Навчальний курс

Технологія сховищ даних і знань

Лекція 5-6

Знання. Моделі подання знань

Лекції читає

Кандидат технічних наук, доцент

Баклан Ігор Всеволодович

baklaniv.at.ua iaa@ukr.net

5.1. 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 5.1

***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 5.2

***Знання* — це зв'язки 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 використається 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 5.3

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 захаращуючі будинок".

Визначення 5.4

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

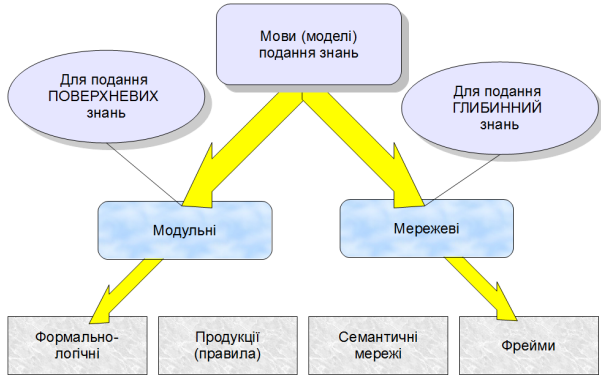
"Розуміння принципів роботи операційної системи й *знання на рівні кваліфікованого системного адміністратора".*

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 5.1. Піраміда Н'юела.

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. 5.1).

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.

5.2. Моделі подання знань

2.2.1. Продуційна модель

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

Визначення 5.5

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 Карра, інструментальні системи KEE, ARTS, PIES), а також промислових ЕС на його основі (наприклад, ЕС, створених засобами G2.

5.2.2.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" ("АКО — 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. 5.3 зображений приклад семантичної мережі. Як вершини отут виступають поняття "людина", "Петренко", "ЗАЗ-1102", "автомобіль", "вид транспорту" й "двигун".

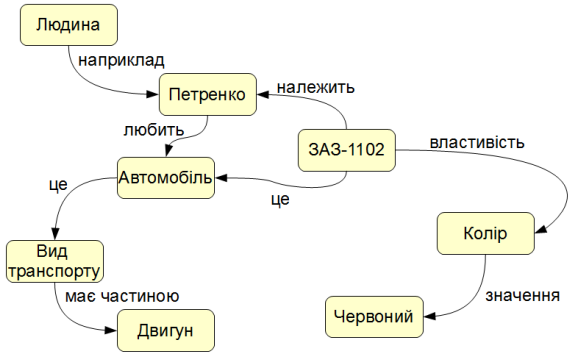


Рис.5.3. Приклад семантичної мережі

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.

5.2.3. 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 5.7

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. 5.4 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.

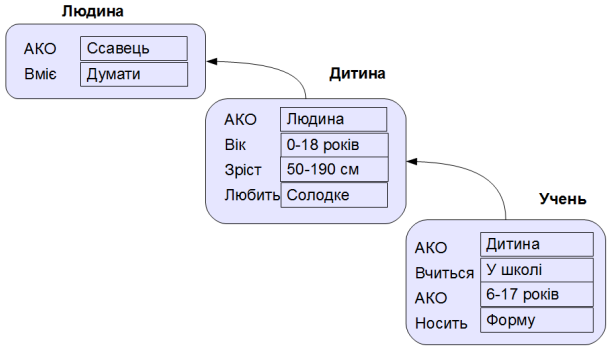


Рис.5.4. Приклад системи фреймів

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].

5.2.4. 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.

5.3. 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 5.8

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

Зазвичай 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, Саекі, 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. 5.5.

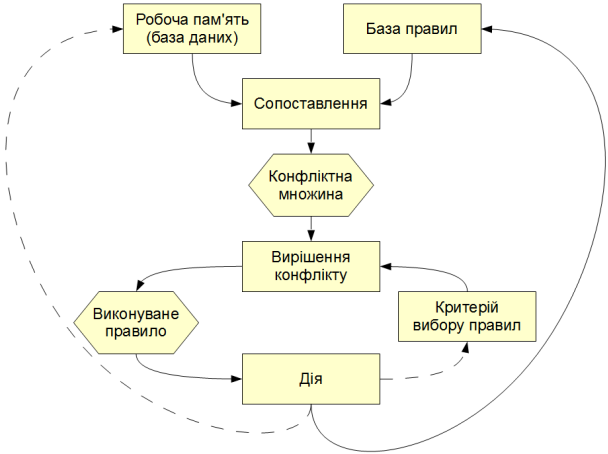


Рис. 5.5. Цикл роботи інтерпретатора

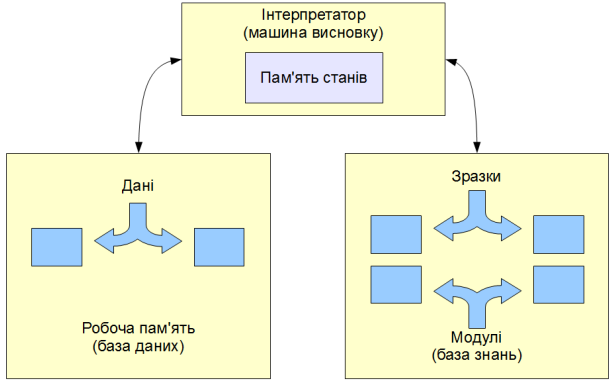


Рис. 5.6. Схема функціювання інтерпретатора

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. 5.6). Звичайно пам'ять станів містить протокол системи.

У системах із *прямим висновком* по відомих фактах відшукується висновок, що із цих фактів треба (див. мал. 5.7, 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", то "відпочинок улітку".

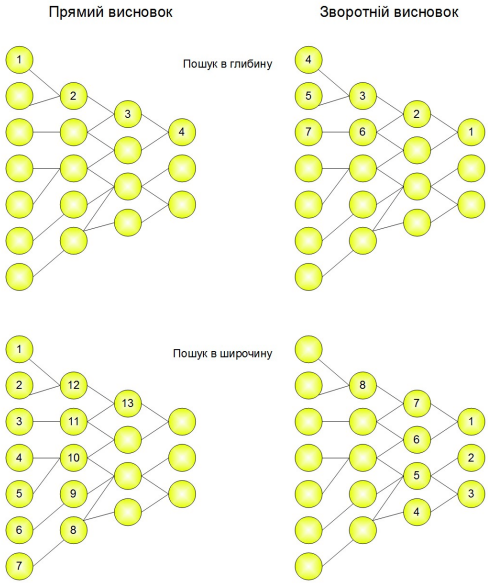


Рис. 5.7. Стратегії висновку

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, яку 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.

5.3.2. 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.

5.4. 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.

5.5. 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}.

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

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

де x_i — *i-me* 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 стоїть задача інтерпретації значень ЛП "вік", таких як "молодий" вік, "похилий" вік або "перехідний" вік. Визначимо "вік" як ЛП (мал. 5.8). 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}.

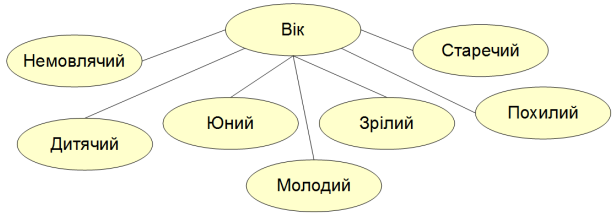


Рис. 5.8. Лінгвістична змінна "вік" і нечіткі множини, що визначають її значення

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:

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

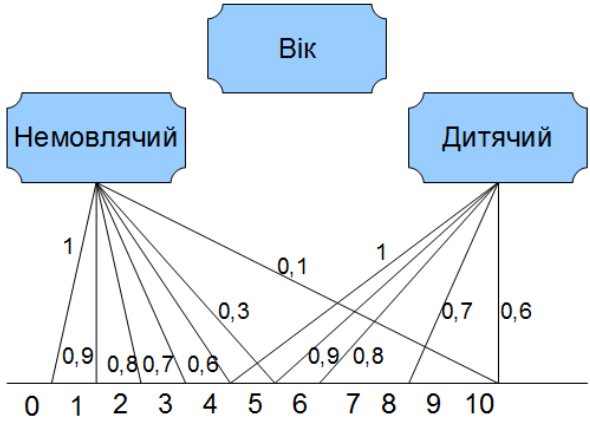


Рис.9.9. Формування нечітких множин

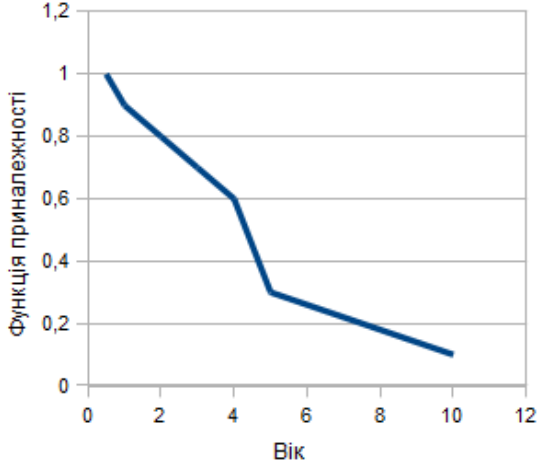


Рис.5.10. Графік функції приналежності нечіткій множині “немовлячий вік”.

Рис. 5.10 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 ($0,5 < < 0,9$), а μ 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.

5.5.2. 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 або 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

$$\sum_i \frac{X_i}{\mu_i^2}$$

$con\ (A) = A^2 =$

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.