ЛЕКЦІЯ 8
з навчальної дисципліни
«ШТУЧНИЙ ІНТЕЛЕКТ В ЗАДАЧАХ КІБЕРБЕЗПЕКИ»
Представлення знань і вивід на знаннях

План

1. Поняття даних та знань
2. Класифікація знань
3. Моделі представлення знань
4. Виведення на знаннях

1. Поняття даних та знань

При вивченні інтелектуальних систем виникає питання, що ж таке знання і чим вони різняться від традиційних даних, яким оперують ЕОМ. Отже,

Дані – це окремі факти, що характеризують об’єкти, процеси та явища предметної області, а також їх властивості. Під час обробки на ЕОМ дані трансформуються, проходячи через наступні етапи:

1. Дані, як результат вимірювання або спостереження.
2. Дані, що містяться на машинних носіях інформації (тексти, таблиці, числа, протоколи).
3. Впорядковані структури даних (діаграми, графіки, функції).
4. Дані у комп’ютері, що представлені мовою опису даних.
5. Бази даних на машинних носіях інформації.

Знання – це закономірності предметної області (принципи, закони, зв’язки), що набуті під час практичної діяльності та професійного досвіду, і в подальшому дозволяють фахівцям вирішувати задачі в цій області.

Під час обробки на ЕОМ знання, так само, як і дані трансформуються в кілька наступних етапів:

1. Знання у пам’яті людини, як результат спостереження та мислення.
2. Знання, що перенесені на матеріальні носії (книги, підручники, довідники).
3. Сформоване поле знань – це умовний опис основних об’єктів предметної області, їх властивостей і закономірностей, що пов’язують ці об’єкти.
4. Знання, що описано і закладено у моделі представлення знань (продукційна модель, семантична сітка, фрейми).
5. Бази знань на машинних носіях інформації.

Отже, знання – це добре структуровані дані про дані.

Існує багато способів визначення терміну «поняття», серед яких:

Інтенсіонал поняття (гіпернім) – це визначення поняття через співвідношення з поняттями вищого рівня абстракції з вказанням специфічних властивостей. Інтенсіонали формулюють знання про об’єкти.

Екстенсіонал поняття (гіпонім) – це співвідношення поняття об'єкту з поняттями низького рівня абстракції або перелік факторів, які відносяться до об'єкту, що визначається.

Для прикладу визначимо поняття «комп'ютер»

Інтенсіонал: комп'ютер, це електронний обчислювальний пристрій, який можна купити за 300-1000 $ і вирішувати за його допомогою широке коло завдань, і зокрема, спілкування в Інтернет.

Екстенціонал: комп'ютер – це IBM PC, Macintosh.

Для збереження даних використовують **Бази Даних**. Вони мають великий об'єм і відносно невелику вартість зберігання інформації.

Для збереження знань використовують **Бази Знань**. Тут, характерним є невеликий об'єм, але дорогі інформаційні масиви. База знань є основою для любої інтелектуальної системи.

2. Класифікація знань

Знання можна класифікувати за різними ознаками.

За рівнем завдання:

Поверхневі знання (нульовий рівень). Це знання про очевидні взаємозв'язки між окремими подіями і фактами в предметній області.

Наприклад, Якщо натиснути кнопку вимикача – загоряється лампочка. Якщо болить голова – потрібно вибити таблетку знеболюючого.

Глибинні знання (метазнання). Це абстракції, аналоги, схеми, які відбивають структуру та природу процесів у предметній області. Вони пояснюють явища і можуть використовуватися для передбачення поведінки системи.

Наприклад, принципова електрична схема проводки та лампочки, знання законів в фізиці. Знання лікарів та фізіологів про причини та види головних болів, а також методи їх лікування.

Сучасні інтелектуальні системи, що базуються на знаннях (наприклад, експертні системи) в основному працюють з поверхневими знаннями. Це пов'язано насамперед з тим, що важко розробити універсальну методику для виявлення і отримання глибинних знань, формування чіткої структури та можливості працювати з ними.

За рівнем деталізації:

В цьому разі до уваги береться ступінь деталізації знань. Кількість рівнів детальності залежить від специфіки задачі, обсягу наявних знань і обраної моделі представлення знань.

Наприклад, користувач запитує в експертної системи, як здійснити певну операцію. У разі найвищого рівня детальності система видає загальний план дій. Якщо цього є недостатньо, система може спуститися на рівень нижче і розписати операцію детальніше.

За способом представлення в алгоритмах:

Процедурні знання. Були історично першими. Процедурні знання розчиняються в алгоритмі, де чітко вказано план дій від А до Я. Процедурні знання керують даними, а для їх зміни – потрібно змінювати програму.

Декларативні знання. З розвитком ШІ пріоритет даних змінюється і більша частина знань зосереджується в структурах даних (таблиці, списки), де будуються взаємозв'язки між об'єктами. Розробнику лишається тільки вірно сформулювати завдання. Роль декларативних знань збільшується.

Сьогодні, знаннями вважаються речення, які наближені до природної мови, написані у форматі моделі представлення знань і є зрозумілими для нефахівця.

Властивості знань, що відрізняє їх від даних

Внутрішня інтерпретація. Кожна інформаційна одиниця повинна мати унікальне ім'я, за яким її знаходить інтелектуальна система, а також відповідає на запитання, де це ім.'я згадується.

Структурованість. Знання повинні мати гнучку структуру: одні інформаційні одиниці можуть міститися у складі інших (клас-ціле, елемент-клас).

Зв'язність. Між різними інформаційними одиницями можуть встановлюватися різні типи зв'язків (причинно-наслідкові, просторові).

Семантична метрика. Для інформаційних одиниць можна задавати відношення, що характеризують ситуаційну близькість.

Активність. Виконання програм в інтелектуальній системі повинно ініціюватися поточним станом бази знань.

3. Моделі представлення знань

Для використання в системах ШІ знання мають бути формалізовані та закладені в певну структуру, тобто представлені моделлю знань.

Модель представлення знань – це фіксована система понять і правил, на підставі якої інтелектуальна система здійснює операції над знаннями.

Моделі представлення знань потрібні для:

- Створення спеціальних мов опису знань.
- Формалізації процедур порівняння нових знань з існуючими.
- Формалізації механізму логічного виведення.

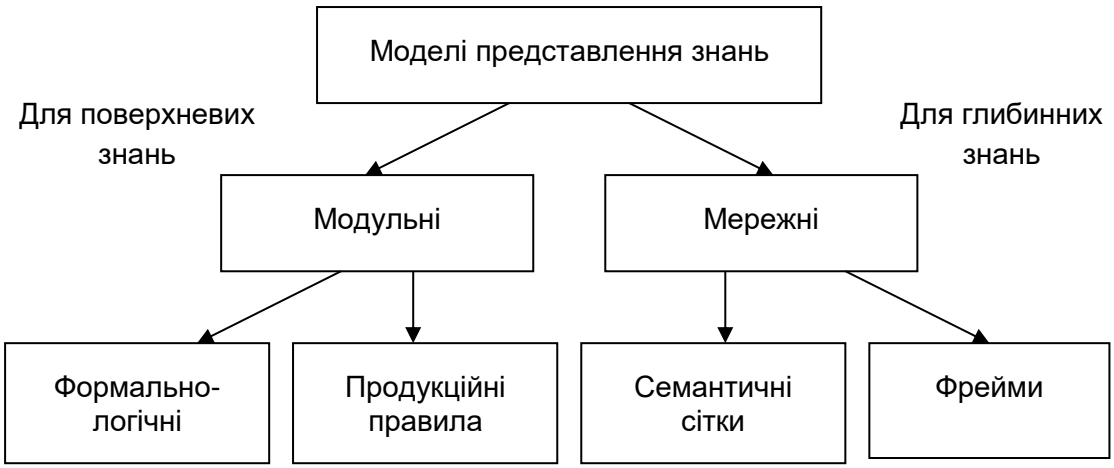
Існує кілька десятків моделей для різних предметних областей. Більшість з них можна звести до класичних моделей:

- Продукційні моделі
- Семантичні сітки
- Фреймові моделі
- Логічні моделі

Ці моделі можна розподілити на дві великі групи:

- **Модульні моделі**. Оперують окремими, не пов'язаними між собою елементами знань, будь то правила або аксіоми предметної області.

- **Мережні моделі.** Надають можливість пов'язувати елементи або фрагменти знань між собою через відношення в семантичних сітках або мережах фреймів.



Для успішного проведення операцій зі знаннями потрібно виокремлювати в базі знань певні фрагменти – **області знань**. Вони мають бути незалежними між собою, щоб відповідати наступним вимогам:

- Зміни в одній області не повинні спричиняти суттєвих змін в інших.
- Вирішення складної задачі можна поділити на вирішення підзадач, в яких буде достатньо наявних знань з певної області.
- Різні області знань можуть проектуватися незалежно одна від одної.

Області знань для інтелектуальних систем

- **Предметна область.** Містить знання про конкретну галузь, в якій працює інтелектуальна система.
- **Область мови.** Містить знання про мову, якою відбувається діалог.
- **Область системи.** Містить знання інтелектуальної системи про власні можливості.
- **Область користувача.** Містить знання про індивідуальні можливості користувача. Наприклад, вибір рівня пояснення системи відповідно до рівня фаховості користувача.
- **Область діалогу.** Містить знання про мету діалогу, форми та методи організації

Класичні моделі представлення знань

Продукційна модель

Продукції є найбільш популярними засобами подання знань в інформаційних системах. У загальному вигляді під продукцією розуміють вираз виду A ® B.

Звичайне прочитання продукції виглядає так: **ЯКЩО A, ТОДІ B**.

Імплікація може тлумачитися в звичайному логічному сенсі, як знак логічного слідування B з істинного A. Можливі й інші інтерпретації продукції, наприклад, A описує деяку умову, необхідну для виконання дії B.

Продукційна модель або модель, що заснована на правилах, дозволяє представити знання у вигляді речень типу

«Якщо (умова), то (дія)».

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

Наприклад:

Правило 1.

ЯКЩО (намір - відпочинок) і (дорога вибоїста) ТО (використовувати джип)

Правило 2.

ЯКЩО (місце відпочинку - гори) ТО (дорога вибоїста)

Продукційна модель привертає розробників своєю наочністю, високою модульність, легкістю внесення доповнень і змін і простотою механізму логічного висновку.

Сильні сторони систем продукцій:

- Модульність.
- Природність (виведення багато в чому аналогічне процесу міркування експерта).
- Простота подання знань та організації логічного виведення.
- Простота створення і розуміння окремих правил.
- Простота поповнення та модифікації.

Слабкі сторони систем продукцій:

- Невідповідність до структур знань, що притаманні людині.
- Неясність взаємовідношень між правилами.
- Складність оцінки цілісного образу знань.
- Низька ефективність обробки знань.

При розробці невеликих систем (десятки правил) проявляються в основному позитивні сторони продукційних моделей знань, однак при збільшенні обсягу знань більш помітними стають слабкі сторони.

Найчастіше продукційна модель застосовується у промислових експертних систем. Вона є наочною, має високу модульність, до неї легко вносити доповнення та зміни, має простий механізм логічного виведення.

Серед існуючих зразків ІС з продукційною моделлю представлення знань є оболонки EXSYS, EXPERT, EKO, KAPPA, а також промислові експертні системи на базі G2.

Семантична модель представлення знань

Дану модель подання знань було запропоновано американським психологом Куіллліаном. Вона добре відбиває представлення про організацію довготривалої пам’яті людини.

Семантичний підхід до побудови систем штучного інтелекту знаходить застосування в системах розуміння природної мови, в системах «питання-відповідь», в різних предметно-орієнтованих системах.

Термін **семантичний** означає змістовний, «**семантика**» - це наука, що встановлює відношення між символами і об'єктами, які вони позначають, тобто, це наука, що визначає зміст знаків.

В основі моделі лежить семантична мережа, яка в загальному випадку представляє інформаційну модель предметної області і має вигляд графа, вершини якого відповідають об'єктам предметної області, а дуги - відношення між ними.

Дуги можуть бути визначені в різний спосіб, що залежить від виду поданих знань. Зазвичай, дуги, які використовують для представлення ієрархії, містять дуги типу «множина», «підмножина», «елемент». Семантичні мережі, які застосовують для опису природних мов, використовують дуги типу «агент», «об'єкт», «реципієнт» (об'єкт, який отримує що-небудь від іншого об'єкта-донора).

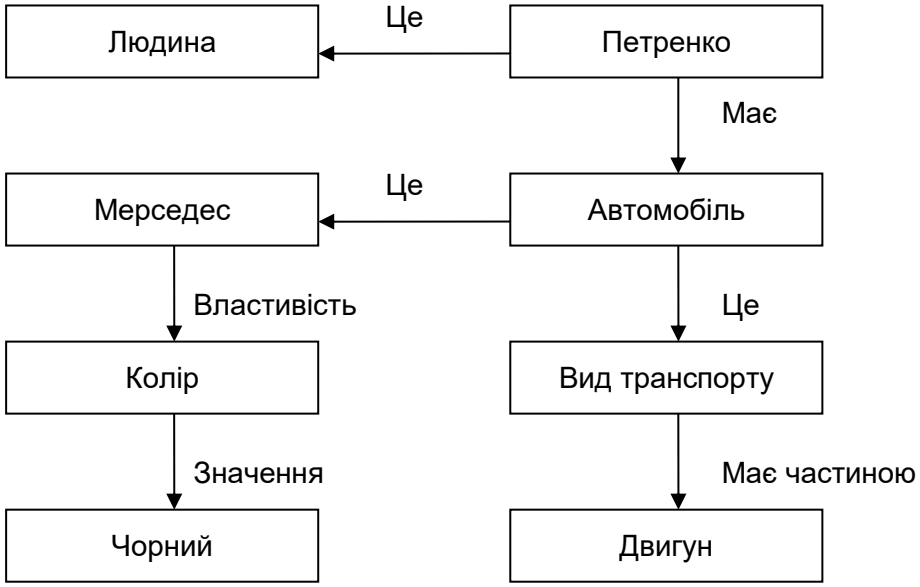
Поняттями, зазвичай, виступають абстрактні або конкретні об'єкти, а відношення - це зв'язки типу: «це» («is»), «має частиною» («has part»), «належить», «любить». Характерною особливістю семантичних мереж є обов'язкова наявність трьох типів відношень:

- клас - елемент класу.
- властивість – значення.
- приклад елемента класу.

Для семантичних сіток обов’язковим є наявність 3 типів відношень:

- Клас – елемент класу (квітка - троянда)
- Властивість – значення (колір - червоний)
- Елемент класу – приклад (троянда - баккара)

Приклади семантичної сітки



Класифікація семантичних сіток

За типом відношень

- **Однорідні**. З одним типом відношень
- **Неоднорідні**. З різними типами відношень

За кількістю відношень

- **Бінарні.** Відношення лише між 2-ма об’єктами
- **N-арні.** Відношення пов’язують більшу кількість об’єктів

Типи відношень

- Зв’язки типу «ціле - частина» (клас – підклас, множина - елемент)
- Функціональні зв’язки («випливає», «діє», «виробляє»)
- Кількісні зв’язки («більше», «менше»)
- Просторові зв’язки («далеко від», «близько до», «за», «під», «над»)
- Часові зв’язки («раніше», «пізніше», «протягом»)
- Атрибутивні зв’язки («має властивість», «має значення»)
- Логічні зв’язки («І», «АБО», «НІ»)
- Лінгвістичні зв’язки

Пошук рішення зводиться до пошуку фрагмента сітки (підсітки), де знаходиться відповідь на запит із бази знань.

Недоліком семантичної моделі можна вважати складність організації процедури пошуку та виведення.

Для реалізації семантичних сіток існують спеціальні мови, наприклад, NET. Також, відомими є зразки експертних систем: PROSPECTOR, CASNET, TORUS.

Фреймові моделі

У 1975 році Марвін Мінскі запропонував гіпотезу, згідно якої знання людини групуються у модулі – фрейми. Він розробив модель для позначення структури знань, що призначена для сприйняття просторових сцен. **Фреймом** тут називається структура даних, яка призначена для опису типових ситуацій або понять.

В психології та філософії відоме поняття абстрактного образу. Наприклад, слово «кімната» викликає у людини образ кімнати: «житлове приміщення з чотирма стінами, підлогою, стелею, вікнами та дверима, площею 6-20 м^2». З цього опису нічого не можна прибрати (наприклад, прибравши вікна, отримаємо вже комору, а не кімнату), але в ньому є «дірки», або «слоти», - це незаповнені значення деяких атрибутів - кількість вікон, колір стін, висота стелі, покриття підлоги й ін. У цій теорії такий абстрактний образ називається фреймом.

Визначення фрейму за Мінскі

Фреймом називається мінімально можливий опис певної сутності, такий, що подальше скорочення цього опису приводить до втрати цієї сутності.

Фреймом називається також і формалізована модель для відображення образу.

В якості ідентифікатора фрейму слугує ім'я фрейму, яке має бути унікальним у всій фреймовій системі.

Фрейм має певну внутрішню структуру, що складається з множини елементів (слотів), яким також привласнюються імена. За слотами слідують шпації, в які поміщають дані, що представляють поточні значення слотів. Кожен слот у свою чергу представляється визначеною структурою даних. В значення слота підставляється конкретна інформація, що відноситься до об'єкта, який описується цим фреймом.

Структуру фрейму можна представити так:

Ім'я фрейму:

(Ім'я 1-го слота: значення 1-го слота),

(Ім'я 2-го слота: значення 2-го слота),

- - - -

(Ім'я N-го слота: значення N-го слота).

Цей запис можна представити у вигляді таблиці, доповнивши двома стовпцями.

Структура фрейму

Ім'я слота	значення слота	спосіб отримання значення	приєднана процедура

У таблиці додаткові стовпці призначені для опису способу отримання слотом його значення і можливого приєднання до того чи іншого слоту спеціальних процедур, що допускається в теорії фреймів. Як значення слота може виступати ім'я іншої фрейма; так утворюють мережі фреймів.

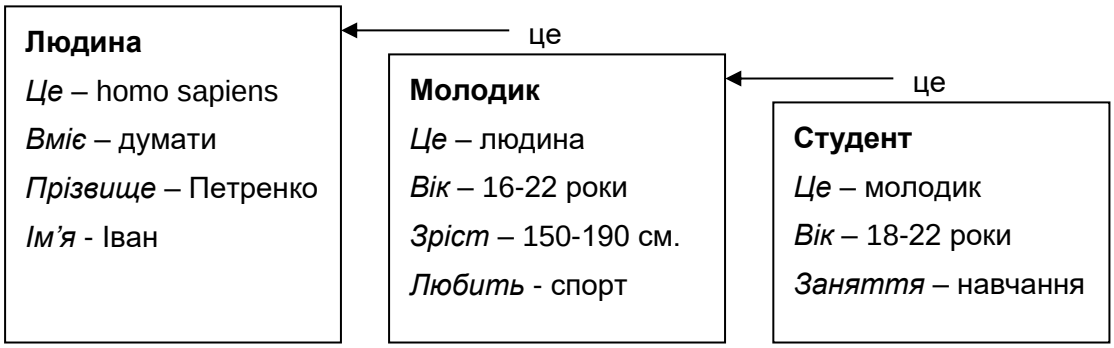
Способи отримання значення у фреймі-екземплярі

- **За замовчуванням** від фрейма-зразка
- **Через спадкування властивостей** від фрейму, що вказаний у слоті «це»
- **За формулою**, що вказано у слоті
- **Через під'єднану процедуру**
- **Явно** з діалогу з користувачем
- **З бази даних**

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

Фрейми-зразки описують узагальнені поняття (класи) однотипних об'єктів з однаковими характеристиками.

Конкретні об'єкти називають **екземплярами фреймів**. Опис екземплярів фрейму формується внаслідок конкретизації фреймів, тобто під час заповнення слотів конкретними значеннями. Якщо при описі фрейма-зразка заповнити певні слоти конкретними значеннями, вони передаються до всіх фреймів-екземлярів.



Для фреймових моделей характерною є ієрархія понять та успадкування властивостей. «Студент» є похідною від «молодик» та «людина», тому слоти «прізвище» та «ім'я» можна не задавати, а задати лише значення для специфічних слотів.

Слот «це» (AKO – a kind of) вказує на фрейм вищого рівня ієрархії, звідки спадкуються значення слотів. Наприклад, на питання чи любить студент спорт, буде відповідь «так», бо це притаманно всім молодикам. Успадкування властивостей може бути частковим, наприклад, «вік» для студента не успадковується, оскільки він вказується явно у власному фреймі.

Модель фрейму є універсальною, бо надає можливість відобразити різноманіття знань про предметну область.

Типи фреймів

Модель фрейма є досить універсальною, оскільки дозволяє відобразити все різноманіття знань про світ:

- **Фрейми-структури** використовують для позначення об’єктів та понять (університет, школа, аудиторія)
- **Фрейми-ролі** (викладач, студент, декан)
- **Фрейми-сценарії** (лекція, іспит, захист диплому)
- **Фрейми-ситуації** (революція, раптова перевірка присутності студентів)

Найважливішою властивістю теорії фреймів є запозичене з теорії семантичних мереж спадкування властивостей. І у фреймах, і в семантичних мережах спадкування відбувається за **АКО-зв'язками** (A-Kind-Of = це). Слот АКО вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Значенням слоту може бути практично що завгодно: числа, формули, тексти на природній мові або програми, правила виведення або посилання на інші слоти даного фрейму або інших фреймів. В якості значення слота може виступати набір слотів більш низького рівня, що дозволяє реалізовувати у фреймових представленнях «принцип матрьошки». Зв'язки між фреймами задаються значеннями спеціального слота з іменем «Зв'язок».

У загальному випадку структура даних фрейму може містити більш широкий набір інформації, до якого входять наступні атрибути.

Ім'я фрейму. Воно служить для ідентифікації фрейму в системі і має бути унікальним. Фрейм являє собою сукупність слотів, число яких може бути довільним. Число слотів в кожному фреймі встановлюється проектувальником системи, при цьому частина слотів визначається самою системою для виконання специфічних функцій (системні слоти), прикладами яких є: слот-покажчик батька даного фрейма (IS-A), слот-покажчик дочірніх фреймів, слот для введення імені користувача, слот для введення дати визначення фрейму, слот для введення дати зміни фрейма і т.д.

Ім'я слоту. Воно має бути унікальним в межах фрейму. Зазвичай, ім'я слота є ідентифікатором, який наділено певною семантикою. Як ім'я слота може виступати довільний текст.

Наприклад, <Ім'я слота> = Президент України, <Значення слота> = Володимир Зеленський.

Імена системних слотів, зазвичай, зарезервовані, в різних системах вони можуть мати різні значення. Приклади імен системних слотів: IS-A, HASPART, RELATIONS і т.д. Системні слоти служать для редагування бази знань і управління виводу у фреймовій системі.

Покажчики успадкування. Вони показують, яку інформацію про атрибути слотів з фрейма верхнього рівня успадковують слоти з аналогічними іменами в даному фреймі. Покажчики успадкування характерні для фреймових систем ієрархічного типу, заснованих на відносинах типу «абстрактне - конкретне». В

конкретних системах покажчики успадкування можуть бути організовано в різний спосіб і мати різні позначення:

- **U (Unique)** - значення слота неуспадковується;
- **S (Same)** - значення слота успадковується;
- **R (Range)** - значення слоту повинні знаходитися в межах інтервалу значень, зазначених в однойменному слоті батьківського фрейму;
- **O (Override)** - при відсутності значення в поточному слоті воно успадковується з фрейма верхнього рівня, однак у випадку визначення значення поточного слота воно може бути унікальним. Цей тип покажчика виконує одночасно функції покажчиків U і S.

Покажчик типу даних. Він показує тип значення слота. Найбільш вживані типи: **frame** - покажчик на фрейм; **real** - дійсне число; **integer** - ціле число; **boolean** - логічний тип; **text** - фрагмент тексту; **list** - список; **table** - таблиця; **expression** - вираження; **lisp** - пов'язана процедура тощо.

Значення слоту. Воно повинно відповідати зазначеному типу даних і умові успадкування.

Демони. Демоном називається процедура, яка автоматично запускається при виконанні деякої умови. Демони автоматично запускаються при зверненні до відповідного слоту. Типи демонів пов'язані з умовою запуску процедури. Демон з умовою IF-NEEDED запускається, якщо в момент звернення до слоту його значення не було встановлено. Демон типу IF-ADDED запускається при спробі зміни значення слота. Демон IF-REMOVED запускається при спробі видалення значення слота. Можливі також інші типи демонів. Демон є різновидом пов'язаної процедури.

Приєднана процедура. Як значення слота може використовуватися процедура, що називається службовою в мові ЛІСП або методом у мовах об'єктно-орієнтованого програмування. Приєднана процедура запускається за повідомленням, яке передається з іншого фрейму. Демони і приєднані процедури є процедурними знаннями, об'єднаними разом з декларативними в єдину систему. Ці процедурні знання є засобами управління виводу у фреймових системах, причому з їх допомогою можна реалізувати будь-який механізм виведення. Подання таких знань і заповнення ними інтелектуальних систем - дуже нелегка справа, яка вимагає додаткових витрат праці і часу розробників. Тому, проектування фреймових систем виконується, як правило, фахівцями, які мають високий рівень кваліфікації в галузі штучного інтелекту.

Частина фахівців з систем штучного інтелекту вважають, що немає необхідності виділяти фреймові моделі представлення знань, так як в них об'єднані всі основні особливості моделей інших типів.

Наведемо кілька прикладів фреймових описів.

Приклад 1. Фрейм, що описує людину.

Фрейм:	***Людина***
Ім'я слоту:	Значення слоту
Клас:	Тварина
Структурний елемент:	Голова, шия, руки, ...

Зріст:	50/ 220 см
Маса:	3/ 200 кг
Хвіст:	Ні
Мова:	Українська, російська, англійська, ...
Зв'язок:	Мавпа

Приклад 2. Опис за допомогою фреймів поняття письмового звіту.

У вигляді семантичної мережі «звіт» можна представити таким чином.

Рис. 1. Семантична мережа поняття «звіт»

Представлення поняття «звіт» за допомогою фреймів.

Рис. 2. Подання поняття «звіт» у вигляді фреймів

Приклад 3. Специфікації фреймів на продукційно-фреймовій мові опису знань PILOT / 2.

Фрейми John і Mary - екземпляри фрейма-прототипа Person.

[**Person is_a prototype**;
Name string, if_changed ask_why ();
Age int, restr_by> = 0;
Sex string, restr_by (== "male" || == "female"),
by_default "male";
Children {frame}];

[**John is_a Person**; if_deleted bury ();
Name = "Johnson";
Age = 32;
Children = {Ann, Tom}];

[**Mary is_a Person**;
without Age;
Name = "Smirnova";
Sex = "female";
Children = empty];

Основною перевагою фреймів як моделі представлення знань є те, що вона відображає концептуальну основу організації пам’яті людини, а також її гнучкість і наочність. Найбільш яскраво переваги фреймових систем представлення знань проявляються в тому випадку, якщо родовидові зв'язки змінюються нечасто і предметна область налічує небагато винятків.

У фреймових системах дані про родовидові зв'язки зберігаються явно, як і знання інших типів. Значення слотів представляються в системі в єдиному екземплярі, оскільки містяться тільки в одному фреймі, що описує найбільш поняття з усіх тих, які містить слот з даним ім'ям. Така властивість систем фреймів забезпечує економне розміщення бази знань у пам'яті комп’ютера.

Ще одна перевага фреймів полягає в тому, що значення любого слоту може бути обчислено за допомогою відповідних процедур або знайдено евристичними методами. Тобто, фрейми дозволяють маніпулювати як декларативними, так і процедурними знаннями.

До недоліків фреймових систем відносять їх відносно високу складність, що проявляється у зниженні швидкості роботи механізму виведення і збільшення трудомісткості внесення змін до родової ієрархії. Тому, при розробці фреймових систем приділяють наочним способам відображення і ефективним засобам редагування фреймових структур.

Спеціальні мови подання знань в мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова оболонка Kappa, PILOT / 2 і інші програмні засоби дозволяють ефективно будувати промислові системи.

В останні роки термін «фреймовий» часто замінюють терміном «об'єктно-орієнтований». Цей підхід є розвитком фреймового подання. *Шаблон фрейму можна розглядати як клас, екземпляр фрейму - як об'єкт*. Мови об'єктно-орієнтованого програмування (ООП) надають засоби створення класів і об'єктів, а також засоби для опису процедур обробки об'єктів (методи). Мови ООП, що не містять засобів реалізації приєднаних процедур, не дозволяють організувати гнучкий механізм логічного висновку, тому розроблені на них програми або представляють собою об'єктно-орієнтовані бази даних, або вимагають інтеграції з іншими засобами обробки знань (наприклад, з мовою PROLOG).

Об'єктно-орієнтована методологія подання знань реалізована в системах G2, RTWorks.

4. Виведення на знаннях

На сьогодні, не дивлячись на недоліки та обмеження, поширеними і популярними є інтелектуальні системи з продукційною моделлю представлення знань.

База даних складається з набору правил «Якщо А, тоді В». Програма, що керує перебором правил називається машиною виведення або інтерпретатором правил. Вона складається з 2 компонент і відповідно виконує 2 функції.

- **Компонента виведення** реалізує власне виведення, тобто перегляд існуючих фактів з БД і правил з БЗ, а також доручення до БД нових фактів.
- **Керуюча компонента**, що керує процесом. Вона визначає порядок перегляду та застосування правил, а також працює в режимі консультації. Ця компонента зберігає для користувача інформацію про отримані висновки і потребує нових даних, якби їх бракувало для обробки чергового правила.

Компонента виведення діє згідно правила:

«Якщо відомо, що твердження А є вірним і існує правило типу *Якщо А, тоді В*, тоді твердження В є також вірним»

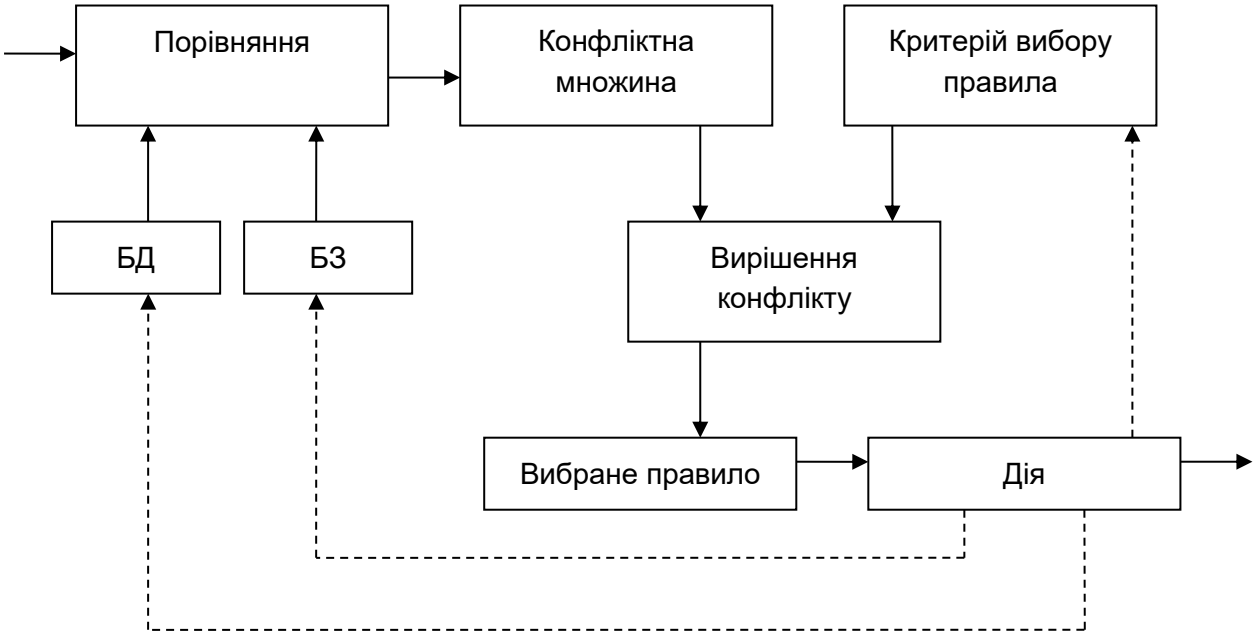
Правила спрацьовують, коли знаходяться факти, що задовольняють умові. Тобто, якщо умова є вірною, тоді висновок є вірним.

Компонента виведення повинна функціонувати, навіть за невеликої кількості фактів. Отриманий висновок може бути і неточним, однак, система не повинна із-за цього зупинятися.

Керуюча компонента визначає порядок застосування правил з БЗ і виконує 4 операції.

1. **Порівняння**. Умова правила порівнюється з існуючими фактами
2. **Вибір (вирішення конфлікту).** Якщо в конкретній ситуації можна застосувати кілька правил, то з них вибирається одне, що найкраще задовольняє заданому критерію.
3. **Спрацювання.** Якщо умова правила при порівнянні збігається з фактами з БД, тоді правило спрацьовує.
4. **Дія.** Якщо в правій частині правила міститься кінцева дія, то вона виконується. Якщо дія є проміжною, тоді вона долучається до БД як новий факт

Машина виведення працює циклічно. В кожному циклі вона переглядає всі правила та існуючі на той момент факти.



Цикл роботи машини виведення

Факти з БД послідовно порівнюють з умовами правил для виявлення успішного порівняння. Сукупність вибраних правил складає конфліктну множину. Для вирішення конфлікту машина виведення має критерій, за яким вибирається лише одне правило. Факти, що утворюють висновок правила заносяться до БД або змінюють критерій вибору конфліктних правил. Якщо у висновку правила міститься кінцева дія, то вона виконується.

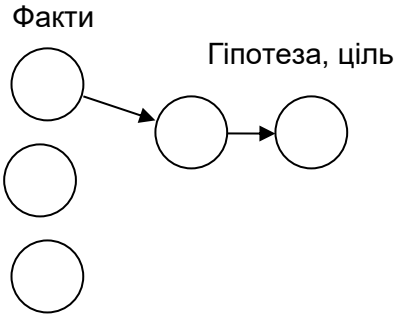
Робота машини виведення залежить від стану БД та БЗ. На практиці враховується поведінка машини виведення у попередніх циклах. Інформація про поведінку машини виведення (протокол системи) запам’ятовується у просторі станів.

Стратегії керування виведенням

Порядок застосування і спрацювання правил залежить від стратегії виведення, тобто напрямку пошуку та методу перебору. Процедури пошуку зашиваються в механізм виведення, і в подальшому не змінюються.

При розробці стратегії виведення слід визначити:

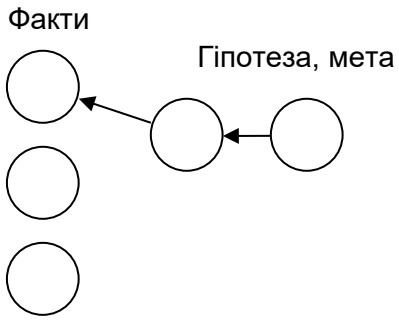
- Яку точку в просторі станів прийняти за початку. Від цього залежить напрямок пошуку – прямий чи зворотний.
- Обрати метод перебору: в глибину, в ширину, розбиття на під задачі та інше.



Пряме виведення (виведення, що керується даними)

В системах з прямим виведенням по відомих фактах шукається гіпотеза, що витікає з цих фактів. Знайдена мета (ціль) заноситься до БД як новий факт.

Зворотне виведення (виведення, що керується цілями)



При зворотному виведенні спочатку висувається гіпотеза, потім машина виведення вертається назад для знаходження тих фактів, що підтверджують цю гіпотезу. Якщо вона є вірною, то обирається наступна гіпотеза, що деталізує першу і є по відношенню до неї підпорядкованою. Знаходять факти, що підтверджують підпорядковану гіпотезу і т.д.

Приклад

Є фрагмент БЗ, що має 2 правила

Правило 1. Якщо «відпочинок влітку» і «людина активна», тоді «їхати в гори»

Правило 2. Якщо «любить сонце», тоді «відпочинок влітку»

В систему надійшли факти «людина активна» і «любить сонце»

Пряме виведення

Маємо факти, потрібно отримати рекомендації

1 прохід

Пробуємо Правило 1, воно не спрацьовує (бракує факта «відпочинок влітку»)

Пробуємо Правило 2, воно спрацьовує, до БД долучається факт «відпочинок влітку»

2 прохід

Пробуємо Правило 1, воно спрацьовує, активізується мета «їхати в гори», що й буде виведено системою як рекомендація.

Зворотне виведення

Потрібно підтвердити вибрану мету за допомогою існуючих правил і фактів

1 прохід

Мета «їхати в гори». Пробуємо Правило 1 – бракує даних «відпочинок влітку». Цей факт стає новою підпорядкованою гіпотезою, яку слід підтвердити.

Мета «відпочинок влітку». Правило 2 підтверджує цю під гіпотезу.

2 прохід

Пробуємо Правило 1 – мета підтверджується

Методи перебору

Стратегії керування виведенням застосовують в системах, бази знань яких налічують сотні і тисячі правил, для мінімізації часу пошуку та підвищення ефективності виведення:

- Перебір в ширину
- Перебір в глибину
- Розбивання на під задачі
- Альфа-бета алгоритм та інші

При пошуку в глибину за наступну під гіпотезу обирається та, яка відповідає наступному, детальнішому рівню опису задачі. Наприклад, класифікація тварин.

При пошуку в ширину система спочатку аналізує всі симптоми, що знаходяться на одному рівні простору станів, потім переходить до симптомів наступного рівня деталізації. Наприклад, діагностика захворювань.

Розбивання на під задачі. Виділяються підзадачі, вирішення яких розглядаються як досягнення проміжних цілей на шляху до кінцевої мети. Наприклад, пошук несправностей в комп'ютері – спочатку виявляють несправну підсистему (живлення, пам'ять тощо), що звужує простір пошуку.

Якщо правильно зрозуміти сутність задачі і оптимально розбити її на систему ієрархічно зв'язаних «цілей - підцілей», то можна досягнути мінімального шляху в просторі пошуку до її вирішення

Альфа-бета алгоритм дозволяє зменшити простір станів шляхом забирання гілок, що є неперспективними для пошуку. Переглядаються лише ті вершини, до яких можна перейти в результаті наступного кроку, всі решта неперспективні напрямки забираються.

Альфа-бета алгоритм широко застосовується в системах, що орієнтовані на ігри (шахи, шашки).