм, графіків, функцій;
- D4 - дані в комп'ютері мовою описи даних;
- D5 - бази даних на машинних носіях інформації.

Знання ж засновані на даних, отриманих емпіричним шляхом. Вони являють собою результат досвіду й розумової діяльності людини, спрямованої на узагальнення цього досвіду, отриманого в результаті практичної діяльності.

Так, якщо озброїти людини даними про те, що в нього висока температура (результат спостереження або виміру), те цей факт не дозволить йому вирішити задачу видужання. А якщо досвідчений лікар поділиться знаннями про те, що температуру можна знизити жарознижуючими препаратами й рясним питвом, те це істотно наблизить рішення задачі видужання, хоча насправді потрібні додаткові дані й більше глибокі знання.

Визначення 2.2

***Знання* — це зв'язку й закономірності предметної області (принципи, моделі, закони), отримані в результаті практичної діяльності й професійного досвіду, що дозволяє фахівцям ставити й вирішувати задачі в даній області.**

При обробці на ЕОМ знання трансформуються аналогічно даним:

- Z1 - знання в пам'яті людини як результат аналізу досвіду й мислення;
- Z2 - матеріальні носії знань (спеціальна література, підручники, методичні посібники);
- Z3 - поле знань - умовний опис основних об'єктів предметної області, їхніх атрибутів і закономірностей, їх єднальних;
- Z4 — знання, описані на мовах подання знань (продукційні мови, семантичні мережі, фрейми — *див. далі);*
- Z5 - база знань на машинних носіях інформації.

Часто використовується й таке визначення знань:

Знання — це добре структуровані дані, або дані про дані, або метадані.

Ключовим етапом при роботі зі знаннями є формування поля знань (третій етап Z3), ця нетривіальна задача включає виявлення й визначення об'єктів і понять предметної області, їхніх властивостей і зв'язків між ними, а також подання їх у наочній й інтуїтивно зрозумілій формі. Цей термін уперше був уведений при практичній розробці експертної системи по психодіагностиці АВТАНТЕСТ і тепер широко використається розробниками ЕС.

Без ретельного пророблення поля знань не може бути мови про створення бази знань.

Істотним для розуміння природи знань є способи визначення понять. Один із широко застосовуваних способів заснований на ідеї інтенсіонала й екстенсіонала.

Визначення 2.3

Інтенсіонал поняття **— це визначення його через співвіднесення з поняттям більш високого рівня абстракції із вказівкою специфічних властивостей.**

Наприклад, інтенсіонал поняття "МЕБЛІ": "предмети, призначені для забезпечення комфортного проживання людини й які захаращують будинок".

Визначення 2.4

***Екстенсіонал* — це визначення поняття через перерахування його конкретних прикладів, тобто понять більше низького рівня абстракції.**

Екстенсіонал поняття "МЕБЛІ": "Шафа, диван, стіл, стілець і т.д.".

Інтенсіонали формують знання про об'єкти, у те час як екстенсіонал поєднує дані. Разом вони формують елементи поля знань конкретної предметної області.

Для зберігання даних використаються бази даних (для них характерні великий об'єм і відносно невелика питома вартість інформації), для зберігання знань - бази знань (невеликого об'єму, але винятково дорогі інформаційні масиви).

База знань — основа будь-якої інтелектуальної системи, де знання описані на деякій мові подання знань, наближеній до природного.

Знання можна розділити на:

- глибинні;
- поверхневі.

Поверхневі — знання про видимі взаємозв'язки між окремими подіями й фактами в предметній області.

Глибинні — абстракції, аналогії, схеми, що відображують структуру й природу процесів, що протікають у предметній області. Ці знання пояснюють явища й можуть використатися для прогнозування поводження об'єктів.

Поверхневі знання

"Якщо ввести правильний пароль, на екрані комп'ютера з'явиться зображення робочого стола".

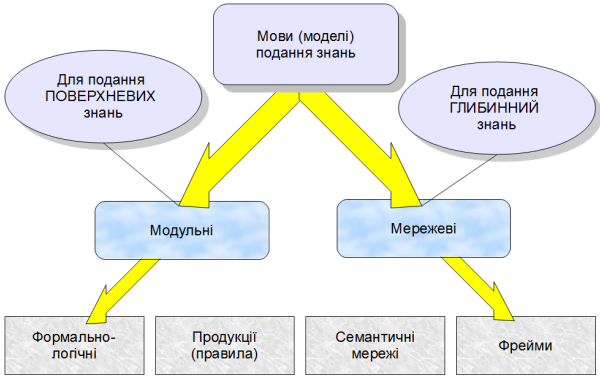
Глибинні знання

"Розуміння принципів роботи операційної системи й *знання на рівні кваліфікованого системного адміністратора"*.

Сучасні експертні системи працюють, в основному, з поверхневими знаннями. Це пов'язане з тим, що на даний момент немає універсальних методик, що дозволяють виявляти глибинні структури знань і працювати з ними.

Рис 2.1. Піраміда Н'юела.

Крім того, у підручниках по ШІ знання традиційно ділять на процедурні й декларативні. Історично первинними були процедурні знання, тобто знання, "розчинені" в алгоритмах. Вони управляли даними. Для їхньої зміни було потрібно змінювати текст програм. Однак з розвитком інформатики й програмного забезпечення все більша частина знань зосереджувала в структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних знань.



Сьогодні знання придбали чисто декларативну форму, тобто знаннями вважаються пропозиції, записані на мовах подання знань, наближених до природної мови й зрозумілих неспеціалістам.

Один з піонерів ШІ Алан Н'юел проілюстрував еволюцію засобів спілкування людини з комп'ютером як перехід від машинних кодів через символьні мови програмування до мов подання знань (рис. 2.1).

2.2. Моделі подання знань

2.2.1. Продуційна модель

МПЗ, засновані на правилах (rule-based), є найпоширенішими й більше 80% ЕС використають саме їх.

Визначення 1.5

Продуційна модель або модель, заснована на правилах, дозволяє представити знання у вигляді пропозицій типу "Якщо (умова), те (дія)".

Під "умовою" (антецедентом) розуміється деяка пропозиція-зразок, по якому здійснюється пошук у базі знань, а під "дією" (консеквентом) - дії, виконувані при успішному результаті пошуку (вони можуть бути проміжними, що виступають далі як умови, і термінальну або цільову, завершальну роботу системи).

Найчастіше висновок на такій базі знань буває прямій (від даних до пошуку мети) або зворотний (від мети для її підтвердження — до даних). Дані — це вихідні факти, що зберігаються в базі фактів, на підставі яких запускається машина висновку або інтерпретатор правил, що перебирає правила із продуційної бази знань.

Продукційна, модель так часто застосовується в промислових експертних системах, оскільки залучає розробників своєю наочністю, високої модульності, легкістю внесення доповнень і змін і простотою механізму логічного висновку.

Є велика кількість програмних засобів, що реалізують продуційний підхід (наприклад, мови високого рівня CLIPS й OPS 5; "оболонки" або "порожні" ЕС - EXSYS Professional і Kappa, інструментальні системи KEE, ARTS, PIES), а також промислових ЕС на його основі (наприклад, ЕС, створених засобами G2.

2.2.2.Семантичні мережі

Термін "семантична" означає "змістовна", а сама семантика — це наука, що встановлює відносини між символами й об'єктами, які вони позначають, тобто наука, що визначає зміст знаків. Модель на основі семантичних мереж була запропонована американським психологом Куїліаном. Основною її перевагою є те, що вона більше інших відповідає сучасним поданням про організації довгострокової пам'яті людини.

Визначення 1.6

***Семантична мережа* — це орієнтований граф, вершини якого — поняття, а дуги — відносини між ними.**

Як поняття звичайно виступають абстрактні або конкретні об'єкти, а відносини це зв'язку типу: "це" ("AKO — A-Kind-Of, "is" або "елемент класу"), "має частиною" ("has part"), "належить", "любить".

Можливо запропонувати кілька класифікацій семантичних мереж, пов'язаних з типами відносин між поняттями.

По кількості типів відносин:

- однорідні (з єдиним типом відносин);
- неоднорідні (з різними типами відносин).
- По типах відносин:
- бінарні (у які відносини зв'язують два об'єкти);
- N-арні (у які є спеціальні відносини, що зв'язують більше двох понять).

Найбільше часто в семантичних мережах використаються наступні відносини:

- елемент класу (троянда *це* квітка);
- атрибутивні зв'язки /мати властивість (пам'ять *має властивість* — об'єм);
- *значення* властивості (кольори *має значення* — жовтий);
- приклад елемента класу (троянда, *наприклад* — чайна);
- зв'язку типу *"частина-ціле"* (велосипед *включає* кермо);
- функціональні зв'язки (обумовлені звичайно дієсловами "робить", "впливає"...);
- кількісні *(більше, менше, дорівнює...);*
- просторові *(далеко від, близько від, за, під, над...);*
- часові *(раніше, пізніше, протягом...);*
- логічні зв'язки *(і, або, не)* і ін.

Мінімальний склад відносин у семантичній мережі такий:

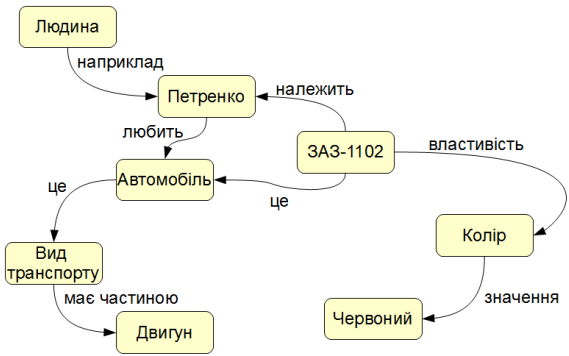
- елемент класу або АКО;
- атрибутивні зв'язки /мати властивість;
- значення властивості.

Недоліком цієї моделі є складність організації процедури організації висновку на семантичній мережі.

Ця проблема зводиться до нетривіальної задачі пошуку фрагмента мережі, що відповідає деякої підмережі, що відбиває поставлений запит до бази.

На мал. 2.3 зображений приклад семантичної мережі. Як вершини отут виступають поняття "людина", "Петренко", "ЗАЗ-1102", "автомобіль", "вид транспорту" й "двигун".

Для реалізації семантичних мереж існують спеціальні мережні мови, наприклад, NET, мова реалізації систем SIMER + MIR й ін. Широко відомі експертні системи, що використають семантичні мережі як мова подання знань - PROSPECTOR, CASNET, TORUS.



2.2.3. Фрейми

Термін *фрейм* (від англ. *frame* — "каркас" або "рамка") був запропонований Марвіном Мінським, одним з піонерів ШІ, в 70-і роки для позначення структури знань для сприйняття просторових сцен. Ця модель, як і семантична мережа, має глибоке психологічне обґрунтування.

Визначення 1.7

Фрейм - це абстрактний образ для подання стереотипу об'єкта, поняття або ситуації.

Інтуїтивно зрозуміло, що під абстрактним образом розуміється деяка узагальнена й спрощена модель або структура. Наприклад, проголошення вголос слова "кімната" породжує в почутому образ кімнати: "житлове приміщення із чотирма стінами, підлогою, стелею, вікнами й дверима, площею 6—20 м^2". Із цього опису нічого не можна забрати (наприклад, забравши вікна, ми одержимо вже прикомірок, а не кімнату), але в ньому є "дірки" або "слоти"— це незаповнені значення деяких атрибутів — наприклад, кількість вікон, кольори стін, висота стелі, покриття підлоги й ін.

У теорії фреймів такий образ кімнати називається фреймом кімнати. Фреймом також називається й формалізована модель для відображення образу.

Розрізняють фрейми-зразки або прототипи, що зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що надходять,. Модель фрейму є досить універсальної, оскільки дозволяє відобразити все різноманіття знань про світ через:

- *фрейму-структури*, *що* використаються для позначення об'єктів і понять (позика, застава, вексель);
- *фреймів-ролі* (менеджер, касир, клієнт);
- *фрейми-сценарії* (банкрутство, збори акціонерів, святкування іменин);
- *фреймів-ситуації* (тривога, аварія, робочий режим пристрою) і ін.

Традиційно структура фрейму може бути представлена як список властивостей:(ІМ'Я ФРЕЙМУ:

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

Той же запис можна представити у вигляді таблиці (див. табл. 2.1), доповнивши її двома стовпцями.

Таблиця 2.1. *Структура фрейму*

Ім'я фрейму			
Ім'я слоту	Значення слоту	Спосіб одержання значення	Приєднана процедура

У таблиці додаткові стовпці (3-й й 4-й) призначені для опису способу одержання слотом його значення й можливого приєднання до того або іншого слоту спеціальних процедур, що допускається в теорії фреймів. Як значення слоту може виступати ім'я іншого фрейму, так утворяться мережі фреймів.

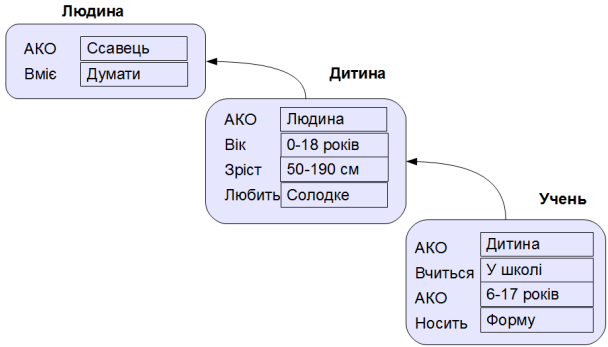
Існує кілька способів одержання слотом значень у фреймі-екземплярі:

- за замовчуванням від фрейму-зразка (Default-значення);
- через спадкування властивостей від фрейму, зазначеного в слоті АКО;
- по формулі, зазначеної в слоті;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з бази даних.

Найважливішою властивістю теорії фреймів є запозичення з теорії семантичних мереж - так називане спадкування властивостей. І у фреймах, і в семантичних мережах спадкування відбувається по Ако-связям (A-Kind-Of = це). Слот АКО вказує на фрейм більше високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Наприклад, у мережі фреймів на мал. 1.4 поняття "учень" успадковує властивості фреймів "дитина" й "людина", які перебувають на більше високому рівні ієрархії. На питання " чилюблять учні солодке?" треба відповідь "так", тому що цією властивістю володіють всі діти, що зазначено у фреймі "дитина". Спадкування властивостей може бути частковим: вік для учнів не успадковується із фрейму "дитина", оскільки зазначено явно у своєму власному фреймі.

Основною перевагою фреймів як моделі подання знань є те, що вона відбиває концептуальну основу організації пам'яті людини [Шенк, Хантер, 1987], а також її гнучкість і наочність.



Спеціальні мови подання знань у мережах фреймів FRL (Frame Representation Language) [Байдун, Бунін, 1990], KRL (Knowledge Representation Language) [Уотермен, 1989], фреймова "оболонка" Kappa [Стрельников, Борисов, 1997] й інші програмні засоби дозволяють ефективно будувати промислові ЕС. Широко відомі такі фрейм-ориентированные експертні системи, як ANALYST, МОДИС, TRISTAN, ALTERID [Ковригин, Перфильев, 1988; Николов, 1988; Sisodia, Warkentin, 1992].

2.2.4. Формальні логічні моделі

Традиційно в поданні знань виділяють *формальні логічні моделі*, *засновані на класичному вирахуванні предикатів 1-го порядку*, коли предметна область або задача описується у вигляді набору аксіом. Реальне вирахування предикатів 1-го порядку в промислових експертних системах практично не використається. Ця логічна модель застосовна в основному в дослідницьких "іграшкових" системах, тому що пред'являє дуже високі вимоги й обмеження до предметної області. У промислових же експертних системах використаються різні її модифікації й розширення, виклад яких виходить за рамки цього циклу лекцій.

2.3. Висновок на знаннях

Найбільше поширення одержала продуційна модель подання знань. При її використанні база знань складається з набору правил, а програма, що управляє перебором правил, називається машиною висновку.

Визначення 1.8

Машина висновку **(інтерпретатор правил) — це програма, що імітує логічний висновок експерта, що користується даною продуційною базою знань для інтерпретації даних, що надійшли в систему.**

Обычно вона виконує дві функції:

- перегляд існуючих даних (фактів) з робочої пам'яті (бази даних) і правил з бази знань і додавання (у міру можливості) у робочу пам'ять нових фактів;
- визначення порядку перегляду й застосування правил. Цей механізм управляє процесом консультації, зберігаючи для користувача інформацію про отримані висновки, і запитує в нього інформацію, коли для спрацьовування чергового правила в робочій пам'яті виявляється недостатньо даних [Осуга, Саэки, 1990].

У переважній більшості систем, заснованих на знаннях, механізм висновку являє собою невелику по об'єму програму й включає два компоненти - один реалізує безпосередньо висновок, інший - управляє цим процесом.

Дія *компонента висновку* засновано на застосуванні правила, називаного modus ponens: "Якщо відомо, що щиро твердження А, і існує правило виду "ЯКЩО А, ТО В", тоді твердження В також істинно".

Таким чином, правила спрацьовують, коли перебувають факти, що задовольняють їхній лівої частини: якщо щиро посилку, то повинне бути істинно й висновок.

*Компонент висновк*у повинен функціонувати навіть при недоліку інформації. Отримане рішення може й не бути точним, однак система не повинна зупинятися через те, що відсутня яка-небудь частина вхідної інформації.

Керуючий компонент визначає порядок застосування правив і виконує чотири функції:

1.*Зіставлення*— зразок правила зіставляється з наявними фактами.
2.*Вибір* — якщо в конкретній ситуації можуть бути застосовані відразу кілька правил, то з них вибирається одне, найбільш підходяще за заданим критерієм (дозвіл конфлікту).
3.*Спрацьовування* — якщо зразок правила при зіставленні збігся з будь-якими фактами з робочої пам'яті, те правило спрацьовує.
4. *Дія* — робоча пам'ять піддається зміні шляхом додавання в неї висновку правила, що спрацювало. Якщо в правій частині правила втримується вказівка на яку-небудь дію, то воно виконується (як, наприклад, у системах забезпечення безпеки інформації).

Інтерпретатор продукцій працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити тих, посилки яких збігаються з відомими на даний момент фактами з робочої пам'яті. Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, і потім цикл повторюється спочатку.

В одному циклі може спрацювати тільки одне правило. Якщо кілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, що спрацьовує в даному циклі. Цикл роботи інтерпретатора схематично представлений на мал. 2.5.

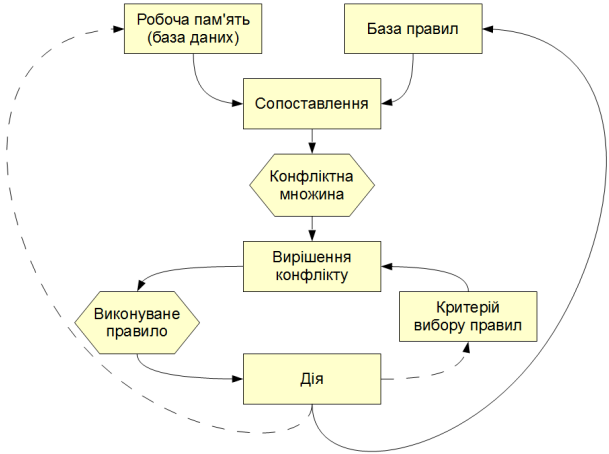


Рис. 2.5. Цикл роботи інтерпретатора

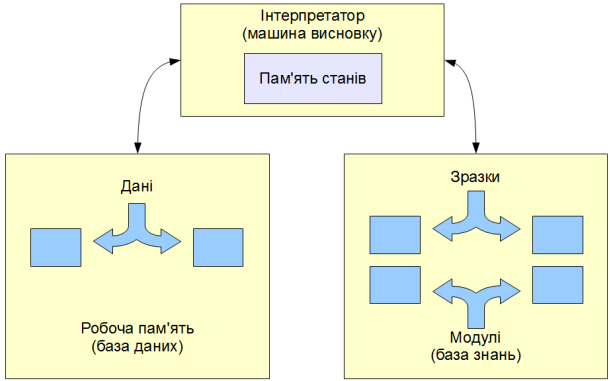


Рис. 2.6. Схема функціювання інтерпретатора

Інформація з робочої пам'яті послідовно зіставляється з посилками правил для виявлення успішного зіставлення. Сукупність відібраних правил становить так називану *конфліктну множину*. Для дозволу конфлікту інтерпретатор має критерій, за допомогою якого він вибирає єдине правило, після чого воно спрацьовує. Це виражається в занесенні фактів, що утворять висновок правила, у робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо ж на закінчення правила входить назва якої-небудь дії, то воно виконується.

Робота машини висновку залежить тільки від стану робочої пам'яті й від складу бази знань. На практиці звичайно враховується історія роботи, тобто поводження механізму висновку в попередніх циклах. Інформація про поводження механізму висновку запам'ятовується в пам'яті станів (мал. 2.6). Звичайно пам'ять станів містить протокол системи.

У системах із *прямим висновком* по відомих фактах відшукується висновок, що із цих фактів треба (див. мал. 1.7, ліва частина). Якщо такий висновок вдається знайти, то воно заноситься в робочу пам'ять. Прямий висновок часто називають висновком, керованим даними, або висновком, керованим антецедентами.

Існують системи, у яких висновок ґрунтується на сполученні згаданих вище методів - зворотного й обмеженого прямого. Такий комбінований метод одержав назву циклічного.

Нехай є фрагмент бази знань із двох правил:

- П1: Якщо "відпочинок - улітку" й "людина - активна", то "їхати в гори".

- П2: Якщо "любить сонце", те "відпочинок улітку".

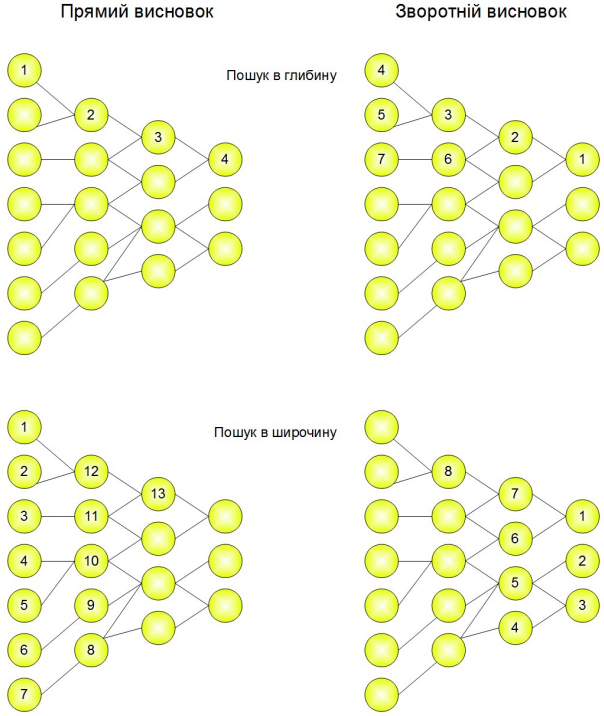


Рис. 1.7. Стратегії висновку

Припустимо, у систему надійшли факти - "людина активна" й "любить сонце".

ПРЯМИЙ ВИСНОВОК— виходячи з фактичних даних, одержати рекомендацію.

- 1-й прохід.

- *Крок 1.* Пробуємо /7/, не працює (не вистачає даних "відпочинок - улітку").
- *Крок 2.* Пробуємо /72, працює, у базу надходить факт "відпочинок - улітку".

- 2-й прохід.

- *Крок 3.* Пробуємо Я/, працює, активізується мета "їхати в гори",
  що і виступає як рада, що дає ЕС.

ЗВОРОТНИЙ ВИСНОВОК— підтвердити обрану мету за допомогою наявних правил і даних.

- 1-й прохід.

- *Крок 1.* Ціль — "їхати в гори": пробуємо *П1* — даних "відпочинок — улітку"
  ні, вони стають новою метою й шукається правило, де вона в лівої частини.
- *Крок 2.* Ціль "відпочинок — улітку": правило *П2* підтверджує мета й акти -
  візує її.

- 2-й прохід.

- Крок 3. Пробуємо П1, підтверджується шукана мета.

2.3.2. Методи пошуку в глибину й завширшки

У системах, база знань яких нараховує сотні правил, бажаним є використання стратегії керування висновком, що дозволяє мінімізувати час пошуку рішення й тим самим підвищити ефективність висновку. До числа таких стратегій ставляться: пошук у глибину, пошук завширшки, розбивку на підзадачі й альфа-бета-алгоритм.

При пошуку в глибину в якості чергової підцілі вибирається та, котра відповідає наступний, більше детальному рівню опису задачі. Наприклад, що діагностує система, зробивши на основі відомих симптомів припущення про наявність певного захворювання, буде продовжувати запитувати уточнювальні ознаки й симптоми цієї хвороби доти, поки повністю не спростує висунуту гіпотезу.

При пошуку завширшки, навпроти, система спочатку проаналізує всі симптоми, що перебувають на одному рівні простору станів, навіть якщо вони ставляться до різних захворювань, і лише потім перейде до симптомів наступного рівня детальності.

Розбивка на підзадачі має на увазі виділення підзадач, рішення яких розглядається як досягнення проміжних цілей на шляху до кінцевої мети. Прикладом, що підтверджує ефективність розбивки на підзадачі, є пошук несправностей у комп'ютері - спочатку виявляється підсистема, що відмовила (живлення, пам'ять і т.д.), що значно звужує простір пошуку. Якщо вдається правильно зрозуміти сутність задачі й оптимально розбити її на систему ієрархічно зв'язаних целей-підцілей, то можна домогтися того, що шлях до її рішення в просторі пошуку буде мінімальний.

Альфа-бета-алгоритм дозволяє зменшити простір станів шляхом видалення галузей, не перспективних для успішного пошуку. Тому проглядаються тільки ті вершини, у які можна потрапити в результаті наступного кроку, після чого безперспективні напрямки виключаються. Альфа-бета-алгоритм знайшов широке застосування в основному в системах, орієнтованих на різні ігри, наприклад, у шахових програмах.

2.4. Об'єкто-орієнтований підхід, об'єднаний із правилами

Найбільш розвиненим способом подання знань в ЕС є об'єкто-орієнтований підхід, що є розвитком фреймового подання. У його основі лежать поняття об'єкт і клас. У предметної області, що цікавить розробника, як об'єкти можуть розглядатися конкретні предмети, а також абстрактні або реальні сутності. Об'єкт має індивідуальність і поводженням, має атрибути, значення яких визначають його стан. Так, наприклад, конкретний покупець, роблячи замовлення, може виявитися в стані, коли грошей на його рахунку не вистачає для оплати, а його поводження в цьому випадку полягає у зверненні до банку за кредитом.

Кожен об'єкт є представником деякого класу однотипних об'єктів. Клас визначає загальні властивості для всіх його об'єктів. До таких властивостей ставляться склад і структура даних, що описують атрибути класу й відповідних об'єктів, і сукупність методів - процедур, що визначають взаємодію об'єктів цього класу із зовнішнім середовищем.

Об'єкти й класи мають характерні властивості, які активно використаються при об'єкто-орієнтованому підході й багато в чому визначають його переваги. До цих властивостей ставляться:

Інкапсуляція – приховання інформації. При об'єкто-орієнтованому програмуванні є можливість заборонити доступ до атрибутів об'єктів, доступ можливий тільки через його методи. Внутрішня структура об'єкта в цьому випадку схована від користувача, об'єкти можна вважати самостійними сутностями, відділеними від зовнішнього миру. Для того щоб об'єкт зробив деяку дію, йому ззовні необхідно послати повідомлення, що ініціює виконання потрібного методу. Інкапсуляція дозволяє змінювати реалізацію будь-якого класу об'єктів без побоювання, що це викличе небажані побічні ефекти в програмній системі. Тим самим спрощується процес виправлення помилок і модифікації програм.

Спадкування – можливість створювати із класів нові класи за принципом «від загального до частки». Спадкування дозволяє новим класам, при збереженні всіх властивостей клас^-батьків, додавати свої риси, що відбивають їхня індивідуальність. З погляду програміста новий клас повинен містити тільки коди й дані для нових або методів, що змінюються. Повідомлення, обробка яких не забезпечується власними методами класу, передається класу-батькові. Спадкування дозволяє створювати ієрархії класів й є ефективним засобом внесення змін і доповнень у програмні системи.

Поліморфізм – здатність об'єктів вибирати метод на основі типів даних, прийнятих у повідомленні. Кожен об'єкт може реагувати по-своєму на те саме повідомлення. Поліморфізм дозволяє спростити вихідні тексти програм, забезпечує їхній розвиток за рахунок введення нових методів обробки.

Отже, об'єкто-орієнтований підхід полягає в поданні системи у вигляді сукупності класів й об'єктів предметного середовища. При цьому ієрархічний характер складної системи відбивається у вигляді ієрархії класів, а її функціонування розглядається як взаємодія об'єктів, з якими асоціюється, наприклад, продуційні правила. Асоціювання продуційних правил ЕС з ієрархією класів здійснюється за рахунок використання загальних правил, як префікс яких звичайно використається посилання на клас, до якого дане правило застосовне. У процедурній інтерпретації наявність префікса, що зв'язує продукцію із класом, викликає необхідність перебору всіх екземплярів зазначеного в префіксі класу і його підкласів і перевірки істинності умови для атрибутів кожного з екземплярів.

Застосування об'єкто-орієнтованого підходу в системах інженерії знань виводить на перший план можливість природної декомпозиції задачі на сукупність підзадач, що представляють досить автономними агентами, що працюють зі знаннями. На сьогоднішній день це єдина практична можливість роботи в умовах експонентного росту складності (кількість взаємозв'язків), характерного для систем, що використають знання. Так практично всі інструментальні засоби для створення динамічних ЕС підтримують об'єкто-орієнтований підхід, об'єднаний із правилами.

2.5. Робота з нечіткістю

При формалізації знань існує проблема, що утрудняє використання традиційного математичного апарата. Це проблема опису понять, що оперують якісними характеристиками об'єктів (багато, мало, сильний, дуже сильний і т.п.). Ці характеристики звичайно розмиті й не можуть бути однозначно інтерпретовані, однак містять важливу інформацію (наприклад, "одним з можливих ознак грипу є висока температура").

Крім того, у задачах, розв'язуваних інтелектуальними системами, часто доводиться користуватися неточними знаннями, які не можуть бути інтерпретовані як повністю щирі або помилкові (логічні true/false або 0/1). Існують знання, вірогідність яких виражається деякою проміжною цифрою, наприклад 0,7.

Як, не руйнуючи властивості розмитості й неточності, представляти подібні знання формально? Для дозволу таких проблем на початку 70-х років XX століття американський математик Лотфи Заде запропонував формальний апарат нечіткої (fuzzy) алгебри й нечіткої логіки [Заде, 1972]. Пізніше цей напрямок одержав широке поширення [Орловський, 1981; Аверкин й ін., 1986; Яшин, 1990] і поклало початок однієї з галузей ШІ за назвою м'які обчислення (soft computing).

Л. Заде ввів одне з головних понять у нечіткій логіці - поняття лінгвістичної змінної.

Лінгвістична змінна (ЛП) — це змінна, значення якої визначається набором вербальних (тобто словесних) характеристик деякої властивості.

Наприклад, ЛП "ріст" визначається через набір {карликовий, низький, середній, високий, дуже високий}.

2.5.1. Основи теорії нечітких множин

Значення лінгвістичної змінної (ЛП) визначаються через так називані нечіткі множини (НМ), які у свою чергу визначені на деякому базовому наборі значень або базовій числовій шкалі, що має розмірність. Кожне значення ЛП визначається як нечітка множина (наприклад, НМ "низький ріст").

Нечітка множина визначається через деяку базову шкалу В и функцію приналежності НМ — μ (x), x_i В Д приймаючого значення на інтервалі [0; 1]. Таким чином, нечітка множина В — це сукупність пар виду (x, μ (x)). Часто зустрічається й такий запис:

$$B = \sum_{i=1}^{n} \frac{x_i}{\mu(x_i)},$$

де x_i — *i-те* значення базової шкали.

Функція приналежності визначає суб'єктивний *ступінь упевненості* експерта в тім, що дане конкретне значення базової шкали відповідає обумовленому НМ. Цю функцію не варто плутати з імовірністю, що носить об'єктивний характер і підкоряється іншим математичним залежностям.

Наприклад, для двох експертів визначення НМ "висока" для ЛП "ціна автомобіля" в умовних одиницях може істотно відрізнятися залежно від їх соціального й фінансового становища.

"Висока_ціна_автомобіля_1" = {50000/1 + 25000/0.8 + 10000/0.6 + 5000/0.4}

"Висока_ціна_автомобіля_2" = {25000/1 + 10000/0.8 + 5000/0.7 + 3000/0.4}

Нехай перед нами задача інтерпретації значень ЛП "вік", таких як "молодий" вік, "похилий" вік або "перехідний" вік. Визначимо "вік" як ЛП (мал. 1.8). Тоді "молодий", "похилий", "перехідний" будуть значеннями цієї лінгвістичної змінної. Більш повно, базовий набір значень ЛП "вік" наступний:

B = {дитячий, дитячий, юний, молодий, зрілий, похилий, старечий}.

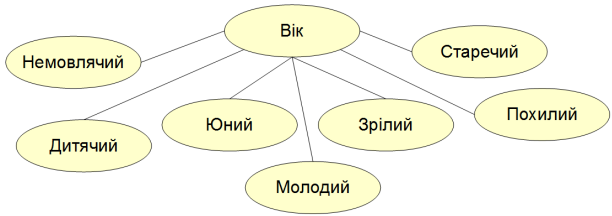


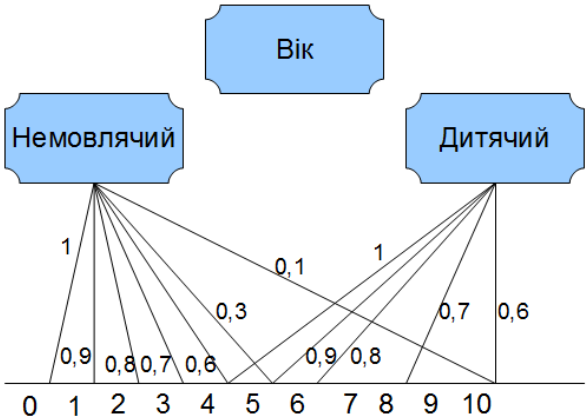
Рис. 1.8. Лінгвістична змінна "вік" і нечіткі множини, що визначають її значення

Для ЛП "вік" базова шкала - це числова шкала від 0 до 120, що позначає кількість прожитого років, а функція приналежності визначає, наскільки ми впевнені в тім, що дана кількість років можна віднести до даної категорії віку. На мал. 1.9 відбито, як ті самі значення базової шкали можуть брати участь у визначенні різних НМ.

Наприклад, визначити значення НМ "дитячий" можна так:

Рис.2.9. Формування нечітких множин

$$\text{немовлячий} = \left\{ \frac{0,5}{1} + \frac{1}{0,9} + \frac{2}{0,8} + \frac{3}{0,7} + \frac{4}{0,6} + \frac{5}{0,3} + \frac{10}{0,1} \right\}$$



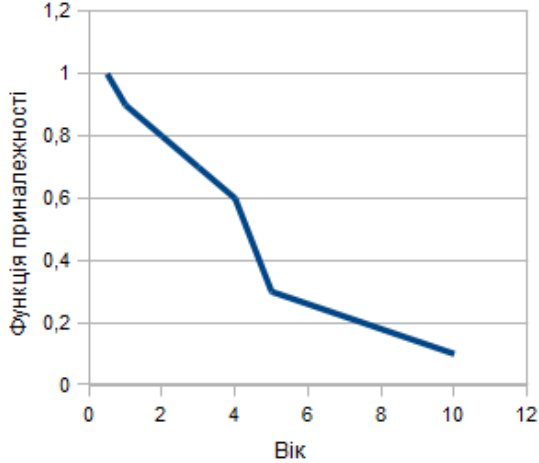


Рис.2.10. Графік функції приналежності нечіткій множині “немовлячий вік”.

Рис. 2.10 ілюструє оцінку НМ якимсь усередненим експертом, що дитини до напівроку з високим ступенем упевненості відносить до дитин ($\mu = 1$). Діти до чотирьох років зараховуються до дитин теж, але з меншим ступенем упевненості ($0,5 < \mu < 0,9$), а в десять років дитини називають так тільки в дуже рідких випадках - приміром, для дев'яностолітньої бабусі й 15 років може вважатися дитинством. Таким чином, нечіткі множини дозволяють при визначенні поняття враховувати суб'єктивні думки окремих індивідуумів.

2.5.2. Операції з нечіткими знаннями

Для операцій з нечіткими знаннями, вираженими за допомогою лінгвістичних змінних, існує багато різних способів. Ці способи є в основному евристиками.

Ми не будемо зупинятися на цьому питанні докладно, укажемо лише для приклада визначення декількох операцій. Наприклад, операція "АБО" часто задається так :

$$\mu(x) = \max(\mu 1(x), \mu 2(x)$$

(так називана логіка Заде)

або так:

$$\mu(x) = \mu 1(x) + \mu 2(x - \mu 1(x) * \mu 2(x)$$

(імовірнісний підхід).

Посилення йди ослаблення лінгвістичних понять досягається введенням спеціальних квантифікаторів. Наприклад, якщо поняття "старечий вік" визначається як

$$\left\{\frac{60}{0,6}+\frac{70}{0,8}+\frac{80}{0,9}+\frac{90}{1}\right\}$$

то поняття "дуже старечий вік" розпізнається як

$$con\,(A) = A^2 = \sum_i \frac{X_i}{\mu_i^2}$$

т. е. дуже старечий вік визначиться так:

$$\left\{\frac{60}{0,36}+\frac{70}{0,64}+\frac{80}{0,81}+\frac{90}{1}\right\}$$

Для висновку на нечітких множинах використаються спеціальні відносини й операції над ними.

Одним з перших застосувань теорії НМ стало використання коефіцієнтів упевненості для висновку рекомендацій медичної системи MYC1N. Цей метод використає кілька евристичних прийомів. Він став прикладом обробки нечітких знань, що вплинули на наступні системи.

У цей час у більшість інструментальних засобів розробки систем, заснованих на знаннях, включені елементи роботи із НМ, крім того, розроблені спеціальні програмні засоби реалізації так називаного нечіткого висновку, наприклад "оболонка" FuzzyCLIPS.