Лекції №3-4.

Моделі подання знань. Мережеві моделі: фрейми, семантичні мережі

ПРЕДСТАВЛЕННЯ ЗНАНЬ

Представлення знань і пошук рішень утворюють ядро штучного інтелекту. Зосередимо свою увагу на основних моделях представлення знань.

Поняття “знання” трактується окремими авторами по-різному і багато в чому носить дискусійний характер.

Модель представлення знань повинна відбивати істотні характеристики задачі, що розв’язується і забезпечувати відповідною інформацією процедури, що виконують пошук рішень. При цьому вона повинна володіти необхідною виразною здатністю, щоб відбивати всі цікавлячі деталі предметної області задачі, а також бути досить ефективною з точки зору пошуку рішень, що розглядають як вивід на знаннях. Виразна сила й обчислювальна ефективність – дві основні характеристики моделей представлення знань. Багато моделей представлення знань, що володіють великими виразними можливостями, не піддаються ефективній реалізації. Тому пошук оптимального співвідношення між виразною силою моделі представлення знань, що використовується і ефективністю її реалізації – основна задача розроблювача СШІ.

Вивчення моделей представлення знань починається з логічної моделі, що базується на численні предикатів. Така модель характеризується добре визначеною семантикою і формально обґрунтованими правилами вивід в. Використовуючи логіку, вводяться основні поняття представлення, знань, і демонструється взаємозв'язок представлення знань з пошуком рішень (виводом).

Крім логічних моделей, у США широко застосовують такі моделі представлення знань, як продукційні системи, семантичні мережі, фрейми.

3.1. Дані і знання

При вивченні інтелектуальних систем традиційно виникає питання що ж таке знання і чим вони відрізняються від звичайних даних, що десятиліттями обробляються ЕОМ. Можна запропонувати декілька робочих визначень, в рамках яких це стає очевидним.

Дані – це окремі факти, що характеризують об'єкти, процеси і явища предметної області, а також їх властивості.

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

1. D1 – дані як результат вимірювань і спостережень;
2. D2 – дані на матеріальних носіях інформації (таблиці, протоколи, довідники);
3. D3 – моделі (структури) даних у вигляді діаграм, графіків, функцій;
4. D4 – дані в комп'ютері на мові опису даних;
5. D5 – бази даних на машинних носіях інформації.

Знання засновані на даних, отриманих емпіричним шляхом. Вони являють собою результат розумової діяльності людини, направленої на узагальнення її досвіду, отриманого внаслідок практичної діяльності.

Знання – це закономірності предметної області (принципи, зв'язки, закони), отримані внаслідок практичної діяльності і професійного досвіду, що дозволяють фахівцям ставити і вирішувати задачі в цій області.

При обробці на ЕОМ знання трансформуються аналогічно до даних.

1. Z1 – знання в пам'яті людини як результат мислення;
2. Z2 – матеріальні носії знань (підручники, методичні допомоги);
3. Z3 – *поле знань* умовний опис основних об'єктів предметної області, їх атрибутів і закономірностей, що їх пов'язують;
4. Z4 – знання, описані на мовах представлення знань (продукційні мови, семантичні мережі, фрейми);
5. Z5 – *база знань* на машинних носіях інформації.

Часто використовується таке визначення знань.

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

Існує багато способів визначати поняття. Один з способів, що широко застосовуються заснований на ідеї інтенсіоналу. *Інтенсіонал* поняття – це визначення його через співвіднесення з поняттям вищого рівня абстракції з вказівкою специфічних властивостей. Інтенсіонали формулюють знання про об'єкти. Інший спосіб визначає поняття через співвіднесення з поняттями нижчого рівня абстракції або перелік фактів, які відносяться до об'єкта, що визначається. Це є визначення через дані, або *екстенсіонал* поняття.

Приклад 1.1.

Поняття "персональний комп'ютер". Його інтенсіонал: "Персональний комп'ютер це дружня ЕОМ, яку можна поставити на стіл і купити менш ніж за $1000".

Екстенсіонал цього поняття: "Персональний комп'ютер це Mac, IBM PC, Sincler…".

Для зберігання даних використовуються бази даних (для них характерні великий об’єм і відносно невелика питома вартість інформації), для зберігання знань – бази знань (невеликого об’єму, але виключно дорогі інформаційні масиви). *База знань* – основа будь-якої інтелектуальної системи. Знання можуть бути класифіковані за наступними категоріями:

- *Поверхневі* – знання про видимі взаємозв’язки між окремими подіями і фактами в предметній області.
- *Глибинні* – абстракції, аналогії, схеми, що відображають структуру і природу процесів, які протікають в предметній області. Ці знання пояснюють явища і можуть використовуватися для прогнозування поведінки об’єктів.

Приклад 1.2

Поверхневі знання:
“Якщо натиснути на кнопку дзвінка, роздасться звук. Якщо болить голова, то потрібно прийняти аспірин”.

Глибинні знання:
“Принципова електрична схема дзвінка і проводки. Знання фізіологів і лікарів високої кваліфікації про причини, види головних болів і методи їх лікування”.

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

Крім того, в підручниках зі ШІ знання традиційно ділять на *процедурні* і *декларативні.* Історично первинними були процедурні знання, тобто знання, “розчинені” в алгоритмах. Вони управляли даними. Для їх зміни потрібно було змінювати програми. Однак з розвитком штучного інтелекту пріоритет даних поступово змінювався, і все більша частина знань зосереджувалася в структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних знань.

Сьогодні знання набули чисто декларативної форми, тобто знаннями вважаються пропозиції, записані на мовах представлення знань, наближених до природної і зрозумілих нефахівцям.

3.2. Знання і їхнє представлення в СШІ

Що ж являють собою знання? Чітка відповідь на це питання дана в енциклопедичному словнику, Знання – це “перевірений практикою результат пізнання дійсності, вірне її відображення в мисленні людини”. Відображення існуючого світу в мисленні людини зв’язано з процесом абстрагування, що полягає у вигляділенні найбільш істотних властивостей і ознак чи явищ об’єктів, що спостерігаються в навколишній дійсності, і представлень так у такій спрощеній формі, що необхідна для логічних висновків і прийняття ефективних рішень у всіляких ситуаціях. Зазначене спрощене представлення дійсності називають моделлю. Людина, пізнаючи навколишній світ, будує моделі явищ, процесів, предметів і т.п. – об’єктів реального світу. Модель є інформаційним еквівалентом частини реального світу (предметної області) і лежить в основі процесу пізнання. В інженерній практиці саме моделі асоціюються зі знанням. Це знання, на основі, яких можна ефективно використовувати об’єкти реального світу для досягнення визначених цілей. Але звідси не випливає, що ці терміни еквівалентні.

Поняття ”знання” ширше поняття “модель”. Хоча, безумовно, знання в основному складаються з моделей. Знання містять у собі не тільки ті чи інші зведення про об’єкти реального світу, але й інформацію про механізми вивід в нових звань на підставі наявних. В даний час у штучному інтелекті немає формального єдиного визначення поняття “знання”. Однак багатьма дослідниками визнається, що знання – складноорганізовані дані, збережені в пам’яті СШІ і відомості про об’єкти і відносини предметної області, процесах взаємодій об’єктів у часі й у просторі, правилах здійснення логічного виведення.

Важливим елементом цього визначення є вказівка на те, що знання – це інформація, в основі якої виконується логічний вивід.

Знання характеризуються послідовністю властивостей, що відрізняють їх від традиційних моделей даних. Перелічимо ці властивості.

Внутрішня інтерпритованість. При збереженні знань у пам’яті СШІ, поряд із традиційними елементами даних, зберігаються й інформаційні структури; що дозволяють інтерпретувати вміст відповідних комірок пам’яті.

Структурованість. Знання складаються з окремих інформаційних одиниць, між якими можна встановити класифікаційні стосунки: рід – вид, клас – елемент, тип – підтип, частина – ціле і т.п.

Зв’язність. Між інформаційними одиницями передбачаються зв’язки різного типу: причина – наслідок, одночасно, бути поруч і ін. Дані зв’язки визначають семантику і прагматику предметної області.

Семантична метрика. На множини інформаційних одиниць, збережених у пам’яті, вводяться деякі шкали, що дозволяють оцінити їхню семантичну близькість. Це дозволяє знаходити в інформаційній базі знання, близькі до вже знайденого.

Активність. За допомогою даної властивості підкреслюється принципова відмінність знань від даних. Виконання тих чи інших дій у СШІ ініціюється станом бази знань; При цьому передбачається, що поява нових фактів і зв’язків може активізувати систему.

Центральним питанням побудови систем, заснованих на знаннях, є вибір форми представлення знань. Представлення знань – це спосіб формального вираження знань про предметну область у комп'ютерно-інтепритованій формі. Відповідні формалізми, що забезпечують зазначене представлення, називають моделями представлення знань.

Моделі представлення знань можна умовно розділити на декларативні і процедурні. У декларативних моделях знання представляються у вигляді описів об'єктів і відносин між об'єктами без вказівки в явному вигляді, як ці знання обробляти. Такі моделі припускають відділення описів (декларацій) інформаційних структур від механізму вивід в, що оперує цими структурами. У процедурних моделях значення представляються алгоритмами (процедурами), що містять необхідний опис інформаційних елементів і одночасно визначають способи їх обробки.

Конкретні моделі, застосовувані на практиці, являють собою комбінацію декларативних і процедурних представлень. Найбільш розповсюдженими є наступні моделі представлення знань:

- логічні моделі;
- продукційні моделі;
- мережні моделі;
- фреймові моделі.

Логічні моделі реалізуються засобами логіки предикатів. У цьому випадку знання про предметну область представляються у вигляді сукупності логічних формул. Тотожні перетворення формул дозволяють одержувати нові знання. Перевагами логічних моделей представлення знань є наявність чіткого синтаксису і широко прийнятої формальної семантики, а також теоретично обґрунтованих процедур автоматичного вивід в. Основним недоліком даних моделей є неможливість одержання висновків в областях, де вимагаються правдоподібні висновки, коли результат виходить з визначеною оцінкою впевненості в його істинності. Крім цього, такі моделі характеризуються монотонним характером вивід в, тобто в базу знань додаються тільки істинні твердження, що виключає можливість протиріч. На практиці часто зустрічаються немонотонні міркування, що важко реалізувати в рамках логічної моделі.

Проте, логічні моделі виступають у якості теоретичної основи опису самої системи представлення знань і поступово розширюють свої можливості. Тому надалі цим моделям приділяється значна увага.

У продукційних моделях знання представляються набором правил виду “Якщо А, то В”, де умова правила А є твердженням про вміст бази фактів, а наслідок В говорить про те, що треба робити, коли дане продукційне правило активізоване.

Продукційні моделі представлення знань завдяки природній модульності правил, наочності і простоті їхнього створення широко застосовуються в інтелектуальних системах.

Семантичні мережі є винятковим випадком мережних моделей представлення знань. Формально мережні моделі задаються у вигляді

$H = <I, C_1, C_2, \ldots, C_n, Q>$,

де І-множину інформаційних елементів, що зберігаються у вузлах мережі; $C_1, C_2, \ldots, C_n$ – типи зв’язків між інформаційними елементами; Q -відображення, що встановлює відповідність між множиною типів зв’язків і множиною інформаційних елементів мережі.

Мережні моделі представлення знань розрізняються між собою типами зв'язків (відносин), що використовуються. Якщо в мережі використовуються ієрархічні зв'язки (клас-підклас, рід-вид і т.п.), то мережа називається класифікованою. Якщо зв'язки між інформаційними елементами представляються функціональними відносинами, що дозволяють обчислювати значення одних інформаційних елементів за значеннями інших, то мережі називають функціональними (обчислювальними). Якщо в мережі допускаються зв'язки різного типу, то її називають семантичною мережею.

Семантична мережа являє собою спрямований граф, в якому вершинам відповідають об'єкти (сутності) предметної області, а дугам – відносини, у яких знаходяться ці об'єкти. Вивід в семантичних мережах може виконуватися на основі алгоритмів співставлення, шляхом виграділення підграфів з визначеними властивостями.

До переваг семантичних мереж відносять: велику виразну здатність, наочність графічного представлення, схожість структури мережі до семантичної структури фраз природної мови. Недоліком представлення знань у вигляді семантичних мереж є відсутність єдиної термінології.

Дана модель представлення знань знаходить різне втілення в різних дослідників.

Фреймові моделі представлення знань використовують теорію організації пам'яті, розуміння і навчання, запропоновану М.Мінським. Фрейм (від англ. frame – рамка, каркас, кістяк) – структура даних, призначена для представлення стереотипних ситуацій. Фрейм складається зі слотів (slot – гніздо, щілинка, паз). Значенням слота можуть бути числа, вираження, тексти, програми, посилання на інші фрейми. Сукупності фреймів утворюють ієрархічні структури, побудовані на родовидових ознаках, що дозволяє успадковувати значення слотів. Така властивість фреймів забезпечує ощадливе розміщення бази знань у пам'яті. Крім цього, значення слотів можуть обчислюватися за допомогою різних процедур, тобто фрейми комбінують у собі декларативні і процедурні представлення знань. Фреймові моделі можна розуміти як мережні моделі уявлення знань, коли фрагмент мережі представляється фреймом з відповідними слотами і значеннями. З фреймовими моделями зв'язані моделі представлення знань на основі сценаріїв і об'єктів.

Знання, якими оперує людина, часто мають якісну природу, характеризуються неповнотою, неточністю, нечіткістю.

3.3. Продукційні моделі

3.3.1. Продукційні системи

Продукційна модель представлення знань є однією з найпоширеніших . Представлення знань за допомогою правил-продукцій має в деяких відносинах подібність із правилами виводу логічних моделей. Це дозволяє за допомогою продукцій виконувати ефективний вивід і, крім того, завдяки природній аналогії процесу міркувань людини дані моделі наочніше відбивають знання.

У системах продукций знання представляються за допомогою наборів правил виду: “якщо А, то В”. Тут А і В можуть розумітися як “ситуація – дія”, “причина – наслідок”, “умова – вивід” і т.п. Часто правило-продукцію записують з використанням знака логічного проходження:

$$A \Rightarrow B$$

.

Однак не слід ототожнювати правило-продукцію і відношення логічного проходження. Справа в тім, що інтерпретація продукції залежить від того, що розташовано ліворуч і праворуч від знака логічного проходження. Часто під А розуміється деяка інформаційна структура (наприклад, фрейм), а під У – деяка дія, що полягає в її трансформації (перетворенні). Логічна інтерпретація розглянутого вираження накладає тверді обмеження на зміст символів А і В, які повинні розглядатися як ППФ. Тому поняття продукції ширше логічного проходження. Як приклад розглянемо правило, взяте з бази знань експертної системи MYCIN, призначеної для діагностики інфекційних захворювань.

Якщо

1. місце виділення культури – кров, і
2. реакція мікроорганізму – грамвід'ємна, і
3. форма мікроорганізму – паличка, і
4. пацієнт відноситься до групи ризику,

то

5. із упевненістю (0,6) назва мікроорганізму – pseudomonia aeruginosa.

Умова правила складається з чотирьох фактів, з'єднаних союзом "і". Якщо всі чотири факти мають місце, то вірним буде наслідок правила. З кожним правилом зв'язується деяке число, що приймає значення в діапазоні від -1 до 1, що виражає ступінь вірогідності наслідку і називало коефіцієнтом упевненості. Це означає, що система працює з неточними і ненадійними фактами.

У базі знань системи MYCIN факти представляються за допомогою триплету: об’єкт-атрибут-значення. Так, розглянуте, вище правило містить п’ять фактів, які можна представити у таблиці 3.1

Таблиця 3.1 – Представлення фактів

№ факту	Об’єкт	Атрибут	Значення
1)	Культура	Місце	Кров
2)	Мікроорганізм	Реакція	
3)	Мікроорганізм	Форма	Паличка
4)	Пацієнт	Належить до групи ризику	Істина
5)	Мікроорганізм	назва	Pseudomonia aeruginosa

Якщо в процесі активізації правила перші чотири факти виявляться істинними, то в базу знань буде поміщений новий факт, що представляється триплетом об’єкт-атрибут-значення. Однією з переваг збереження фактів у вигляді триплетів є підвищення ефективності процедур пошуку в базі знань, тому що з’являється можливість упорядкування фактів по об’єктам і атрибутам.

Розглянута форма запису правило-продукції є спрощеною і представляє лише ядро продукції. У багатьох випадках правило-продукцію записують в узагальненій формі [21,4,5]

$$Rnj : (\Pr, Bc, A \Rightarrow B, Ac)$$

, (3.4)

де Rnj – ідентифікатор j-продукції в n-наборі продукций; Pr -пріоритет правила продукції; Bc – передумова застосування ядра продукції, що представляє предикат, при виконанні якого активізується ядро продукції; Ac – післяумова продукції, що визначає дії і процедури, які необхідно виконати після виконання ядра продукції.

У загальному випадку продукціонна система включає наступні компоненти:

– базу продукційних правил;

– базу даних (робочу пам’ять);

– інтерпретатор.

Множину продукційних правил утворить базу правил, кожне з який представляє відособлений фрагмент знань про розв’язувану проблему. Психологи називають такі фрагменти чанками (від англ. chunk). Вважається, що чан – це об’єктивно існуюча одиниця знань, що вигляділяється людиною в процесі пізнання навколишнього світу.

Передумова правила часто розглядається як зразок. Це деяка інформаційна структура, що визначає узагальнену ситуацію (умова, стан і т.п.) навколишньої дійсності, при якій активізується правило. Робоча пам’ять (глобальна база даних) відбиває конкретні ситуації (стану, умови), що виникають у зовнішньому середовищі.

Інформаційна структура, що представляє конкретну ситуацію зовнішнього середовища в робочій пам’яті, називається образом.

Інтерпретатор реалізує логічний вивід. Процес виводу є циклічним і називається пошуком за зразком. Розглянемо його в спрощеній формі. Поточний стан предметної області, що моделюється відбивається в робочій пам’яті у вигляді сукупності образів, кожний з яких представляється за допомогою фактів. Робоча пам’ять ініціалізується фактами, що описують задачу. Потім вибираються ті правила, для яких зразки, що представляються передумовами правил, порівнянні з образами в робочій пам’яті. Данні правила утворюють конфліктну множину. Усі правила, що входять у конфліктну множину можуть бути активізовані. Відповідно до обраного механізму дозволу конфлікту активізується одне з правил. Виконання дії, що міститься у виводу правила, приводить до зміни стану робочої пам’яті. Надалі цикл керування виведенням повторюється. Зазначений процес завершується, коли не виявиться правил, передумови яких порівнянні з образами робочої пам’яті (Рисунок 3.2).

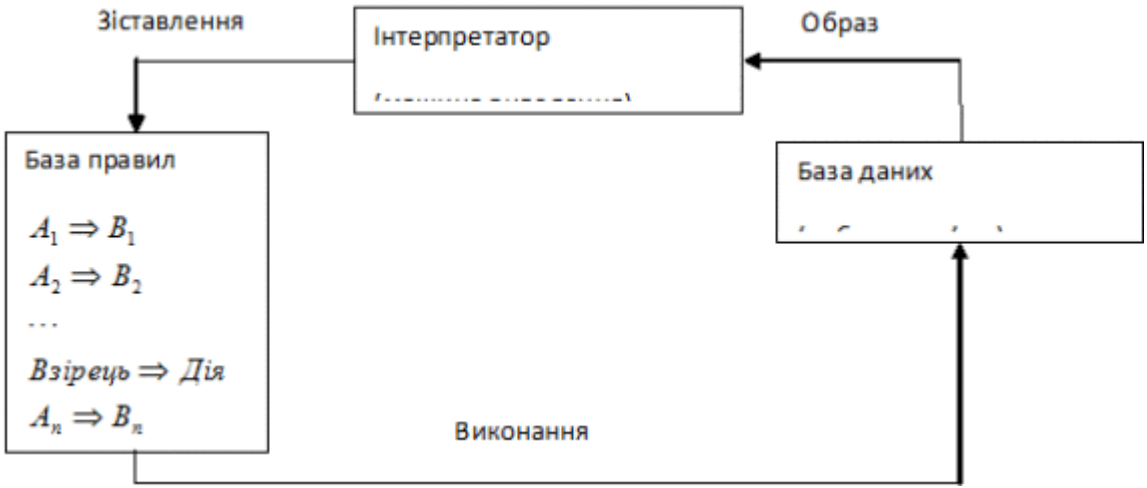

Зіставлення
Інтерпретатор
Образ
База правил
$A_1 \Rightarrow B_1$
$A_2 \Rightarrow B_2$
...
$Взірець \Rightarrow Дія$
$A_n \Rightarrow B_n$
База даних
Виконання

Рисунок 3.2 – Продукційна система

Таким чином, процес виводу, заснований на пошуку за зразком, складається з чотирьох кроків:

– вибір образа;

– співставлення образа зі зразком і формування конфліктного набору правил;

– дозвіл конфліктів;

– виконання правила.

Пояснимо процес функціонування продукційної системи на простому прикладі сортування рядка, що складається з букв a, b і c [77]. Правила сортування представлені на рисунку 3.3. Якщо зразок, заданий передумовою правила, зіставимо з частиною сортованого рядка, то правило активізується. У результаті цього підвираз, що збіглася з умовою правила, заміщається підвиразом із заключної частини правила.

Розглядаючи вихідний і відсортований рядки як початковий і кінцевий стан задачі, а продукційні правила як оператори, що перетворюють один стан в інший, дійдемо виводу, що пошук рішення в продукційних системах відповідає пошуку в просторі станів. На рисунку 3.3 зображене дерево станів задачі. Тут вершини дерева відповідають проміжним станам сортованого рядка, а ребра – правилам-продукциям.

Продукційні системи були запропоновані американським математиком Е. Постом (1943) і розглядалися як модель організації обчислюваного процесу. При цьому правило-продукція трактувалась як оператор заміни одного ланцюжка літер у деякому слові на інший ланцюжок. Систему продукцій можна розглядати як формальну систему з алфавітом A = { a_1, a_2, …, a_n}, кінцевою множинучю аксіом і кінцевою множинучю правил виводу, що мають вигляд

$$P_i : \alpha_i s \Rightarrow s\beta_i, i = 1,2,\ldots,k$$

де α_i, β_i, s – слова алфавіту A. Застосування даного правила до деякого ланцюжка літер, що складається зі слів α_i, і s , означає викреслення α_i, і приписування β_i,. Зазначена заміна відповідає підстановці, що є основним оператором нормальних алгоритмів Маркова [25].

Множина продукцій

1) $ba \Rightarrow ab$

2) $ca \Rightarrow ac$

3) $cb \Rightarrow bc$

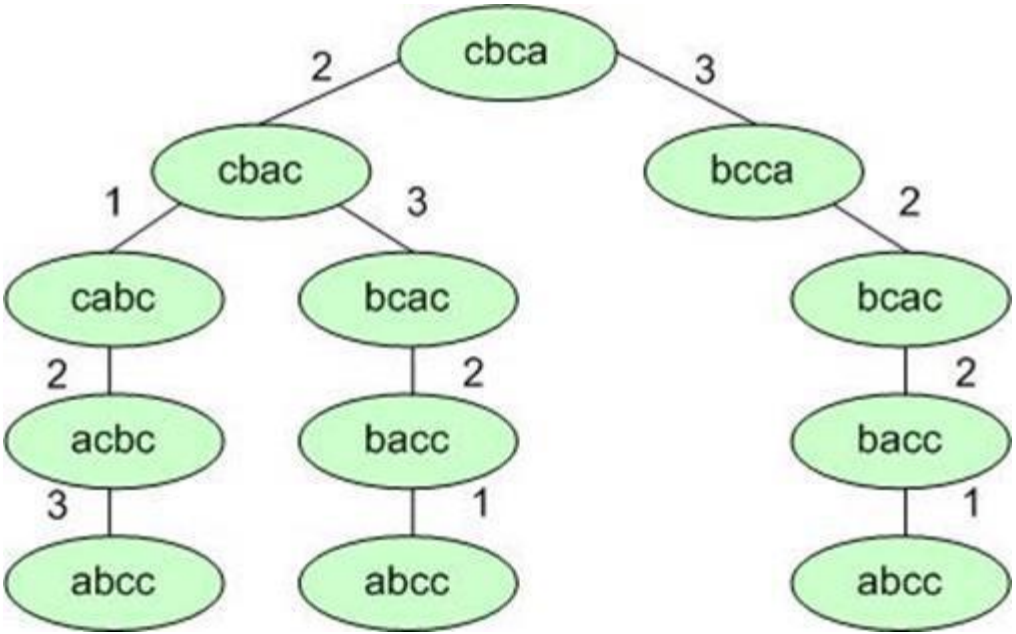


Рисунок 3.3 – Сортування рядка за допомогою правил-продукцій

Цікаві результати були отримані А.Ньюеллом і Г.Саймоном при розробці системи GPS. Вони встановили, що правило-продукції відповідають елементам знань, що накопичуються в довгостроковій пам’яті людини Такі елементи знань активізуються, якщо виникає відповідна ситуація (за зразком), Нові елементи знань можуть накопичуватися в пам’яті без необхідності перезаписування вже існуючих елементів. Робоча пам’ять продукційних систем аналогічна короткочасній пам’яті людини, що утримує в центрі уваги поточну ситуацію. Вміст робочої пам’яті, як правило, не зберігається після рішення задачі. Завдяки зазначеній аналогії, продукційні моделі представлення знань одержали широке поширення в експертних системах, що моделюють рішення задач людиною-експертом у тій чи іншій області.

Широке застосування продукційних моделей визначається наступними основними перевагами:

- універсальністю, практично будь-яка область знань може бути представлена в продукційній формі;
- модульностью, кожна продукція являє собою елемент знань про предметну область, видалення одних і додавання інших продукцій виконується незалежно;
- декларативністю, продукції визначають ситуації, предметної області, а не механізму керування;
- природністю процесу виводу, що багато в чому аналогічний процесу міркувань експерта;
- асинхронностю і природним паралелізмом, що; робить їх дуже перспективними для реалізації на рівнобіжних ЕОМ:

Однак продукційні моделі не свободні від недоліків:

– процес виводу має низьку ефективність, тому що при великому числі продукцій значна частина часу затрачається на невиробничу перевірку умов застосування правил;

– перевірка несуперечності системи продукцій стає дуже складною через недетермінованності вибору виконуваної продукції з конфліктної множини.

3.3.2. Керування виведенням у продукційних системах

У ході рішення проблеми машина виводу (інтерпретатор) виконує дві задачі – власне логічний вивід і керування виведенням. Логічний вивід у продукційних системах не відрізняється особливою складністю і реалізується на основі процедури пошуку по зразку записаному вище.

Керування виведенням у продукційних системах передбачає рішення двох питань:

1) з чого варто починати процес виводу;

2) як надійти, якщо на деякому кроці виводу можливий вибір різних варіантів його продовження.

Відповідь на перше питання приведе до прямого і зворотного ланцюжка роздумів, а на друге питання – до механізмів дозволу конфліктів у продукційних системах.

Як було відзначено вище, застосування процедури пошуку за зразком у продукційних системах відповідає пошуку рішення задач у просторі чи станів простору редукцій задачі. Тому для керування висновками у продукційних системах можуть застосовуватися стратегії сліпого й упорядкованого пошуку, розглянуті в главі 2.

У даному параграфі обмежимося розглядом стратегій прямого і зворотнього виводу (відповідно пошуку, керованого даними і метою), а також основних методів рішення конфліктів.

Прямий і зворотній вивід. Прямий вивід починається з завдання вихідних даних розв'язуваної задачі, що фіксуються у вигляді фактів у робочій пам'яті системи. Правила, що застосовуються до вихідних даних, забезпечують генерацію нових фактів, що додаються в робочу пам'ять.

Процес продовжується; доки не буде отриманий цільовий стан робочої пам'яті.

На рисунку 3.4 представлений приклад пошуку, керованого даними, на множини продукцій, записаних у вигляді формул пропозиційної логіки.

Тут на кожному кроці роботи машини виводу застосовується проста стратегія розв’язання конфліктів: активізується останнє з успішно зіставлених правил. Порядок співставлення правил відповідає їх номерам.

Множина правил-продукцій

1) $G \wedge H \rightarrow C$

б) $C \rightarrow A$

10) $A \rightarrow goal$

2) $I \wedge K \rightarrow D$

7) $D \rightarrow A$

11) $B \rightarrow goal$

3) $L \wedge M \rightarrow E$

8) $E \rightarrow B$

4) $N \rightarrow F$

9) $F \rightarrow B$

5) $O \rightarrow F$

Початкові данні:

START={L, M, N}

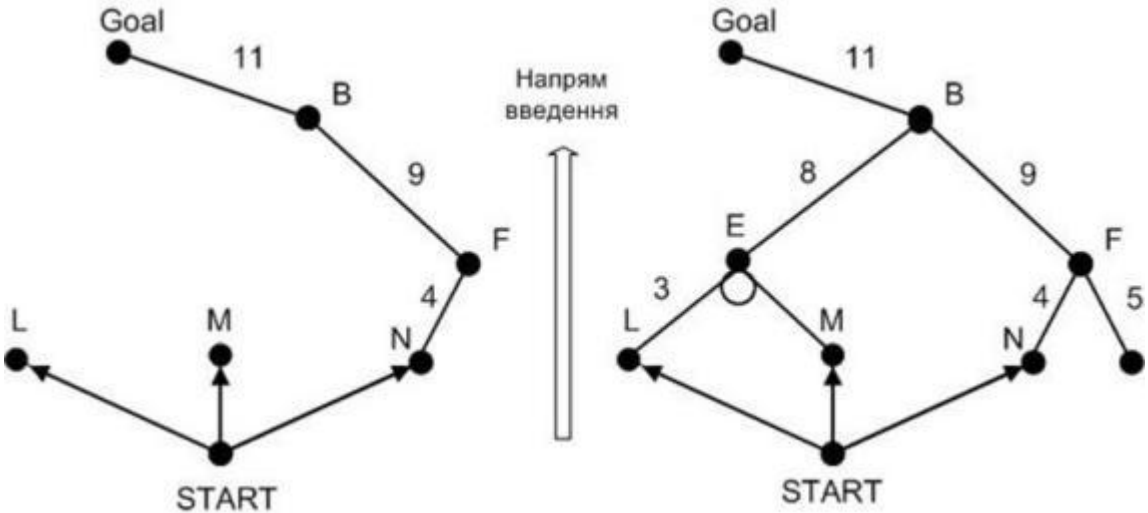
№ кроку	Робоча пам’ять	Конфліктна множина	Активізоване правило
0	L, M, N	–	–
1	L, M, N	3,4	4
2	L, M, N, F	3,9	9
3	L, M, N, F, B	3,11	11
4	L, M, N, F, B, Goal	3	Зупинка

Рисунок 3.4 Прямий вивід у продукційних системах

На рисунку 3.5,а зображений граф рішення. Вершини графа відповідають висловам, а ребра – відповідним правилам введення. Процес виводу починається з розміщення множини вихідних фактів *L*, *M i N* у робочій пам'яті і закінчується, коли в робочу пам'ять буде поміщена цільова вершина *Goal*. Напрямок виводу позначений поруч із графом і відповідає руху від стартової до цільової вершини, тобто від вихідних даних (висловлення *L, M i N*). Тому такий вивід називають виведенням, керованим даними.

Відзначимо, що обрана стратегія активізацій останнього правила, доданого в конфліктну множину, відповідає пошуку в глибину. Саме ця стратегія найбільш часто застосовується в експертних системах продукційного типу, тому що вона відповідає процесу міркувань експерта при рішенні тієї чи іншої задачі. Поводження системи для користувача виглядає досить логічним: він бачить, що система прагне звести кожну мету до підцілі.

Якщо активізувати на кожному кроці роботи машини виводу перше правило конфліктної множини, то граф рішення буде будуватися методом пошуку в ширину (Рисунок 3.5,6). У цьому випадку всі альтернативні ланцюжки міркування, що розташовані на графі рішень, продовжуються паралельно, і жодна не обганяє іншу.



а) б)

Рисунок 3.5. – Пряме виведення: пошук в глибину (а) і пошук в ширину (б).

Безумовно, можливі і змішані стратегії виводу, коли поєднуються обидва види пошуку – пошук у ширину і пошук у глибину, а також різні евристичні стратегії, розглянуті в Главі 2. Оскільки ці стратегії в продукційних системах зв'язані з правилами розв'язування конфліктів, то нижче ми розглянемо їх докладніше.

Крім прямого виводу, у продукційних системах широко застосовується і зворотній вивід, тобто вивід керований цільовими умовами. Такий вивід починається з цільового твердження, що фіксується в робочій пам'яті. Потім відшукується правло-продукція, вивід якої порівняний з метою. Умови даного правила розміщуються в робочу пам'ять і стають новою подціллю. Процес повторюється доти, доки в робочій пам'яті не будуть знайдені факти, що підтверджують цільове твердження. Проілюструємо зворотній вивід на множини продукцій попереднього приклада.

Процес виводу починається з того, що в робочу пам'ять розміщуються цільове твердження Goal, істинність якого необхідно підтвердити чи спростувати, а також множину вихідних фактів, {L, M, N}, що вважаються істинними твердженнями.

Зручно розглядати змінні правил як багатозначні об’єкти, що характеризуються трьома можливими значеннями: “не визначена”, “неправда”, “істина”. У цьому випадку початкове значення змінної Goal -“не визначене”, а факти L, M, і N мають значення “істина”.

Зворотний ланцюжок міркувань для розглянутого приклада зображений на рисунку 3.6. Тут факти, що мають значення “істина”, вигляділені курсивом. Вивід починається з твердження Goal, що розглядається як поточна підціль. Усі продукційні правила, висновки яких порівнянно з поточною підцілью додаються в конфліктну множину правил. На кожному кроці активізується перше правило конфліктної множини. Посилання даного правила додаються в робочу пам’ять і в наступному кроці виступають у якості нової підцілі системи. Така стратегія вирішення конфліктів відповідає пошуку в ширину.

№ кроку	Робоча пам’ять	Конфліктна множина	Активізоване правило
0	Goal, **L, M**, N	10, 11	10
1	Goal, A, **L, M**, N	11, 6, 7	11
2	Goal, A, B, **L, M**, N	6, 7, 8, 9	6
3	Goal, A, B, C, **L, M**, N	7, 8, 9, 1	7
4	Goal, A, B, C, D, **L, M**, N	8, 9, 1 ,2	8
5	Goal, A, B, C, D, E, **L, M**, N	9, 1, 2, 3	9
6	Goal, A, B, C, D, E, F, **L, M**, N	1, 2, 3, 4, 5	1
7	Goal, A, B, C, D, E, F, G, H, **L, M**, N	2, 3, 4,5	2
8	Goal, A, B, C, D, E, F, G, H, I, K, **L, M**, N	3, 4, 5	3
9	Goal, A, **B**, C, D, **E**, F, G, H, I, K, **L, M**, N	4, 5	Зупинка

1) $G \wedge H \rightarrow C$

2) $I \wedge K \rightarrow D$

3) $L \wedge M \rightarrow E$

4) $N \rightarrow F$

5) $O \rightarrow F$

6) $C \rightarrow A$

7) $D \rightarrow A$

8) $E \rightarrow B$

9) $F \rightarrow B$

10) $A \rightarrow goal$

9) $B \rightarrow goal$

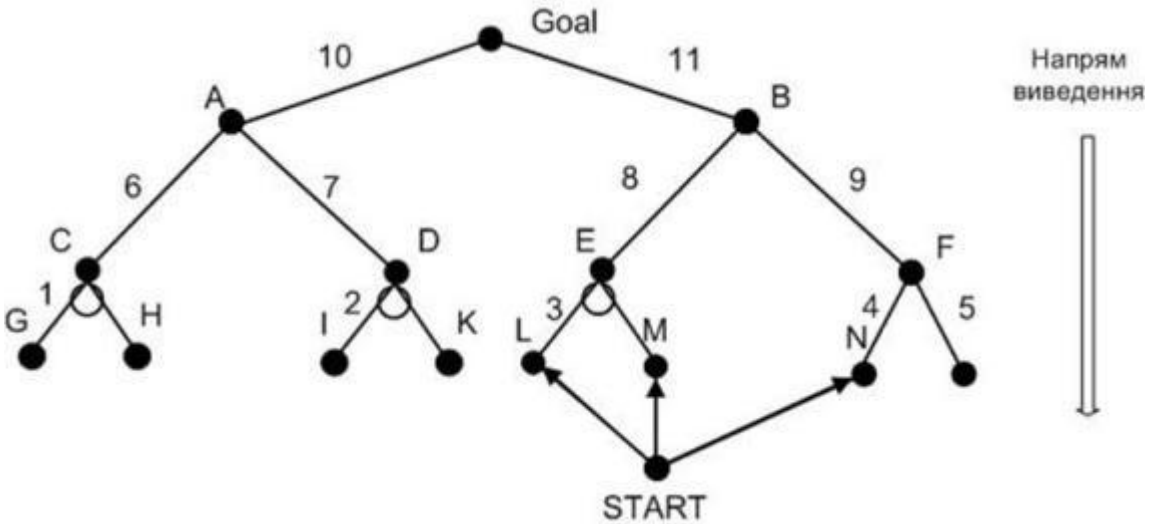


Рисунок 3.6 – Обернене виведення в продукцій них системах

Зазначений процес повторюєте доти, поки всі умови (посилання) деякого правила не стануть істинними, тобто зпівпадуть з фактами, що є в робочій пам’яті. У цьому випадку вивід, правила теж одержує значення “істина”. Дане істинне значення поширюється вгору по ланцюзку активізованих правил. Якщо цільова вершина, отримує значення “істина”, то процес пошуку на цьому закінчується. Якщо цього не відбувається, то знову вибирається перше правило конфліктної множини і т.д.

На рисунку 3.6 при активізації правила 3 встановлюється, що його посилки L і M є фактами. Отже, фактом є і вивід E. Далі, якщо істинно E, то істинно й B (правило 8). І нарешті, якщо істинно B, то справедливо і цільове твердження Goal (правило 11).

Розглянуті приклади показують, що вивід у продукційних системах у загальному випадку відповідає пошуку рішень в І-АБО графах. В окремому випадку, коли правило-продукції не містять в умовній частині кон'юнкцій, вивід у продукційних системах буде відповідати пошуку рішень у просторі станів. Тому розглянуті раніше методи пошуку рішень у просторі станів і в І-АБО графах успішно застосовуються в продукційних системах. Часто в продукційних системах застосовують комбіновані методи пошуку рішень. Наприклад, застосовують двонаправленный пошук. У цьому випадку спочатку пошук ведуть у прямому напрямку доти, поки число станів не стане великим. Потім виконується зворотний пошук уже досягнутих станів (подцелей). При такій організації пошуку існує одна небезпека. Якщо застосовуються методи евристичного пошуку, то, можливо, що підграфи, обстежувані в прямому і зворотному напрямку, виявляться різними і не "зустрінуться". Це приведе до збільшення часу пошука, у порівнянні з односпрямованим пошуком. Цієї небезпеки немає, якщо застосовуються методи "сліпого" пошуку. Тоді комбінований метод значно скорочує час пошуку [77].

Відзначимо, що ситуація, коли в робочу пам’ять заздалегідь вносяться усі відомі факти, зустрічається рідко. Звичайно в робочій пам’яті споконвічно розміщуються тільки частина фактів, необхідних для виводу. Інші зведення продукційна система з’ясовує сама, задаючи питання користувачу, опитуючи вимірювальні чи інші підсистеми, сполучені з нею. Наприклад, слідуючи ланцюжку зворотних міркувань, зображеної на рисунку 3.6, система направляє пошук від мети до фактів. Коли буде активізоване правило 1, то система намагається встановити, чи є висловлення G і H фактами, тобто чи приписане їм значення “істина”. Через те що значення висловів G і H невідомі, то система може задати користувачу питання, щодо істинності висловів G і H. Отримана відповідь вноситься в робочу пам’ять. Якщо висловлення ,G і H одержать значення “істина”, то це змінить хід міркувань системи і відбудеться відмовлення від аналізу інших ланцюжків правил.

Порівнюючи прямий і зворотний вивід, слід зазначити, що прямий вивід має більш широку область застосування. Мається на увазі, що зворотний вивід буде ефективним у тому випадку, якщо проблема, що розв’язується характеризується деякими, чітко визначеними цільовими станами. Коли кількість цільових станів велика, чи їхнє визначення є частиною самої проблеми, те зворотній вивід мало придатний. Подібні ситуації зустрічаються, наприклад, при рішенні задач планування, де план є одночасно і цільовим станом, і шуканим рішенням, у медичних експертних системах при призначенні методики лікування й ін.[53].

Керування вирішенням конфліктів. Як відзначалося вище, вирішення конфліктів – важлива проблема, зв’язана з керуванням порядком застосування правил, що утворюють конфліктну множину.

Порядок активізації правил конфліктної множини визначається обраною стратегією вирішення конфліктів. Раніше, у прикладах, конфліктна множину правил представлялась у вигляді упорядкованого списку. При цьому конфліктні правила дописувалися в кінець цього списку. Прості стратегії вирішення конфліктів засновані на тому, що вибирається або перше, або останнє правило,що входить в список: Вибір першого правила відповідає пошуку в ширину, а вибір останнього правила (тобто щойно доданого) -пошуку в глибину. У багатьох продукційних системах найчастіше застосовують другий спосіб.

Іншими принципами, що використовуються при вирішенні конфліктів, є [21]:

– принцип “стопи книг”;

– принцип найбільш довгої умови;

– принцип метапродукцій;

– принцип пріоритетного вибору.

Принцип “стопки книг” полягає в тому, що список конфліктуючих правил упорядковується відповідно до частоти використання продукцій у минулому. У першу чергу вибирається продукція, що використовувалася найчастіше.

Принцип найбільш довгої умови віддає пріоритет тій продукції, ядро якої має найбільш довга умова. Такі продукції відповідають специфічним (вузьким) ситуаціям. Даний принцип спирається на той факт, що продукції з довгими умовами враховують більше інформації про поточну ситуацію, і це повинно приводити до прискорення пошуку рішення.

Принцип метапродукцій заснований на введенні в базу знань спеціальних метаправил, що упорядковують процес вирішення конфліктів. На підставі метаправил здійснюється аналіз множини конфліктних правил і, у визначених ситуаціях, активізуються ті чи інші правила з цієї множини.

У випадку пріоритетного вибору з кожною продукцією зв’язується статичний чи динамічний пріоритет Pr (см.3.4), що визначає порядок її активізації.

Іноді в післяумовах продукції може вказуватися ім’я наступної продукції, яку необхідно виконати. Це перетворює систему продукцій у звичайний алгоритм.

3.4. Семантичні мережі

3.4.1. Загальне поняття про семантичні мережі

Семантичні мережі не є однорідним класом моделей представлення знань. Часто загальною основою віднесення схеми представлення знань до семантичної мережі є те, що вона представляється у вигляді спрямованого графа, вершини якого відповідають об'єктам (поняттям, сутностям) предметної області, а дуги – відносинам (зв'язкам) між ними. І вузли, і дуги, як правило, мають мітки (імена), Імена вершин і дуг звичайно збігаються з іменами відповідних об'єктів і відносин предметної області.

Об'єкти предметної області, що відображаються в семантичній мережі, можна умовно розділити на три групи: узагальнені, індивідуальні (конкретні) і агрегатні об'єкти.

Загалиний об'єкт відповідає деякій збірній абстракції реально існуючого об'єкта, чи процесу явища предметної області. Наприклад, "виріб", "підприємство", "співробітник" і т.д. Узагальнені об'єкти фактично представляють визначені класи предметної області.

Індивідуальний об’єкт – це певним чином вигляділенний одиничний представник (екземпляр) класу. Наприклад, “співробітник Петров И.Н.” Агрегатним називається складений об’єкт, утворений з інших об’єктів, що розглядаються як його складові частини. Наприклад, виріб складається з сукупності деталей, підприємство складається з сукупності відділів, служб, цехів.

Введена класифікація об’єктів є відносно. В залежності від розв’язуваної задачі той самий об’єкт може розглядатися як узагальнений чи індивідний, як агрегатний чи неагрегатний.

Типи, зв’язків між об’єктами семантичних мереж можуть бути будь-якими. Але найчастіше застосовуються наступні основні зв’язки (відносини): “рід-вид”, “є представником”, “є частиною”. Наявність зв’язку типу “рід-вид” між узагальненими об’єктами А і В означає, що поняття А більш загальне, ніж поняття В. Будь-який об’єкт, що відображається поняттям В, відображається і поняттям А, але не навпаки. Наприклад, поняття “тварина” – це родове поняття для о’бъекта “птах”. Усі властивості родового об’єкта А, як правило, притаманні і видовому об’єкту В. Іншими словами, об’єкт В успадковує властивості об’єкта А.

Зв’язок “є представником” існує звичайно між узагальненим та індивідним об’єктом, коли індивідний об’єкт виступає в ролі представника деякого класу. Так, індивідний об’єкт “вівчарка Альма” є представником (екземпляром) узагальненого об’єкта “вівчарка”. Екземпляр може бути представником декількох узагальнених об’єктів. У цьому випадку йому притаманні властивості декількох узагальнених об’єктів, що відповідають множинному успадкуванню.

У ряді випадків між зв’язками “рід-вид” і “є представником” не роблять розбіжностей, відзначаючи, що ці зв’язки задають відношення “загальна-частка” (Рисунок 3.7). Іноді це приводить до непорозумінь. Тому для формалізації таких зв’язків будемо використовувати відносини ako (від англ. a-kind-of – різновид) і is a (від англ. is a member of the class -бути представником класу) [94].

Не менш важливе відношення “є частиною” (англ. part of). Дане відношення зв’язує агрегатний об’єкт із його складовими частинами. Воно дозволяє відбивати в базі знань структуру об’єктів предметної області. Іноді дане відношення позначають міткою “має” (англ. has).

Як приклад на рисунку зображена семантична мережа, що представляє частину знань про тваринний світ.

Дана мережа була розроблена А.М. Коллінзом і М.Р. Куілліаном (1969) і використовувалася, для моделювання механізмів пам'яті людини. Завдяки відносинам ako мережа являє собою якусь ієрархічну структуру і дозволяє відповідати на такі питання, як: "чи є канарка птахом?", "чи може канарка літати?" і т.д. У ході експериментів із представленою моделлю було встановлено, що людина прагне запам'ятовувати інформацію, що відповідає найбільш абстрактному рівню. Наприклад, замість того, щоб безпосередньо запам'ятовувати факт "канарка вміє літати", людина представляє цей факт у вигляді властивості, і зв'язує його з поняттям "птах". Аналогічно властивості: "дихати", "їсти", "переміщатися" – зв'язуються з поняттям "тварина". Такий метод запам'ятовування інформації дозволяє виключити її дублювання в базі знань завдяки спадкуванню властивостей у відповідності з ako ієрархією. Наприклад, властивість "мати крила" притаманне птахам, отже, воно притаманне і канарці.

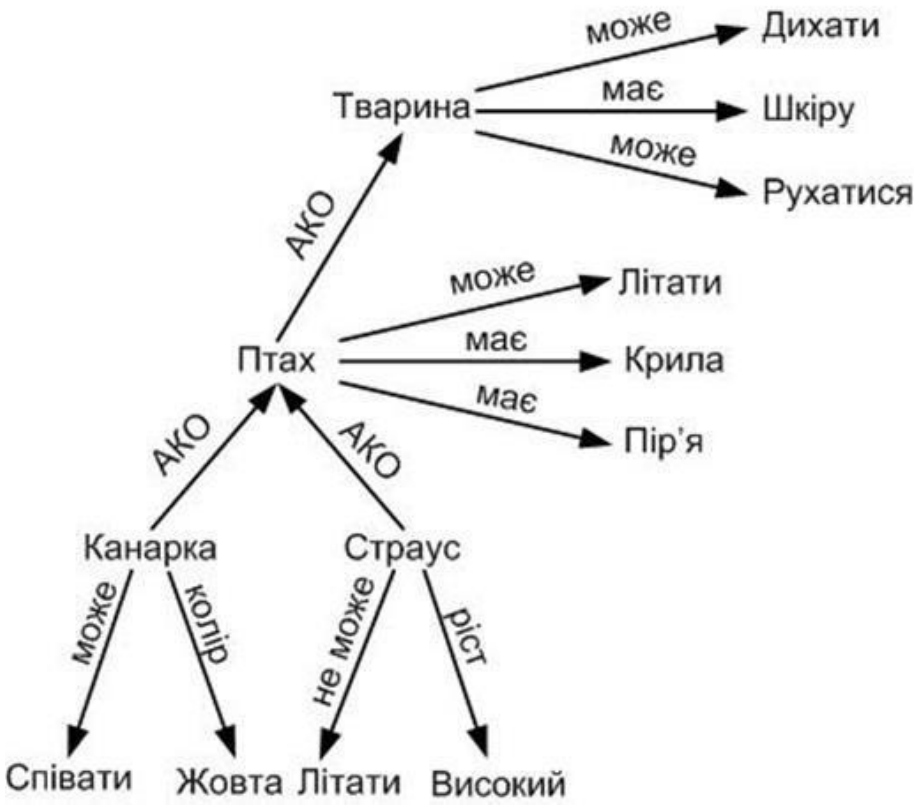


Рисунок 3.7 – Світ тварин у вигляді семантичної мережі.

Вперше семантичні мережі з’явилися в області машинної лінгвистики як засіб аналізу змісту (семантики) природної мови. У 1967 році М.Р. Куілліаном була розроблена програма, що фіксує зміст англійських слів, побудована за принципом тлумачного словника. У цій програмі деяке слово англійської мови визначається в термінах, що виражаються іншими словами. Кожне слово визначається множиною покажчиків на інші слова. Сукупність вказівників утворить мережу, “подорожуючи” у якій можна з’ясувати зміст того чи іншого слова. У семантичній мережі Куілліана вершини відповідають обумовленому поняттю і забезпечуються вказівниками на інші слова, що розкривають дане поняття. База знань організована у вигляді сторінок (площин), на кожній з який представляється граф, що визначає одне слово. На рисунку 3.8 зображені три різних визначення англійського слова “plant”.

Вказівники, позначені на рисунку 3.8 суцільними стрільцями, формують наступні визначення слова “plant “;

1) жива (live) структура (structure), що не відноситься до тварин (animal), часто має листи (leaf), одержує (get) харчування (food) з (from) повітря (air), води (water) і ґрунту (earth);

2) устаткування (апаратура – apparatus), використовуване (use) для (for) реалізацій будь-якого процесу (process) у (in) промисловості (industry);

3) поміщати (put) щось (seed – насіння, plant – рослина і т.п.) у ґрунт (earth) для (for) вирощування (grow)

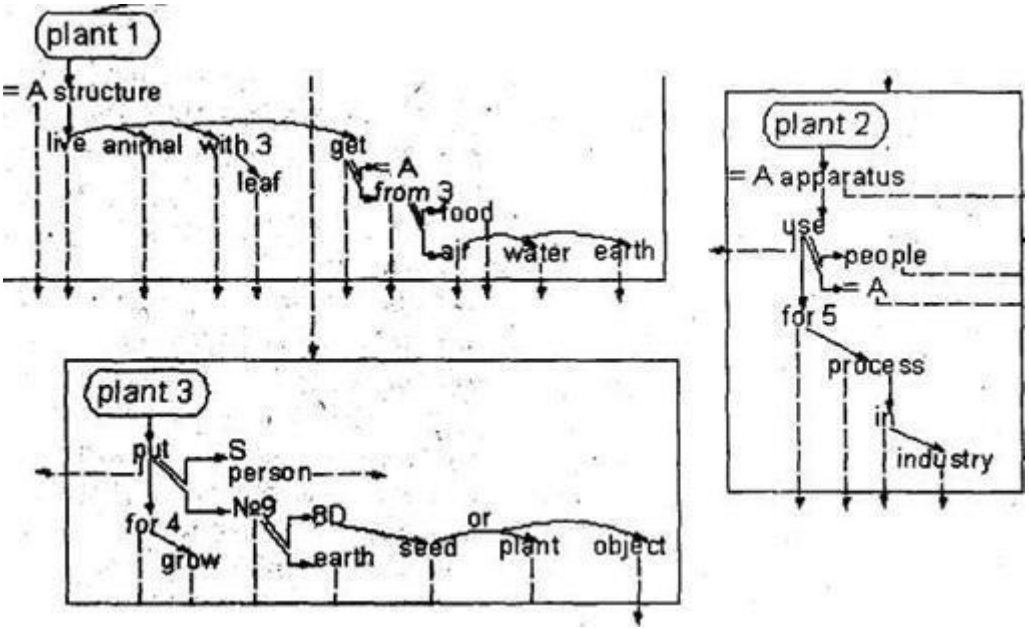


Рисунок 3.8-визначення слова “plant”

Перше визначення (plant 1) відповідає іменнику “рослина”, друге (plant 2) – іменнику “завод”, третє (plant 3) – дієслову “садити”. Вказівники, позначені на рисунку штриховими лініями, посилаються на слова-поняття, представлені на інших сторінках.

Один з механізмів виводу, що застосовується у розглянутій моделі семантичної мережі, заснований на пошуку по перетинанню. Він дозволяє встановлювати стосунки між двома словами. Пошук виконується методом у ширину по зовнішніх вказівниках із двох вихідних сторінок доти, поки не буде знайдено загальне поняття, що відповідає вершині перетинання двох напрямків пошуку. Знайдені в результаті пошуку шляху від вихідних вершин до вершини перетинання і будуть представляти відношення між вихідними словами-поняттями. Наприклад, слова-поняття “cry” (плач, плакати) і “comfort” (розрада, утішати) приводять до перетинання шляхів пошуку у вершині “sad” (сумний, сумовитий). У результаті аналізу шляхів пошуку модель дозволяє зробити наступний вивід:

“Плач – це один зі способів створення сумних звуків. Утішати – робити щось менш сумним”.

Таким чином, модель дозволяє формувати відносини, що представлені в ній неявно отже, на її основі можна одержувати нові знання. Це дозволяє будувати системи розуміння природної мови, що володіють наступними можливостями [77]:

1) визначати основний зміст тексту шляхом пошуку множини вершин перетинання;

2) здійснювати вибір необхідного значення багатозначного слова на основі найкоротшого шляху від цього слова до інших слів розглянутої пропозиції (свого роду семантична метрика);

3) формувати відповіді на різні запити шляхом встановлення взаємозв'язків між словами-поняттями запиту і словами-поняттями, що зберігаються в пам'яті системи.

Розглянутий підхід одержав деякий розвиток у системі TLC (Teachable Language Comprehender: навчальна система розуміння мови), що розроблена Квілліаном. Однак практичний успіх TLC був обмежений. Це мається на увазі тому, що набір використовуваних типів зв'язків (відносин) між поняттями був недостатній. Використовувалися тільки такі зв'язки, як: "класс – екземпляр класу", "атрибут.- значення". Окрім цього, сам пошук не враховував зміст зв'язків. Представлення знань вимагає введення в розгляд більш багатого набору відносин між поняттями (об'єктами, сутностями) предметної області. Подальший розвиток мережних моделей представлення знань здійснювався саме в цьому напрямку. Роботи Р.Симмонса і К.Филмора вказали на важливу роль дієслів при аналізі змісту пропозицій природної мови. Пропозицію можна представити вершиною-дієсловом і різними відмінковими зв'язками (відносинами). Таку структуру називають відмінковою рамкою. Серед відмінкових відносин вигляділяють наступні: агент – відношення між подією і тим хто (що) його робить; об'єкт – відношення між подією і тим, над чим виконується дія; інструмент – об'єкт, за допомогою якого відбувається дія; місце – місце здійснення події; час – час здійснення події. При аналізі пропозиції програма знаходить дієслово і встановлює відповідні відмінкові відносини між частинами пропозиції.

Як приклад на рисунку 3.9 зображена відмінкова рамка пропозиції: “Іван закріпв деталь клеєм”. Відмінкова рамка фіксує знання відмінкової (лінгвістичної) структури природних мов у вигляді мережного формалізму.

У відповідності з введеним раніше визначенням семантичної мережі, відсутні будь-які обмеження як на типи відносин, так і на типи об’єктів, відображуваних у мережі. У більшості випадків різноманіття об’єктів мережі можна розділити на три групи [29,16]:

1. об’єкта-поняття- зведення про фізичні й абстрактні об’єкти, предметну область;
2. об’єкта-події- абстрактні чи конкретні дії, що можуть привести до зміни стану предметної області;
3. об’єкта-властивості – уточнюють поняття і події, наприклад, указують характеристики понять (колір, форму, розміри і т.п.), фіксують параметри подій (місце, час, тривалість).

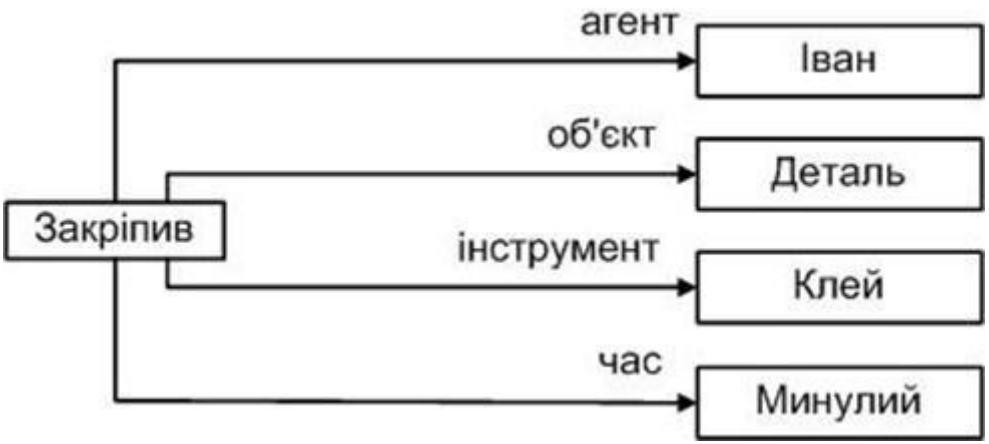


Рисунок 3:9 – Відмінкова рамка

Різноманіття відносин, що використовуються у семантичних мережах, підрозділяються на наступні групи:

1. лінгвістичні відносини, що, у свою чергу, підрозділяються на відмінкові (агент, об’єкт, інструмент, час, місце), дієслівні (нахилення, час, вид, число, застава) і атрибутивні (колір, розмір, форма і т.п.);
2. логічні відносини (кон’юнкция, диз’юнкція, заперечення, імплікація);
3. теоретико-множинні відносини містять у собі відносини типу “множину-підмножина” (”рід-вид”, “клас-підклас”), “ціле-частина”, “елемент множини” і ін.
4. квантифіковані відносини, що підрозділяються на логічні квантори спільності й існування, нечіткі квантори (багато, трохи, часто т.д.).

Таким чином, розглянуті раніше відносини, (“рід-вид”, “бути представником”, “бути частиною”, відмінкові відносини) далеко не вичерпують усього набору відносин, застосовуваних у семантичних мережах. Але вони утворять гарну основу для побудови прикладних баз знань. Особливе місце при цьому займають теоретико-множинні відносини, що володіють транзитивними властивостями. Відношення R називається транзитивним, якщо для будь-яких об’єктів α, β, γ таких, що α знаходиться у відношенні $Rc\beta(\alpha R\beta)i\beta$ – у відношенні $Rc\gamma(\beta R\gamma)$, випливає справедливість твердження “α знаходиться у відношенні R с γ“
, тобто

$$(\alpha R\beta \wedge \beta R\gamma) \Rightarrow \alpha R\gamma.$$

Наприклад, транзитивність родовидових відносин забезпечує можливість спадкування властивостей від родового об’єкта до виду і підвиду. Механізм спадкування властивостей забезпечує проведення найпростіших дедуктивних висновків типу: “Усі люди смертні. Сократ – людина. Отже, Сократ смертний” [34].

У базах знань вигляділяють інтенсіональні і єкстенсіональні знання. Якщо є кінцева множину атрибутів $A=\{A_1, A_2, \ldots, A_n\}$ і кінцева множинf відносин $R=\{ R_1, R_2, \ldots, R_m \}$, то схемою чи інтенсіоналом відносини R_i {i=l,2,…,m) називається набір пар виду.

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

де $DOM(A_j)$ – домен атрибута A_j, тобто множину можливих значень атрибута. Екстенсіонал відносини R_i,- – це множина фактів,

$EXT(R_i)=\{ F_1, F_2, \ldots, F_p \}$

де $F_1, F_2, \ldots, F_p$ – факти відносини R_i, що задаються звичайно у вигляді сукупності пар “атрибут-значення”. Факт конкретизує визначене відношення. У графічній інтерпретації він являє собою подграф, що має зіркоподібну структуру. Коренем подграфа є вершина предикатного типу з міткою, що відповідає імені відносин. Ребра підграфа відзначені іменами атрибутів.

Інтенсіональна семантична мережа представляє предметну область, що моделюється на узагальненому, концептуальному рівні, а екстенсіональна – наповнює її конкретними, фактичними даними. Таким чином, семантичну базу знань можна розглядати як сукупність об’єктів і відносин, частина з яких визначена інтенсионально, а частина – екстенсионально.

3.4.2. Способи опису семантичних мереж і логічний вивід

Після знайомства з загальними поняттями семантичних мереж розглянемо способи опису семантичних мереж. Найчастіше для цих цілей використовують концептуальні графи Дж. Соува і блокові структури Г.Хендрікса. Блокові структури докладно описані в [8,35]. Тому розглянемо опис семантичних мереж у вигляді концептуальних графів.

Вершинами концептуального графа є або об'єкти (поняття, сутності) предметної області, або концептуальні відносини. Ребра концептуального графа зв'язують між собою поняття-вершини-поняття і відносини-вершина-відносини. При цьому ребра можуть виходити з вершини-поняття і закінчуватися в ставленні-вершині-відношенні, і навпаки. На рисунку 3.10 вершини "м'яч" і "червоний" відповідають вершинам-поняттям, а вершина "колір" – бінарному відношенню. Для того щоб розрізняти зазначені типи вершин, на концептуальних графах їх зображують прямокутником і еліпсом, відповідно.

Одне з переваг вигляділенння відносин у самостійні вершини концептуального графа полягає в спрощенні представлення парных відносин. Таке відношення представляється ставленням-вершиною-відношенням з n-ребрами (Рисунок 3.10).

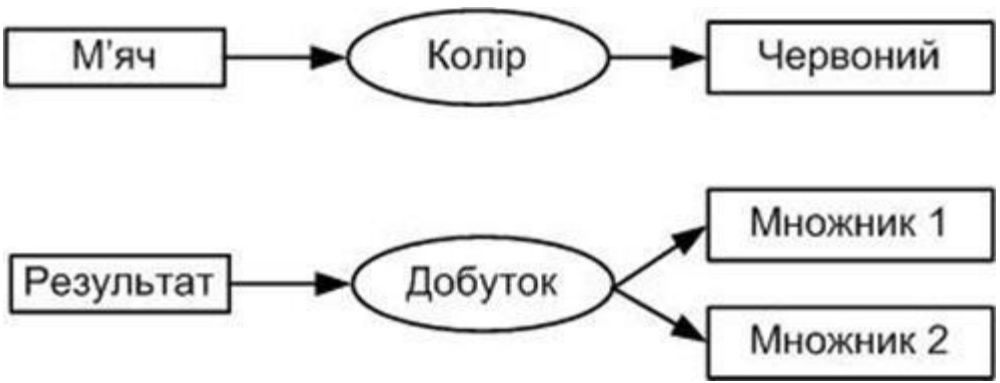


Рисунок 3.10 – Приклади концептуальних графів

Зазвичай кожний концептуальний граф фіксує одне речення. Тоді база знань будет представлятись у вигляді сукупності таких графов. Наприклад, верхній граф на рисунку 3.10 відповідає реченню “Мяч має червоний колір”. Граф, що зображений на рисунку 3.11, відображає більш складне речення: “Іван закріпив деталь стільця клеєм”.

Рисунок 3.11 – Концептуальний граф пропозиції

Цей граф використовує відмінкові відносини дієслова “закріпити” і демонструє можливості концептуальних графів для представлення пропозицій природної мови.

Кожна поняття-вершина-поняття концептуального графа може мати мітку типу. Тип позначає клас приналежності вершини. На концептуальному графі мітку типу вершини відокремлюють двокрапкою від конкретного імені вершини. На рисунку 3.11 за допомогою мітки типу показано, що Іван є столяром. На концептуальних графах можна також вводити індивідні поняття-вершини-поняття з однаковими іменами. Щоб розрізняти єкземпляри об’єктів (понять) з однаковими іменами, використовується спеціальний числовий маркер, перед яким записаний символ #.

Граф на рисунку 3.12 вказує, що конкретний “автомобіль” з маркером #1235 має зелений колір.

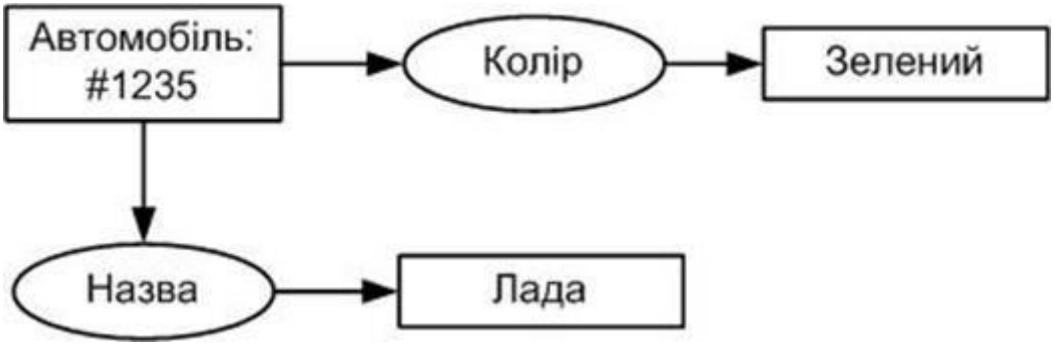


Рисунок 3.12- Концептуальний граф з числовим маркером

Крім числового маркера, на концептуальних графах можуть застосовуватися узагальнені маркери, що позначаються знаком *. У сполученні з змінною, що записується після цього знака, узагальнений маркер виявляється корисним у ситуаціях, коли дві різні вершини графа представляють той самий об’єкт. На відміну від числових маркерів, що застосовуються для вигляділенння в семантичній мережі індивідних об’єктів, узагальнені маркери дозволяють вигляділяти той самий узагальнений об’єкт заданого типу (Рисунок 3.13). Як приклад на рисунку зображений концептуальний граф, що відповідає пропозиції: “Маляр забруднив свою руку фарбою”.

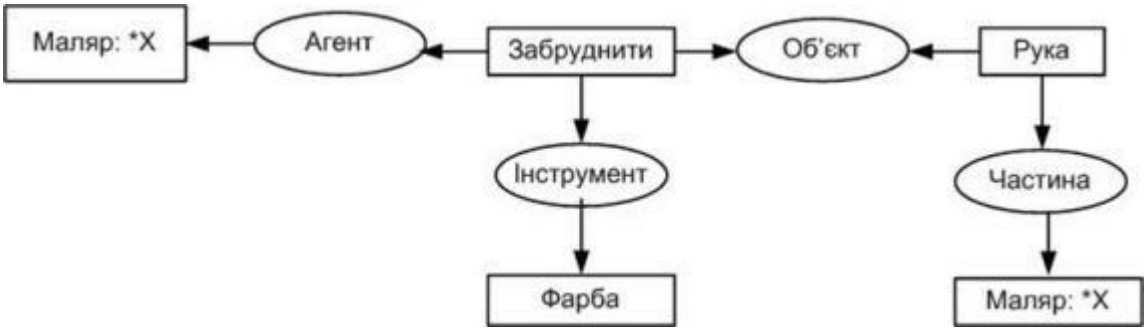


Рисунок 3.13 – Концептуарний граф з загальним маркером

На концептуальних графах вводяться операції, що дозволяють виконувати їхнє перетворення. Нові графи виходять за допомогою чотирьох операцій: копіювання, спеціалізації, об'єднання і спрощення. Розглянемо їх докладніше.

Операція копіювання дозволяє сформувати з вихідного графа G його копію, наприклад, G_1.

Операція спеціалізації полягає в заміні узагальненої вершини її індивідним (конкретним) варіантом. При цьому може бути два випадки:

1) якщо вершина відзначена узагальненим маркером, те узагальнений маркер заміняють конкретним індивідним маркером (іменним чи числовим).

2) тип вершини можна замінити відповідними підтипами, що володіє необхідними властивостями.

Операція об’єднання дозволяє одержувати з двох графів один. Якщо на графі G_1 мається вершина-поняття V_1, що аналогічна вершині-поняттю V_2 графа G_2, то можна сформувати новий граф G_3 видаливши V_2. При цьому всі зв’язки V_2 необхідно переорієнтувати з V_2 на V_1.

Операція спрощення дозволяє виключити на графах дублікати відносин. При цьому виключаються ставлення-вершина-відношення і зв’язані з нею дуги. Дублювання відносин часто виникає в результаті виконання операції об’єднання.

Операція спеціалізації дозволяє зіставляти двох вершин концептуального графа і, якщо зпівставлення успішне, виконувати об’єднання. Спільне використання операцій спеціалізації й об’єднання забезпечує реалізацію механізму спадкування. Наприклад, замінивши узагальнений маркер індивідним, ми поширюємо властивості типу на конкретний індивідний об’єкт. Якщо, у ході виконання операції обмеження, відбувається заміна типу на його підтип, то має місце спадкування “підклас-клас-підклас”.

Операції спеціалізації й об’єднання конкретизують вихідний граф. Якщо граф G_1 – це конкретизація графа G_2, то можна сказати, що істинно і зворотно, тобто G_2 – є узагальнення G_1. Узагальнення широке використовується при навчанні СШІ, дозволяючи виводити правдоподібні умови по приватних прикладах (див. §4.3).

Розглянуті вище найпростіші операції не гарантують одержання правильних тверджень на основі перетворених графів. Конкретизація вершин-понять, виконувана в ході операцій спеціалізації й об’єднання, не завжди відповідає дійсності. Наприклад, якщо в ході виконання операції об’єднання з’єднуються два графи, агенти яких позначені тим самим поняттям, то це зовсім не означає, що не може статись так, що ці поняття відповідають двом різним індивідним об’єктам. Наприклад, пропозиції “Собака гризе кістку” і “Собака знаходиться в парку” можна об’єднати в одне, якщо мова йде про одне й ту саму тварину, а не про двох різних тварин. Проте, розглянуті операції не приводять до тверджень, позбавлених змісту, і дозволяють виконувати правдоподібні міркування.

Концептуальні графи можуть містити вершини, що є пропозиціями. Такі вершини зображуються у вигляді прямокутного блоку, що містить підграф, що відповідає пропозиції. Як приклад на рисунку 3.14 зображений граф твердження: “Олексій думає, що Тетяні подобається іграшка”.

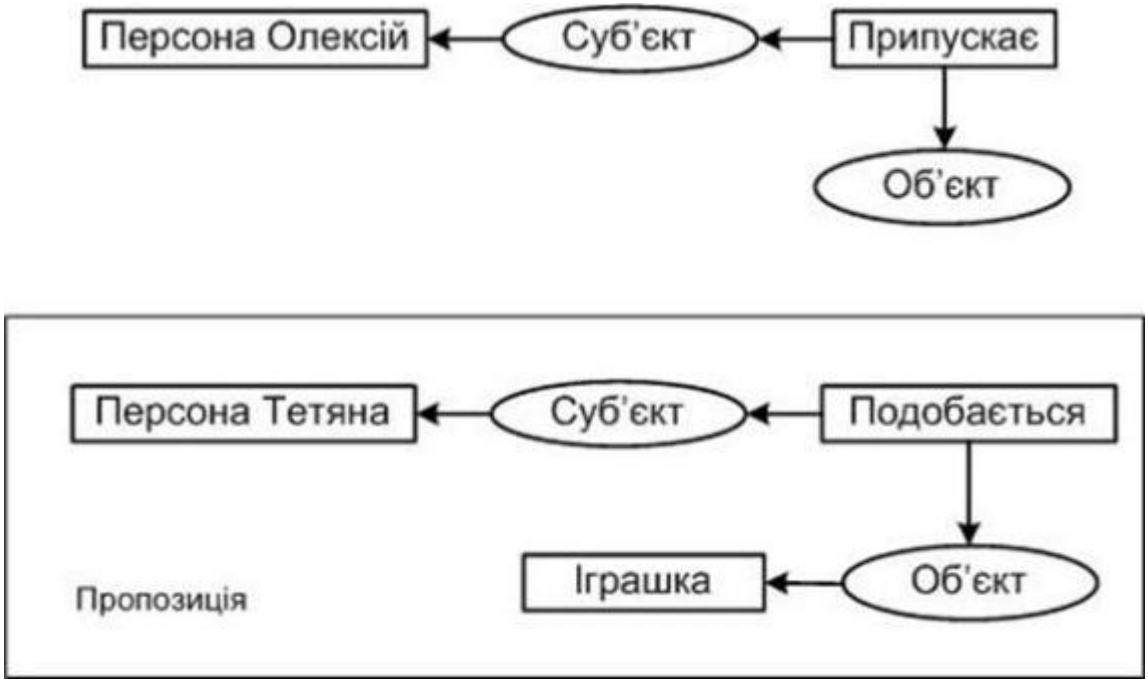


Рисунок 3.14 Концептуальний граф, що містить пропозицію

Введення окремих блоків для позначення пропозицій дозволяє фіксувати відносини між пропозиціями. У приведеному прикладі дієслово “думає” представляє не конкретна фізична дія, а виражає відчуття Олексія. Тому на графі введене нове відношення “суб’єкт”, що представляє обличчя, що випробує ті чи інші відчуття. Твердження, що містять подібні лексичні одиниці (довіряти, вірити, мати намір і т.п.) є об’єктом досліджень модальної логіки.

Концептуальний граф дозволяє природним образом виражати кон’юнкцію. Наприклад, за допомогою концептуального графа ми легко можемо представити пропозицію “Собака знаходилася в саду і гризла кісту”. Складніше з іншими логічними операціями: диз’юнкцією, логічним запереченням, кванторами.

Для представлення заперечення на концептуальних графах, застосовується унарный логічний оператор “не”. Оператор застосовують до пропозиції. Твердження, що фіксується пропозицією, вважається в цьому випадку помилковим. Граф, зображений на рисунку 3.15, відповідає пропозиції: “Немає іграшок, що подобаються Тетяні”.

Використовуючи заперечення і кон’юнкцію, можна представити на концептуальних графах диз’юнкцію. Однак для спрощення концептуальних схем диз’юнкцію представляють у вигляді спеціального бінарного відношення “чи”, аргументами якого є вершини-пропозиції.

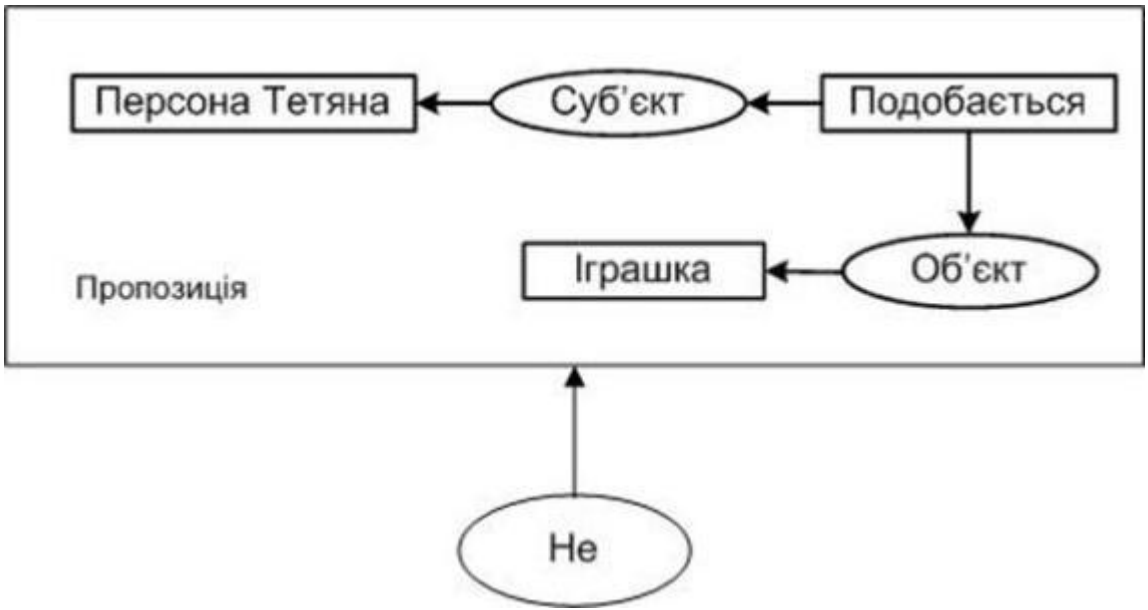


Рисунок 3.15 – Концептуальний граф з логічним запереченням

Узагальнені поняття-об’єкти-поняття, відображувані на концептуальних графах, зв’язані квантором існування. Так, граф, зображений на рисунку 3.10, можна описати формулою

$\exists x \exists y$ (м’яч (x) $\wedge$ колір (x, y) $\wedge$ червоний (y)).

Квантор спільності виходить при запереченні пропозицій, зв’язаних квантором існування. Наприклад, застосувавши заперечення до графа, зображеному на рисунку 3.10, одержимо

$\forall x \forall y \neg$ (м’яч (x) $\wedge$ колір (x, y) $\wedge$ червоний (y)).

По своїх виразних можливостях концептуальні графи еквівалентні логіці числення предикатів. Існує можливість взаємного переходу від описів предметної області мовою числення предикатів до концептуальних графів, і навпаки. Вибір тих чи інших засобів визначається характером розв’язуваних задач і схильностями розроблювачів СШІ.

Методи виводу на семантичних мережах використовують асоціативні й алгоритми, що зіставляють, що зводяться до перебування шляхів на графі, побудові транзитивних замикань, вигляділенню підграфів з визначеними властивостями. Основні підходи до побудови процедур виводу на семантичних мережах викладені в [8,29,34]. У багатьох випадках вони базуються на простих операціях над графами,: видалення і додавання нових вершин і ребер, спеціалізація, пошук вершини чи ребра по імені, перехід від однієї вершини до іншої по зв’язках, об’єднання підграфів і ін.

Одним із простих засобів виводу на семантичних мережах є пошук по перетинанню, що був розглянутий раніше.

Іншим могутнім засобом виводу є зпівставлення зі зразком. У даному випадку відбувається зпівставлення окремих фрагментів семантичної мережі. При цьому запит до бази знань представляється у вигляді автономного підграфа, що будується за тими ж правилам, що і семантична мережа. Пошук відповіді на запит реалізується зпівставленням підграфа запиту з фрагментами семантичної мережі. Для цього здійснюють накладення підграфа запиту на відповідний фрагмент мережі. Успішним буде те накладення, у результаті якого фрагмент мережі виявляється ідентичним підграфу запиту. При цьому допускається використання в запиті змінних. Змінна запиту зіставляється з константою фрагмента мережі.

Нехай, наприклад, є запит: ” чи існує такий птах X, що вміє пекти і має жовтий колір?” (Рисунок 3.16).

Рисунок 3.16 – Підграфа запиту

Ізоморфне вкладення підграфа запиту в семантичну мережу, зображену на рисунку 3.7, дозволяє дати відповідь X=”канарка”.

У загальному випадку в підграфі запиту можуть бути задані об’єкти, атрибути, імена відносин, не представлені в базі знань явно. Це вимагає виконання попередніх перетворень фрагментів семантичних мереж, що зіставляються. У результаті таких перетворень можуть бути отримані нові зв’язки. Зазначені перетворення виконуються на основі найпростіших базових операцій розглянутих раніше.

Методи виводу на семантичних мережах, що використовують ідею перетинання чи шляхів зпівставлення фрагментів мереж, мають істотний недолік. Він зв’язаний з комбінаторним ростом числа чи зіставлень перетинань у мережах досить великої розмірності.

У ряді систем, заснованих на семантичних мережах, використовуються спеціалізовані правила виводу. У [8] розглядаються методи дедуктивного виводу на семантичних мережах, засновані на поняттях розфарбування графів і операторах перетворення мереж.

3.5. Фрейми

3.5.1. Структура фрейму

Фрейм можна розглядати як фрагмент семантичної мережі, призначений для опису деякого об’єкта (події, ситуації) предметної області з усією сукупністю властивих йому (їй) властивостей. У цьому змісті фрейми – це підграфи семантичних мереж. Вони представляють блоки знань, кожним з який можна маніпулювати, як єдиним цілим, що дозволяє краще структурувати базу знань.

Фрейми вперше були запропоновані для представлення знань у 1975р. М. Мінським. Відповідно до його визначення фрейми – це структури даних, призначені для представлення стереотипних ситуацій. Коли людина потрапляє в нову ситуацію, він витягає з пам'яті раніше накопичені блоки знань, що мають відношення до поточної ситуації, і L намагається застосувати їх. Ці блоки знань і являють собою фрейми. Істинно, знання людини організовани у вигляді мережі фреймів, що відбивають його минулий досвід. Наприклад, ми легко можемо представити типовий одномісний номер у готелі. Звичайно він має ліжко, ванну кімнату, шафу для одягу, телефон, план евакуації у випадку пожежі і т.п. Деталі кожного конкретного номера можуть відрізнятися від приведеного опису. Але вони легко уточнюються, коли людина виявляється в конкретному номері: колір шпалери і фіранок, положення вимикачів світла й ін.

У найпростішому випадку під фреймом розуміють наступну структуру,

{n, (ns_1, vs_1, ps_1), (ns_2, vs_2, ps_2), … , (ns_k, vs_k, ps_k)},

де **n** – ім’я фрейму, $\mathbf{ns_1}$ – ім’я слота, $\mathbf{vs_1}$ – значення слота, $\mathbf{ps_1}$ – ім’я приєднаної процедури. Фрейм складається зі слотів. Слоти – це деякі незаповнені підструктури фрейму. Після заповнення слотів конкретними даними, фрейм буде представляти ту чи іншу ситуацію, явище чи об’єкт предметної області. Процедура є необов’язковим елементом слота. Як значення слотів можуть виступати імена інших фреймів, що забезпечує побудову мережі фреймів. Як приклад на рисунку 3.17 зображено фрагмент мережі фреймів, що представляють типовий одномісний номер у готелі. Тут “меблі”, “інвентар”, “санвузол”, “оплата” – це імена слотів. Значеннями слотів “меблі”, “інвентар”, “санвузол” є список, імен інших фреймів, що відповідно описують предмети й устаткування, що знаходяться в номері.

У фреймових системах частина фреймів представляє індивідні й об’єкти предметної області. Такі фрейми називають екземплярами фреймів чи фреймами-прикладами. Інші фрейми, що представляють узагальнені об’єкти, називають класами чи фреймами-прототипами. Фрейм-прототип відповідає інтенсіональному опису множини фреймів-прикладів.

Фрейм може містити спеціальний слот, за допомогою якого задається відношення даного фрейму з іншими фреймами мережі. Найчастіше в мережі фреймів використовують теоретико-множинні відносини, що дозволяють будувати ієрархічні структури. Спеціальний слот, що називається **is a** слотом, фіксує відношення “екземпляр класу”. Слот **ako** вказує, що данний фрейм є прямим підкласом розташованого вище по ієрархії фрейму. Розташований вище фрейм називають батьківським фреймом чи суперкласом.

Наприклад, ліжко “Емілія”, встановлена в готельному номері, є конкретизацією узагальненого об’єкта “ліжко” (Рисунок 3.17). Це позначено за допомогою слота **is a,** що вказує, що ліжко “Емілія” – екземпляр класу (фрейму-прототипу) “ліжко”. Слот **ako** вказує, що одномісні номери є підкласом готельних номерів.

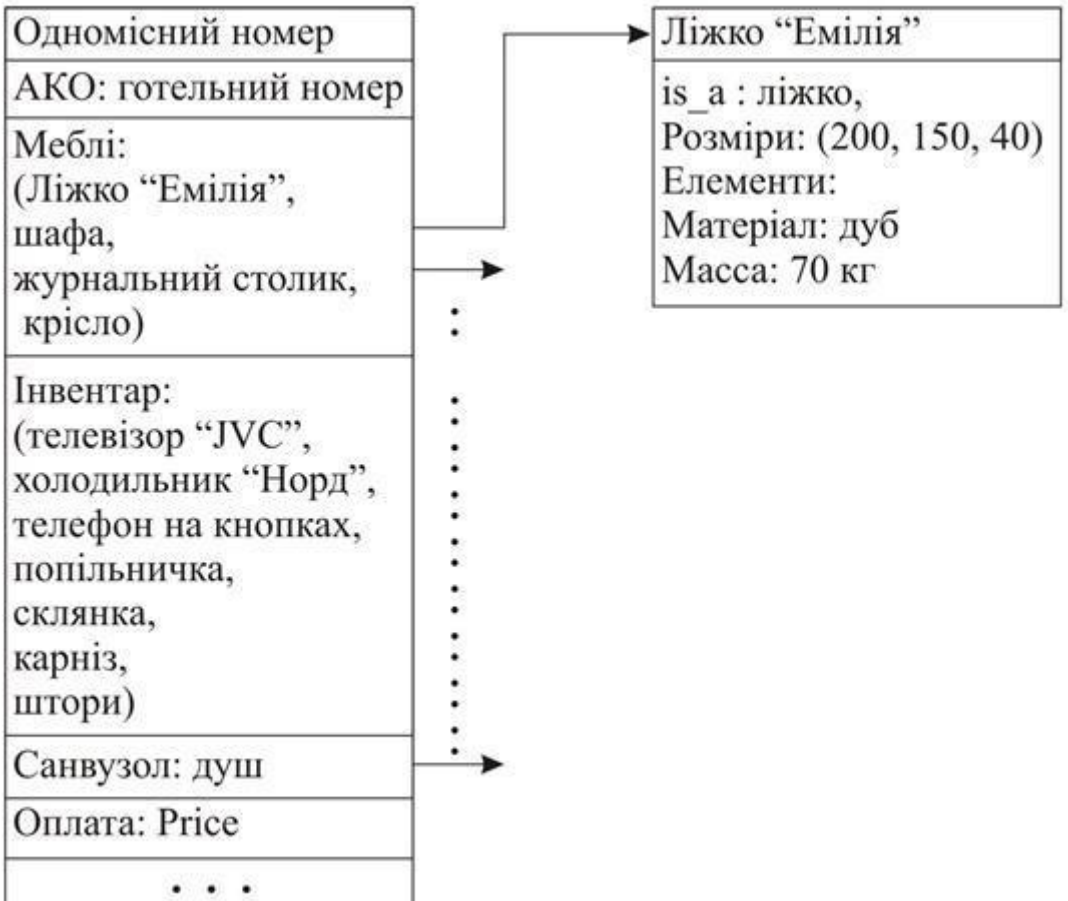

Одномісний номер
АКО: готельний номер
Меблі:
(Ліжко “Емілія”,
шафа,
журнальний столик,
крісло)
Інвентар:
(телевізор “JVC”,
холодильник “Норд”,
телефон на кнопках,
попільничка,
склянка,
карніз,
штори)
Санвузол: душ
Оплата: Price
. . .
Ліжко “Емілія”
is_a : ліжко,
Розміри: (200, 150, 40)
Елементи:
Матеріал: дуб
Масса: 70 кг

Рисунок 3.17 – Фрейм, що представляє готельний номер

Важливою характеристикою фреймів є можливість включення в слоти приєднаних процедур. Вигляділяють два види процедур: процедури-демони і процедури-слуги. Процедури-демони активізуються автоматично при кожній спробі додавання чи видалення даних зі слота. Найчастіше використовуються три типи процедур-демонів: **“якщо-додано”, “якщо-вилучено”** і **“якщо-потрібно”.** Процедура “якщо-дабавлено” виконується, коли нові дані містяться в слоті. Процедура “якщо-вилучено” виконується, коли дані віддаляються зі слота. Процедура “треба” автоматично запускається, коли запитується інформація зі слота, а з ним не зв’язані ніякі значення. За допомогою зазначених процедур звичайно виконуються всі рутинні операції, що забезпечують підтримку бази знань в актуальному стані. Процедури-слуги активізуються тільки по запиту при звертанні до слоту. Наприклад, при звертанні до слоту “оплата” буде активізована процедура “Price”, що визначить вартість проживання в номері за добу.

Фреймові системи являють собою ієрархічно організовані структури, що реалізують принцип спадкування інформації. Спадкування значень відбувається в напрямку “суперклас – підклас”, “клас – екземпляр класу”. Наприклад, ліжко “Емілія” є екземпляром класу “ліжко”. Тому ліжко

“Емілія” успадковує усі властивості свого суперкласу. Зокрема, значення слота “елементи” може бути успадковане від суперкласу. У ряді фреймових систем в склад слотів включають різні вказівники спадкування [37]. Покажчик спадкування фіксує ту информацію, що передається від слотів фрейму верхнього рівня до відповідних слотів фрейму нижнього рівня. Розглянемо як приклад структуру фрейму системи FMS [37]. Тут вигляділяють наступні типи покажчиків спадкування: unique, same, range, override і ін.

Unique, (унікальний) означає, що слоти з тими самими іменами у фреймах, що знаходяться на різних рівнях ієрархії, можуть мати різні значення. **Same, (такий же)** показує, що слоти повинні мати однакові значення. **Range (діапазон)** установлює деякі границі, у яких може знаходиться значення слота фрейму нижнього рівня. **Override** комбінує властивості покажчиків **unique** і **same.** При описі слотів фрейму в системі FMS вказується також тип даних слота. Типи даних слотів відповідають типам даних, що застосовуються у мовах програмування високого рівня. Наприклад, INTEGER (цілий), REAL (дійсний), BOOL (логічний), STRING (стрічковий). Однак застосовуються також і інші типи даних, що підвищують ефективність роботи із системою. Наприклад, TEXT (текстовий), TABLE (табличний),

LIST (обліковий), LISP (приєднана процедура), FRAME (покажчик на інший фрейм) і ін.

Узагальнюючи сказане, фреймову структуру даних системи FMS можна спрощено описати у вигляді формул:

<фрейм>::=<ім’я фрейму> <посилання на суперклас> <слот> {<слот>}

<слот>::=<ім’я> <покажчик спадкування> <тип даних> <значення> [<процедури-демони>]

Покажчик спадкування>::= unique| same| range| override

<тип даних>::= INTEGER| REAL| BOOL| STRING| TEXTI TABLE| LISP) LIST] FRAME

<процедури-демони>::=<якщо-додано>| <якщо-видалено>| <якщо-треба>|

Іншим прикладом фреймової системи є мова FRL (Frame Representation Language), що представляє розширення мови Лісп. Об’єкти предметної області в мові FRL називаються фреймами, а їх властивості – слотами. Слот містить трохи “фацет” (англ. facet – аспект, грань, сторона), що

характеризуються значеннями. Так, пропозиція “Кінь сірого кольору” можна представити мовою FRL у вигляді: frame Кінь, slot колір, facet $value, value сірий. До складу слота входить шість фацет:

$value – поточне значення слота;

$require – припустимі значення фацета value;

$default – значення за замовчуванням для фацета value;

$if-added – процедура-демон “якщо-доданий”;

$if-removed – процедура-демон “якщо-вилучений”;

$if-needed – процедура-демон “треба”.

Кожен фрейм, що створюється мовою FRL, містить попередньо визначений слот ako, що дозволяє успадковувати значення слотів відповідно до ієрархії фреймів.

У моделях представлення знань фреймами поєднуються переваги декларативного і процедурного підходів представлення знань. Фреймовий

підхід широко використовується на практиці, про що свідчить наявність розвинутих фреймових мов, наприклад, GUS, KRL, CONSUL і ін. Однак у фреймових системах відсутній спеціальний механізм керування виведенням. Тому вивід звичайно реалізується за допомогою процедур, вбудованих у фрейми.

3.5.2. Керування виведенням

В фреймових системах використовується три способи керування виведенням [37]: за допомогою механізму спадкування; за допомогою процедур-демонів; за допомогою приєднаних процедур.

Механізм спадкування є основним вбудованим засобом виводу, яким оснащуються фреймові системи. Він забезпечує значну економію пам’яті й автоматичне визначення значень для слотів фреймів нижніх рівнів.

Фреймові системи також оснащуються набором спеціальних процедур, до яких відносять процедури: конструювання класу, конструювання екземпляра класу, запису значення в слот, читання слота. Процедура конструювання

класу формує фрейм-прототип з необхідним набором слотів і відповідними посиланнями на суперкласи. Фрейм може бути зв’язаний з декількома суперкласами. Процедура конструювання екземпляра класу дозволяє формувати фрейми-приклади. Вона автоматично встановлює зв’язок усіх таких фреймів з відповідним класом за допомогою is a слота. Процедури запису і читання значень слотів здійснюють доступ до слотів відповідних фреймів і дозволяють користувачу ввести чи визначити значення відповідного слота. Для цього, при їх виклику, їм передається ім’я фрейму й ім’я відповідного слота. У випадку якщо значення якого-небудь слота при виклику відповідної процедури конструювання не задається, то автоматично викликається процедура, що дозволяє встановити значення слота за замовчуванням. Це часто виконується за допомогою механізму спадкування. Розглянемо деякі особливості функціонування механізму спадкування.

На рисунку 3.18 зображена найпростіша ієрархічна структура, у якій кожен фрейм має тільки один суперклас.


Птахи
АКО: хребетні
Спосіб розмноження: відкладення яєць
Здатність літати: так
Вкриття: пір'я
Пінгвіни
АКО: птахи
Здатність літати: так
Зоне проживання: морський берег
Здатність плавати: так
Зріст: 40.. 120 см
Червона книга: ні
Пінгвін Чиллі-Віллі
is_a: пінгвіни
Вид:
Червона книга:

Рисунок 3.18 – Найпростіша ієрорхія фреймів

Кожен підклас чи екземпляр класу успадковує слоти свого суперкласу. Якщо підклас (екземпляр класу) і суперклас мають слоти, зі співпадаючими іменами, то визначення значень слотів, зроблені всередині підкласу (екземпляра класу), перекривають визначення суперкласу. Наприклад, відповідь на питання: “чи здатний пінгвін Чиллі-Віллі літати?” буде негативним. При пошуку відповіді на це питання фрейм-екземпляр “пінгвін Чиллі-Віллі” успадковує всі “слоти фрейму “пінгвіни”. Значення слота “здатність літати” фрейму “пінгвіни” перекриває значення однойменного слота “птаха”.

У загальному випадку порядок спадкування визначається за допомогою списку передувань. У випадку лінійної ієрархічної схеми цей список формується у відповідності з напрямком is a і ako зв’язків. Для розглянутого випадку його можна записати у вигляді:

(“пінгвін Чиллі-Віллі”, “пінгвіни”, “птахи”).

Більш складна ситуація виникає, якщо фрейм має трохи is a чи ako зв’язків. У цьому випадку кожуть про множинне спадкування. Правила формування

списку передувань у цьому випадку будуть складніші. Припустимо, що пінгвін “Чиллі-Віллі” – це персонаж мультфільму. Тоді фрейм-екземпляр “пінгвін Чиллі-Віллі” буде мати два is a зв’язків (Рисунок 3.19).

Рисунок 3.19 – Множинне спадкування

При спробі встановити місця проживання цього пінгвіна виникає двозначність, що вирішується за допомогою списку передувань, сформованого з урахуванням ієрархії фреймів. Перегляд ієрархії фреймів виконується спочатку в глибину, а потім ліворуч – праворуч. Тоді список передуваньбуде мати вигляд:(“пінгвін Чиллі-Віллі”, “пінгвін”, “птаха”, “персонаж мультфільму”).

При такому способі вирішення невизначеності місця проживання пінгвіна по імені Чиллі-Віллі – це морські узбережжя. Якщо користувача не задовольняє результат спадкування, то можна ввести додатковий клас, явно вказавши значення відповідного слота. Наприклад, можна ввести клас “пінгвіни-мультяшки” (Рисунок 3.20) і явно вказати значення слота ” місце

проживання”. Однак це виключає можливість передачі значення слота за замовчуванням і вимагає додаткову пам’ять [1)].

Рисунок 3.20 – Введення додаткового класу

В реальних ситуаціях взаємозв’язок фреймів виявляється значно складніше, ніж у розглянутих прикладах. Тому правила формування списку передуваньускладнюються.

[1)]Звичайно, у даному прикладі необхідний результат можна одержати простіше. Необхідно лише змінити порядок проходження імен фреймів у слоті is_a фрейму “пінгвін Чиллі-Віллі”. Тобто порядок зліва направо відповідає порядку проходження імен фреймів у слоте is_a чи ako.

У цьому випадку, поряд з пошуком в глибину і зліва направо, застосовується принцип виключення з списка передуваньелементів, що повторюються. В

списку завжди залишається тільки та повторювана вершина, що зустрілася останньої. Наприклад, для малюнка 3.21 список передуваньможна представити у вигляді:

(екземпляр, суперклас 2, суперклас 4, суперклас 3, суперклас 5, суперклас 1, універсальний клас).

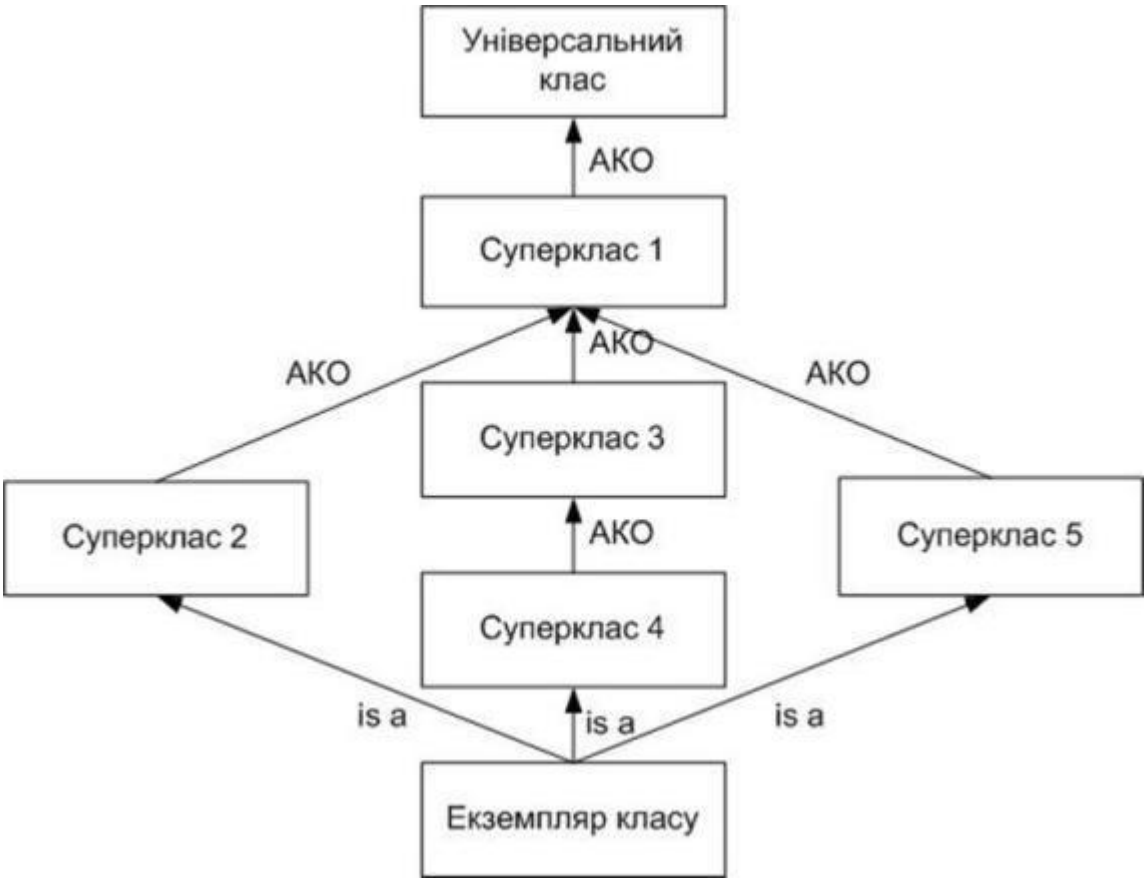

Універсальний клас
АКО
Суперклас 1
АКО
АКО
АКО
Суперклас 3
Суперклас 2
Суперклас 5
АКО
Суперклас 4
is a
is a
is a
Екземпляр класу

Рисунок 3.21 – Ієрархія фреймів з Декількома is_a відношеннями

Даний принцип дозволяє реалізувати вимогу, що полягає в тому, що будь-який клас повинний з'являтися в списку передуваньраніше, ніж його безпосередній суперклас [94]. Реалізація цієї вимоги забезпечує передачу за замовчуванням найбільш специфічних даних, що відповідає нашому інтуїтивному представленню про спадкування властивостей. Разом з тим застосування розглянутої процедури спадкування до ієрархії фреймів, зображеної на рисунку 3.22, приведе до формування наступного списку передувань:

(екземпляр, суперклас 5, суперклас 3, суперклас 6, суперклас 2, суперклас 4, суперклас 1).

У цьому списку порушена вимога, що полягає в тому, що будь-який безпосередній суперклас деякого класу повинний з'явитися в списку передуваньраніше, ніж інший безпосередній суперклас цього ж класу, розташований правіше (суперклас 2 слідує після суперкласу 3). Щоб задовольнити зазначену вимогу, необхідно використовувати процедуру топологічного сортування [94].

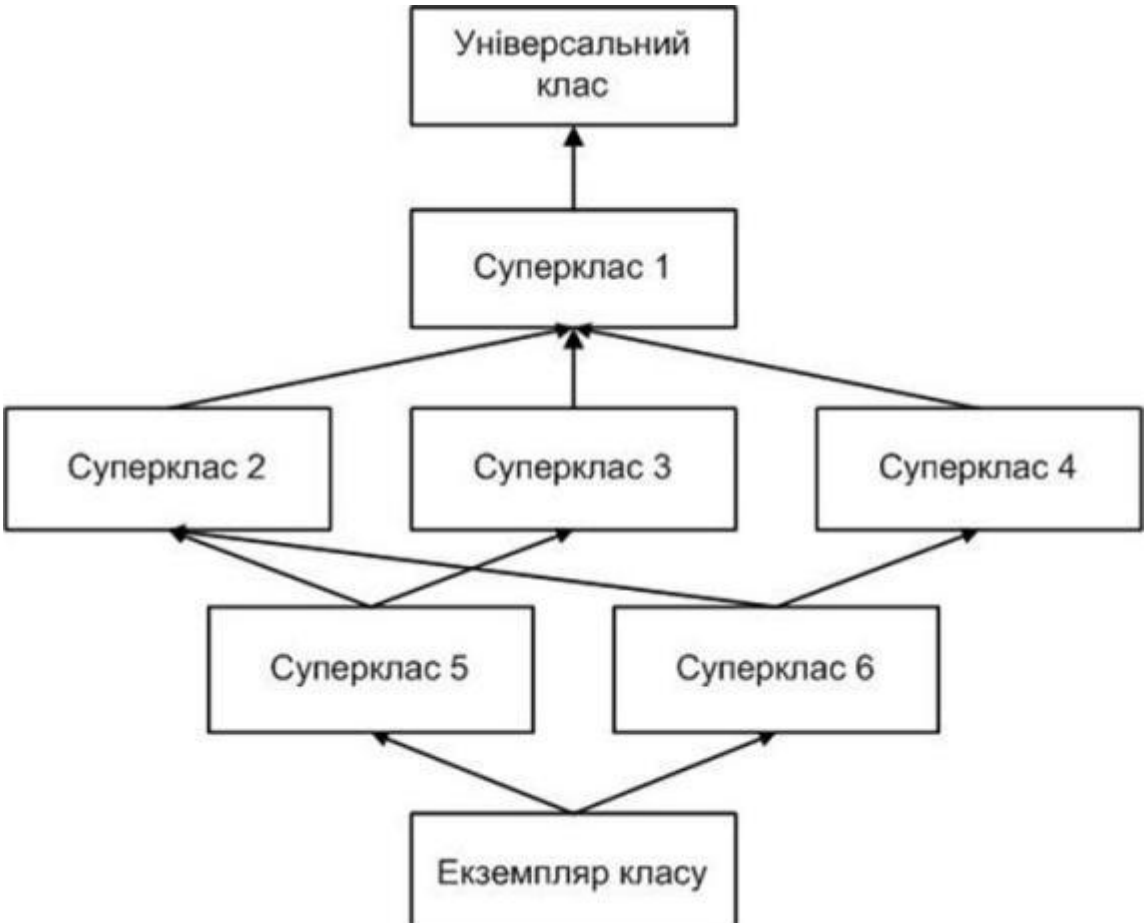

Універсальний клас
Суперклас 1
Суперклас 2
Суперклас 3
Суперклас 4
Суперклас 5
Суперклас 6
Екземпляр класу

Рисунок 3.22 – Ієрархія фреймів з декількома ako відношеннями

Ідея такого сортування полягає в наступному. На першому кроці складається список екземплярів і суперкласів, що (для розглянутого фрейму) досяжні за допомогою is_a і ako зв'язків. Для розглянутого прикладу (Рисунок 3.22) одержимо

(екземпляр, суперклас 5, суперклас 2, суперклас 1, суперклас 3, суперклас 6, суперклас 4, універсальний клас).

На другому кроці для кожного елемента отриманого списку формується список пар відповідно до принципу, зображеному на рисунку 3.23. Даний принцип називають принципом "рибальського гачка", тому що схема об'єднання суперкласів і екземплярів у відповідні пари нагадує нанизування наживки на рибальський гачок. Слідуючи зазначеному принципу, для кожного елемента списку одержимо список відповідних пар (таблиця 3.2).

На третьому кроці переглядають елементи таблиці 3.2 і виділяють елемент, що зустрічається тільки з лівої сторони пари і ніколи не зустрічається з правої сторони. Таким елементом є "екземпляр". Будемо називати цей елемент поточним. Поточний елемент додається в кінець списку передувань,

і всі пари, в яких він зустрічається, викреслюються. В розглянутому випадку це пари “єкземпляр-суперкласс 5”. Потім дії повторюються. Поточним елементом стає суперклас 5, і викреслюються пари “суперклас 5 – суперклас 6” і “суперклас 5 – суперклас 2” і т.д. У підсумку формується наступний список передувань:

(екземпляр, суперклас 5, суперклас 6, суперклас 2, суперклас 4, суперклас 3, суперклас 1, універсальний клас).

Рисунок 3.23 – Принцип “рибальського гачка”

Якщо на деякому кроці описаної процедури на роль поточного елемента претендують відразу кілька суперкласів, то конфлікт вирішується на користь того суперкласу, що є безпосереднім суперкласом для того суперкласу, що знаходиться найправіше і, що вже знаходиться в списку передувань. У розглянутому прикладі конфлікт виникає між суперкласом 3 і суперкласом 4 після того, як у список передуваньбуде поміщений суперклас 2. Перевага віддається суперкласу 4, тому що він є суперкласом найправішого суперкласу, що вже знаходиться в списку передувань, тобто суперкласу 6.

Вивід значень слотів на основі спадкування є, по своїй суті, немонотонним, тому що перекриття значень у процесі спадкування може змінювати істинність раніше установлених фактів. Наприклад, з малюнка 3.18 випливає, що пінгвін Чиллі-Віллі вкритий пір'ям. Однак, якщо ми додамо слот "вкриття" зі значенням "пух" у фрейм "пінгвіни", то вихідне твердження вже буде неправильним.

Інша можливість виводу у фреймових системах заснована на використанні процедур-демонів: " якщо-треба", "якщо-додане", "якщо- вилучене".

Процедура " якщо-треба" викликається, коли надходить запит, що вимагає встановлення значень відповідного слота. Дана процедура, як і значення

слота, успадковується підкласами й екземплярами класів. Тому зв’язування її з одним із суперкласів може вплинути на значення відповідних слотів фреймів, розташованих нижче по ієрархії. При цьому процедура безпосередньо не змінює значень слота, а щораз обчислює його знову, з огляду на визначені умови. Зокрема , для слота “зріст” екземпляра фрейму “пінгвін” (Рисунок 3.18) процедура може встановлювати зріст пінгвіна в залежності від значення слота “вид”. Наприклад, якщо значенням слота “вид” є “імператорський пінгвін”, те повернути значення 100 – 120см, інакше повернути значення 40-50см.

Процедури “якщо-додане” і “якщо-вилучено” активізуються при записі і видаленні значень зі слота. Наприклад, процедура “якщо-додане” може активізуватися, коли слот “вид” одержить значення (Рисунок 3.18). Якщо нове значення – “галапагосський пінгвін”, то записати в слот “червона книга” значення “так”. Очевидно, що дана процедура встановлює обмежуючі стосунки між значеннями слотів “вид” і “червона книга”. Можна сказати, що процедури “якщо-додане” і “якщо-вилучене” встановлюють залежність значень одного слота від значень іншого і відіграють роль хранителя відносин обмеження у фреймових системах.

Третя можливість виводу у фреймових системах заснована на застосуванні приєднаних (службових) процедур. Ім’я приєднаної процедури виступає як значення слота. Приєднана процедура запускається за повідомленням, переданим з іншого фрейму. Для цього може використовуватися спеціальна функція MSG, формат виклику якої має вид:

MSG((<ім’я фрейма>, <ім’я слота>,<параметри>).

Дана функція забезпечує запуск приєднаної процедури, що знаходиться у відповідному слоті відповідного фрейму. При цьому приєднана процедура одержує необхідні параметри. У свою чергу, викликана приєднана процедура може також передати повідомлення в інший фрейм і викликати іншу приєднану процедуру і т.д. Виникає ланцюжок передач повідомлень від одного фрейму до іншого і повернення відповідей. У підсумку відповідь повертається в місце первісного виклику функції MSG і повідомляється користувачу. Таким чином, у фреймових системах на кожний із фреймів за допомогою приєднаних процедур покладаються визначені обов’язки, і забезпечується їхнє погоджене виконання.

На закінчення відзначимо, що якщо у фреймових системах не використовувати процедури-демони і приєднані процедури, те фреймова

модель представлення знань буде відповідати семантичній мережі. Використання процедурних знань у фреймових системах підвищує їх гнучкість, але ускладнює керування. Свій подальший розвиток фреймові системи одержали в системах об’єктно-орієнтованого програмування. У главі 5 ми розглянемо систему об’єктно-орієнтованого програмування мови Коммон Лісп.