Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмного забезпечення комп'ютерних систем

МЕТОДИ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Навчальний посібник
для студентів спеціальності
122 «Комп'ютерні науки»

Дніпро
НГУ
2017

Навчальний посібник «Методи та системи штучного інтелекту» для студентів спеціальності 122 «Комп’ютерні науки» / Уклад.: І.М. Удовик, Г.М. Коротенко, Л.М. Коротенко, В.О. Трусов, А.Т. Харь. – Д.: Державний ВНЗ «Національний гірничий університет», 2017. – 105 с.

Укладачі: І.М. Удовик, Г.М. Коротенко, Л.М. Коротенко, В.О. Трусов, А.Т. Харь.

(Надано гриф «Рекомендовано Вченою радою ДВНЗ «НГУ» як навчальний посібник для бакалаврів та магістрів спеціальності 122 «Комп’ютерні науки». Протокол № 11, від 26 червня 2017 р.)

Відповідальна за випуск: завідувач кафедри ПЗКС
к.т.н., доцент Удовик Ірина Михайлівна.

ЗМІСТ

Вступ

Багато проблем з області штучного інтелекту і штучного розуму самі по собі виходять за межі інформатики. Перш за все, виникає питання «a thinking machine?», Тобто вміють (і чи можуть вміти) комп'ютери мислити як людина або ні?

Реферативний журнал «Abstracts in Artificial Intelligence» (The Turing Institute, ed. J. Ritchie) розділяє штучний інтелект на наступні проблеми: експертні системи, застосування штучного інтелекту, автоматичне програмування, автоматичний доказ теорем та логічне програмування, машинне навчання, природна мова, пошук, управління і планування, робототехніка, зір і обробка зображень, розпізнавання образів, когнітивне моделювання, взаємодія людини та комп'ютера, технічні засоби для штучного інтелекту. Але що об'єднує ці настільки різні завдання?

Пропоновані визначення інтелекту несхожі один на одного настільки, ніби мова йде про різні речі. Одні вважають, що інтелект – це вміння вирішувати складні завдання; інші розглядають його як здатність до навчання, узагальнень до знаходження аналогій; треті – як можливість взаємодії із зовнішнім світом шляхом спілкування, сприйняття і усвідомлення сприйнятого. Деякі вчені розвивають навіть теоретичну модель, в якій за здійснення інтелектуальної діяльності відповідає близько 120 різних чинників, з яких тільки 50-60 сьогодні відомі.

Разом з тим, на даний час не існує єдиного і визнаного усіма точного і всеосяжного визначення передового наукового напрямку, що має назву ***«штучний інтелект»*** (ШІ), як і немає універсального визначення поняття людського інтелекту.

Вікіпедія, визначає область досліджень і застосувань ШІ, (англ. *Artificial Intelligence*, AI), в такий спосіб.

Штучний інтелект – це:

1) наука і технологія створення інтелектуальних машин, особливо інтелектуальних комп'ютерних програм;

2) властивість інтелектуальних систем виконувати творчі функції, які традиційно вважаються прерогативою людини.

Серед багатьох точок зору на штучний інтелект домінують три. Згідно з першою, дослідження в галузі ШІ є фундаментальними дослідженнями, в рамках яких розробляються моделі і методи розв'язання задач, що традиційно вважаються інтелектуальними і не піддавалися раніше формалізації і автоматизації. Відповідно до другої точки зору новий напрямок пов'язано з новими ідеями вирішення завдань на комп'ютерах, з розробкою принципово інших технологій програмування і створенням нових архітектур, що відкидають класичну архітектуру, яка відноситься ще до перших розробок обчислювальної техніки. Нарешті, третя точка зору, мабуть, найбільш прагматична, полягає в тому, що в результаті робіт в області ШІ народжується безліч прикладних систем, що вирішують завдання, для яких раніше створювані системи були непридатні.

Завданням штучного інтелекту як науки є відтворення за допомогою штучних пристроїв (в основному ноутбуків, планшетів, смартфонів, пристроїв що носяться, таких як годинники, брелки та ін., а також робототехнічних пристроїв: домашніх роботів, безпілотних автомобілів та ін.) розумних дій та міркувань. Вивчення теоретичних основ даної науки дозволяє відшукувати найбільш доцільні шляхи розробки нових інформаційних технологій і вирішення перспективних інтелектуальних завдань.

У дисципліні «Методи і системи штучного інтелекту» вивчаються основи представлення знань в сучасних комп'ютерах, методологія математичного моделювання та автоматизації логічного та аналітичного мислення, відомі системи штучного інтелекту. Навчальний посібник з цього курсу складається з семи розділів. У першому розділі визначаються основні завдання досліджень в області ШІ. Другий розділ присвячений вивченню основних методів і моделей представлення знань в комп'ютерах. У третьому розділі викладаються основи логіки висловлювань і методи вирішення завдань в даній області знань. У четвертому розділі розглядаються основи логіки предикатів і наводяться приклади розв'язання задач методом резолюцій. П'ятий розділ присвячений розгляду етапів проектування систем штучного інтелекту і розробки експертних систем. У шостому розділі викладаються стратегії створення експертних систем на основі продукційних моделей подання знань і особливостей побудови програм машинного виведення. У сьомому розділі розглядаються підходи, які використовуються при розробці та особливості функціонування агентів і систем, побудованих з їх використанням.

Наведені в навчальному посібнику матеріали з теорії та практики ШІ допоможуть у вивченні курсу «Методи і системи штучного інтелекту» і забезпечать закріплення знань в ході виконання лабораторних робіт.

1. Цілі і завдання досліджень у галузі штучного інтелекту

Крім класичних застосувань обчислювальної техніки, пов'язаних з виконанням інженерних і економічних розрахунків, розробкою автоматизованих систем управління, створенням інформаційно-пошукових систем і т.д., в даний час активно розвивається напрямок, який називається ***«штучний інтелект»*** (ШІ). Будь-яка задача, для якої алгоритм вирішення невідомий, апріорно відноситься до області штучного інтелекту.

В даний час виділяють чотири основні напрямки, за якими ведуться дослідження в галузі ШІ [1 - 6, 11]:

1) моделювання окремих функцій творчих процесів;

2) зовнішня інтелектуалізація комп'ютерів;

3) внутрішня інтелектуалізація комп'ютерів;

4) цілеспрямована поведінка роботів та різноманітних електронних пристроїв (смартфонів, гаджетів та ін.).

Перший напрямок почав розвиватися в ШІ раніше за інші; саме він і породив цей термін: моделювання на комп'ютерах окремих функцій творчих процесів (гра в шахи, шашки, доміно та ін., автоматичний доказ теорем, автоматичний синтез програм, аналіз і синтез музичних творів, автоматичний переклад, розпізнавання образів і ін.).

Другий напрямок утворюють фундаментальні і прикладні дослідження, спрямовані на підвищення рівня взаємодії людини з комп'ютерними системами і пристроями в рамках вдосконалення функцій діалогового інтерфейсу. Інтелектуальний інтерфейс виводить на новий рівень ефективність використання автоматизованих систем управління (АСУ), систем автоматизованого проектування (САПР), автоматизованих систем наукових досліджень (АСНД) і оперативного управління виробництвом в цілому.

Промислові роботи стали не тільки однією з рушійних сил автоматизації, а й найважливішим засобом здійснення глибоких соціально–економічних змін в сфері праці. Розробка і виробництво промислових роботів з високим рівнем інтелектуальності та багатофункціональності, які мають вражаючу надточність, дозволили підняти продуктивність праці на небувалу висоту. Кількість промислових роботів, що випускаються фірмами Японії (FANUC, KAWASAKI, MOTOMAN, OTC DAIHEN, PANASONIC), Німеччини (KUKA), США (KC ROBOTICS, TRITON MANUFACTURING, KAMAN CORPORATION), Швеції (ABB) і ряду інших країн вже перевищує сотні тисяч найменувань (рис. 1.1).

Використовуючи акумульовані в комп'ютерах і базах даних знання про розвиток предметних областей, які їх цікавлять, фахівці багатьох галузей науки отримують можливість, не виходячи за межі мови своїх особистих предметних областей[1] (підмов природної мови), здійснювати розпізнавання

[1]**Предметна галузь** – сукупність об’єктів реального або уявного світу, що розглядаються в рамках даного контексту, який розуміється як окреме припущення, фрагмент наукової теорії або теорія в цілому і обмежується рамками інформаційних технологій вибраної галузі.

та діагностику процесів в складних системах, приймати оптимальні рішення, формулювати плани дій, висувати гіпотези, а також виявляти закономірності в результатах спостережень. Ці можливості реалізуються експертними системами [2–3], які стали інтенсивно поширюватися в важко формалізованих галузях знань, наприклад, медицині, освіті, технічних дисциплінах і т.д. (рис. 1.2).

Рис. 1.1. Роботи японської компанії FANUC, призначені для зварювання, навантаження, сортування, транспортування та інших точних робіт

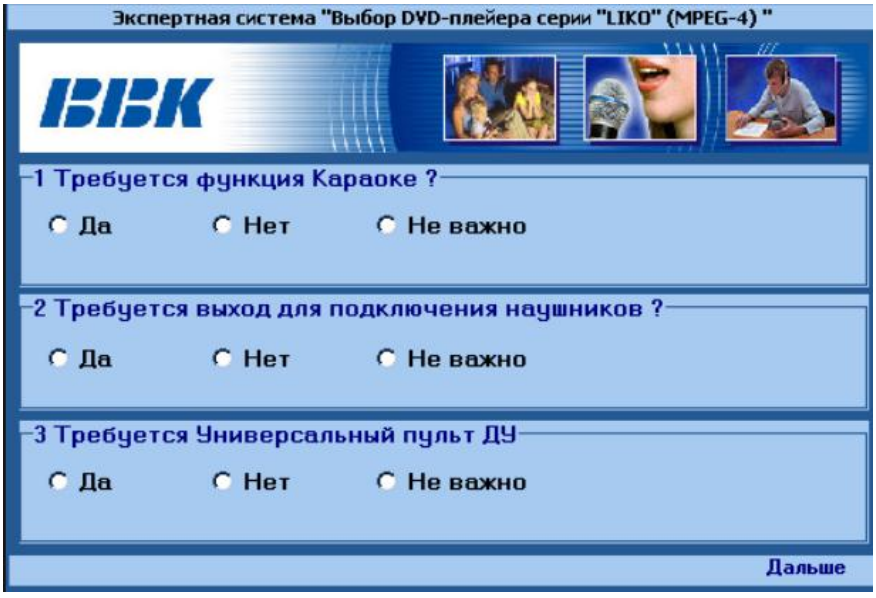

Рис. 1.2. Приклад інтерфейсу експертної системи для вибору DVD–програвача

Третій напрям використання штучного інтелекту вирішує проблеми побудови комп'ютерів нових поколінь, оскільки для вирішення завдань створення застосувань ШІ важливі і самі апаратні засоби, і нові методи обробки символьної інформації. Як правило, в подібних системах використовується інформація, представлена в символьній формі: літери, слова, знаки, рисунки. Це відрізняє область штучного інтелекту від областей, в яких традиційно комп'ютерам довіряється обробка даних в числовій формі. У системах ШІ передбачається наявність вибору між багатьма варіантами можливих рішень в умовах невизначеності, що вимагає принципово нових архітектурних побудов обчислювальних пристроїв і програмних компонентів для управління ними.

Так, наприклад, для операційної системи iOS смартфонів iPhone (Apple) розроблений персональний помічник і питально–відповідаюча система Siri (sɪəri; рос. Сірі, від англ. **S**peech **I**nterpretation and **R**ecognition **I**nterface – інтерфейс розпізнавання і інтерпретації мови). Цей застосунок використовує обробку природної мови, щоб відповідати на питання і видавати рекомендації. Крім того, Сірі пристосовується до кожного користувача індивідуально, вивчаючи його вподобання та телефонні контакти протягом довгого часу (рис. 1.3).

Рис.. 1.3. Приклад результату взаємодії користувача смартфону з персональним помічником Сірі

Четвертий напрямок додатків ШІ пов'язано з створення інтелектуальних роботів для різних областей діяльності людини. Ця науково-технічна проблема вимагає розробки, як спеціалізованих обчислювальних засобів, так і цілого комплексу складних технічних, механічних, програмних

і енергетичних систем: сенсорів, рушіїв і т.д. Як і всі системи ШІ, інтелектуальні роботи орієнтовані на використання знань. Наприклад, знання про зовнішнє середовище надходять в бортові комп'ютери роботів від численних сенсорів (зорових, акустичних, радіолокаційних, тактильних і т.д.) (рис. 1.4).

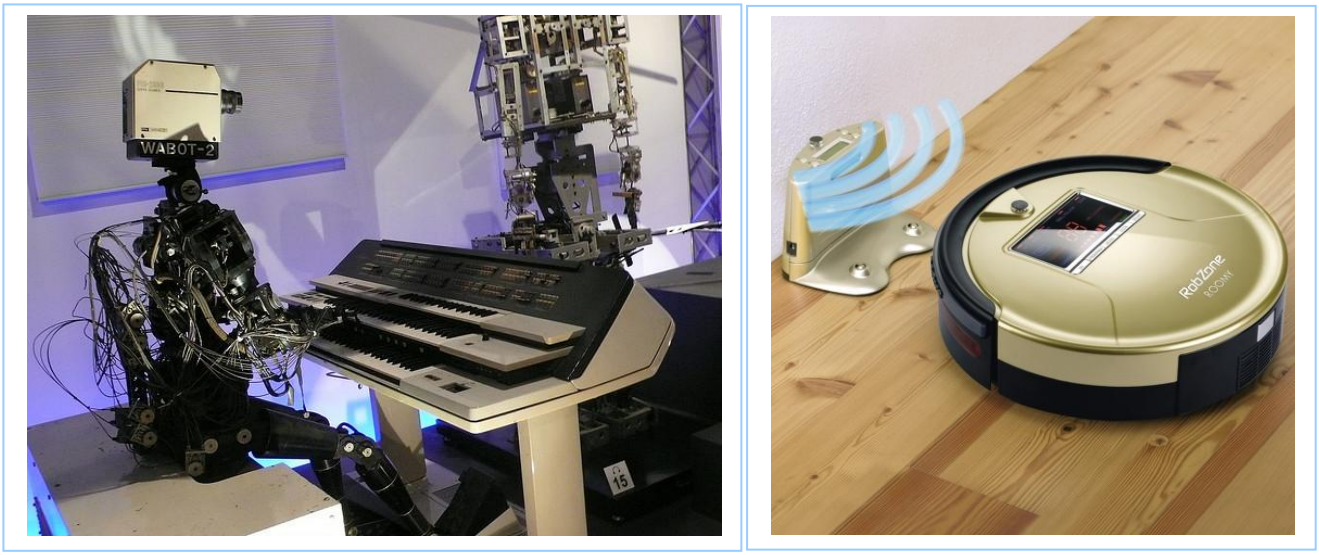

Рис.. 1.4. Робот–піаніст (зліва) і робот–пилосос (праворуч)

Штучний інтелект, як основа нової інформаційної технології, примножує інтелектуальні ресурси суспільства, оскільки взаємодія користувача з обчислювальними пристроями своєю професійною мовою підвищує рівень інтелекту користувача і розширює його можливості в сфері формування нових елементів логічного висновку. І якщо раніше комп'ютери були основою індустрії обробки даних, то зараз, у зв'язку з активним використанням ідей і методів ШІ, стало правомірним говорити про створення на базі комп'ютерних пристроїв індустрії робототехніки і технологій проектування інтелектуальних систем.

Зазначимо основні галузі застосування штучного інтелекту [2, 4, 13].

1. Використання дедуктивних міркувань (інакше званих *численнями*) при вирішенні за допомогою кібернетичних систем інтелектуальних завдань, тобто задач, які не мають апріорі відомих алгоритмів вирішення.
2. Автоматичний доказ теорем в аксіоматичних теоріях.
3. Розпізнавання та розуміння мови і текстів, зорових образів.
4. Розуміння процесу навчання і автоматизоване формування сценаріїв взаємодії між учнем та навчальним.
5. Автоматизоване проектування інтегральних (інтелектуальних) роботів.
6. Рішення екстремальних задач при нечітких і неповних цільових функціях.
7. Проектування систем взаємодії людини і баз даних і знань при їх створенні та експлуатації.
8. Розробка експертних систем для різних предметних областей.

9. Проектування систем прийняття рішень для некоректних задач і завдань з нечіткою постановкою.

10. Автоматизоване проектування інтерактивних графічних систем зі структурованими графічними операндами високого рівня.

2. Представлення знань в системах штучного інтелекту

В основі розробки і використання обчислювальної техніки традиційно лежать такі поняття, як ***програми*** і ***дані***. При цьому перші призначені для обробки другого. На ранніх етапах розвитку даної галузі програміст, як правило, сам розробляв програму і сам вводив в неї необхідні дані.

Потім відбулася серйозна зміна в їх взаємодії – з появою ***баз даних*** різної структури (ієрархічних, мережевих, реляційних, об'єктно-орієнтованих) і систем управління базами даних (СКБД) ***дані*** були відокремлені від ***програм***. Для вирішення такого завдання використовувалися засоби опису даних, що містяться в мовах програмування. Такі мови, як ФОРТРАН і АЛГОЛ, мали засоби опису відносно простих структур даних в пам'яті електронно-обчислювальних машин (ЕОМ). Наприклад, цілі – INTEGER, дійсні – REAL і т.д. Більш складні засоби опису ієрархічних структур даних вбудовані в мови КОБОЛ, СІ, ПАСКАЛЬ. Модула-2, АДА і ін. В цих мовах також є засоби для конструювання структур даних самим користувачем.

Паралельно з вищевказаними процесами розвивалися концепції представлення даних у зовнішній пам'яті великих за розмірами ЕОМ, які паралельно з удосконаленням компонентів елементної бази і переходу від ламп, діодів і тріодів до інтегральних мікросхем трансформувалися в невеликі і компактні персональні комп'ютери (рис. 2.1).

Рис. 2.1. ЕОМ БЕСМ–6 (зліва) і персональний комп'ютер (праворуч)

Іншим компонентом, що дозволив остаточно відокремити дані від програм, став пристрій, який називається магнітний диск.

Революційність його появи пояснювалася тим, що після знеструмлення ЕОМ всі дані, присутні в її пам'яті, пропадали.

Магнітний диск забезпечив зберігання інформації і після вимикання комп'ютера. Більш того, накопичувач на жорстких магнітних дисках (НЖМД) може бути витягнутий з одного комп'ютера і підключений до іншого, де він може надавати доступ не тільки до записаних на ньому даних, але й до операційних систем і прикладних програм, записаних на ньому до цього (рис. 2.2).

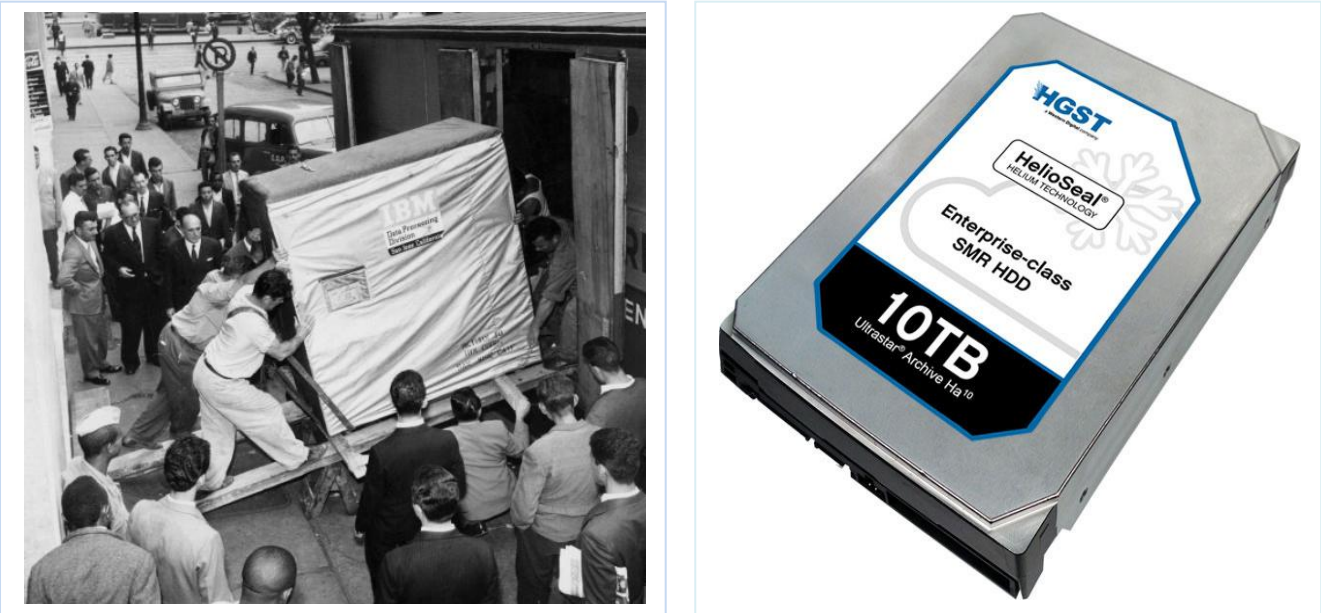

Рис. 2.2. Завантаження для перевезення 5-мегабайтного жорсткого диска компанії IBM, 1965 рік (США) (ліворуч) та 10-ти терабайтний диск фірми Western Digital, 2016 рік (праворуч)

На цьому етапі фундаментальним поняттям інформатики стало поняття інформаційного масиву – ***файлу***, що є спеціально організованою структурою даних, яка розпізнається комп'ютером як єдине ціле, має ім'я і містить всі необхідні структуровані записи даних про різні об'єкти, з якими веде роботу комп'ютер. Файл можна трактувати як інформаційну модель деякого об'єкту, наприклад, програми, документа, таблиці, музичного твору і ін. (рис. 2.3).

Таким чином, уявлення даних у зовнішній пам'яті комп'ютера пройшло через три етапи:

1) на першому етапі, способи формування записів даних в файлах, ведення файлів і організація доступу до них повністю визначалися в конкретних програмах користувача;

2) на другому етапі, управління файлами і організацію доступу до них стали здійснювати операційні системи комп'ютерів;

3) на третьому, файли стали елементами створюваних баз даних і розвинених систем управління ними (т.зв. СУБД – системи управління базами даних).

При цьому, була реалізована можливість ефективної роботи з великими базами даних (зокрема, з інтегрованими базами, що містять різнорідні дані), що обробляються в інтересах цілого підприємства, галузі і т.д., і призначеними для використання в прикладних задачах.

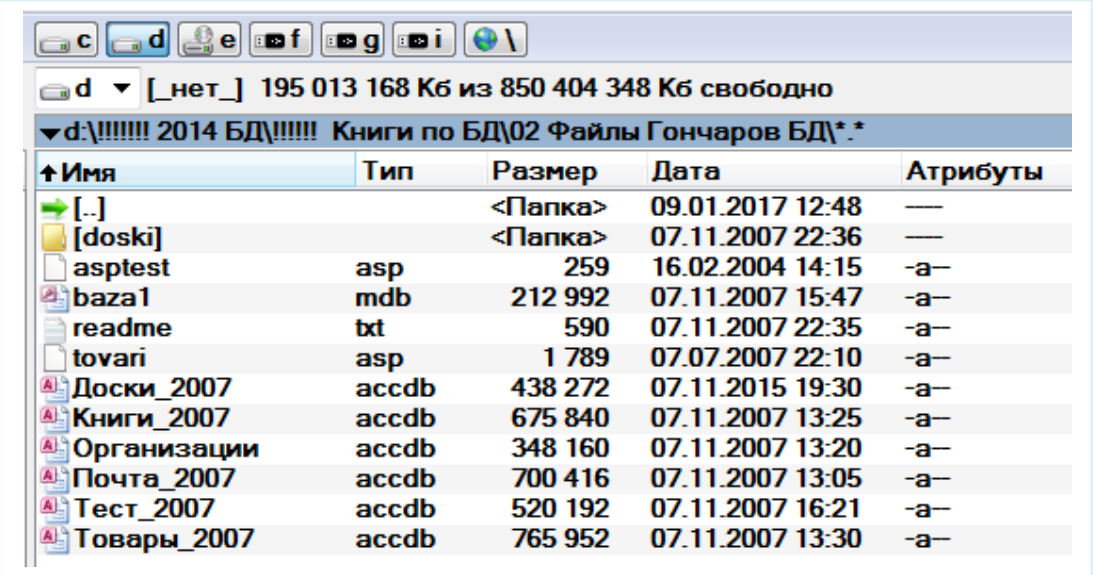

Рис.. 2.3. Представлення в вікні файл–менеджера файлів, що відносяться до бази даних організації

Таким чином, на першому етапі розвитку методів обробки даних створення, підтримка і організація доступу до них, як на логічному, так і на фізичному рівнях, цілком покладалися або на розробника, або на користувача кожної окремої програми. Робота з даними, що відносяться до конкретної програми, а тим більше їх використання в інших програмах були вкрай трудомісткими і малоефективними.

Поліпшення становища сталося на другому етапі, коли частина турбот, що були пов’язані з обробкою даних у зовнішній пам'яті комп'ютерів (в основному на фізичному рівні) взяла на себе операційна система (ОС).

Однак робота з інтегрованими даними стала реальністю лише на третьому етапі розвитку інформаційних систем (ІС), коли з'явилася можливість ефективно організовувати бази даних зі складною структурою, а в рамках СУБД були розроблені потужні засоби для роботи з наборами різноманітних відомостей про навколишній світ. Це зробило виправданим і ефективним існування в інформаційних системах наборів даних, незалежних від прикладних програм, в яких ці дані створюються і використовуються, а також дозволило технологічно відокремити один від одного різні програми (що створюють, підтримують і використовують дані). З'явилася можливість ефективно зв'язувати програми для обробки даних з самими цими даними і вже викликати програми на основі і виходячи з існуючих даних, а не навпаки, як було раніше. Для забезпечення функціонування величезних масивів даних, накопичених і продовжують накопичуватися організаціями та науково–дослідними інститутами в даний час використовуються т.зв. центри

обробки даних (дата-центри, серверні ферми, «хмарні» дата центри і т.д.) (рис. 2.4).

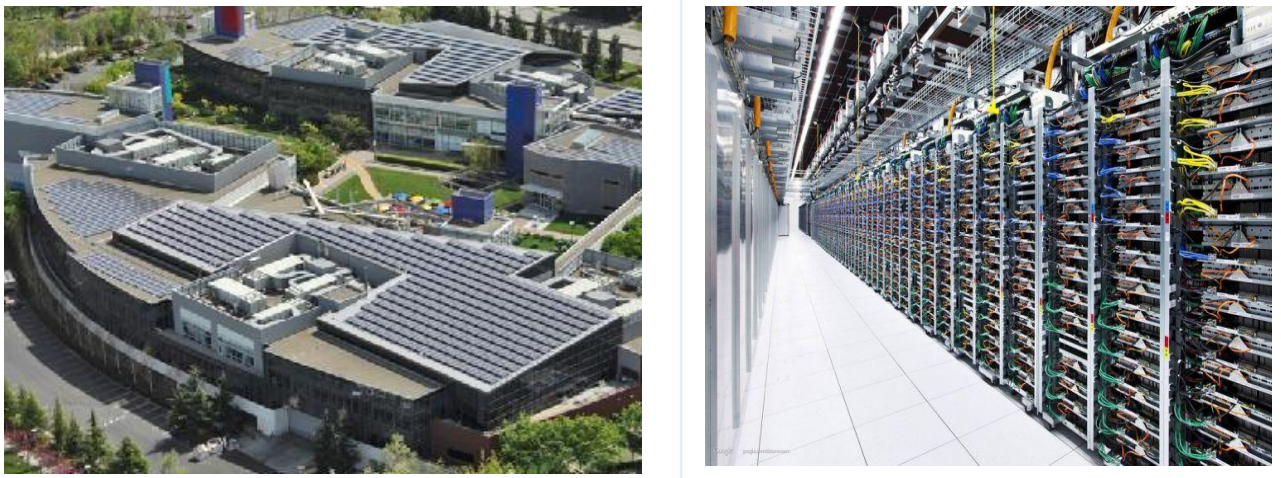

Рис. 2.4. Зовнішній (зліва) і внутрішній вигляд, зі стойками (шафами) дискових накопичувачів (праворуч), одного з дата-центрів компанії Google

І, нарешті, СУБД забезпечили засоби для створення в кожному програмному комплексі проміжного шару програмного забезпечення, що відокремлює власне прикладні програми від використовуваних даних, ефективно реалізує пошук, розміщення та інші операції над даними і тим самим звільняє від цієї діяльності прикладних програмістів. Проміжний шар частково складається з системних програм СУБД, частково добудовується користувачем. Засоби для цього надаються спеціалізованими мовами опису даних і мовами маніпулювання даними, що включаються в СУБД. Такі мови, як, наприклад, SQL (Structured Query Language – мова структурованих запитів), доповнюють традиційні мови програмування засобами для організації та обробки великих груп (наборів) даних. Щоб забезпечити можливість прямого використання цих засобів в прикладних програмах на традиційних мовах програмування, мова опису даних і мова маніпулювання даними часто оформлюються як розширення розвиненої мови програмування – т.зв. «мови, що включається».

В даний час можна говорити про новий етап представлення даних в пам'яті комп'ютера – про створення інформаційно-обчислювальних мереж і на їх основі – розподілених баз даних колективного користування. Це призводить як до зниження витрат на створення і введення баз даних, так і до підвищення якості інформації, що зберігається, оскільки для ведення баз даних можливо залучення більш кваліфікованих працівників. Одночасно різко зростає доступність цієї інформації для користувачів.

З появою систем ШІ з'явилося нові поняття – ***«знання»*** і ***«база знань»*** (БЗ) [1]. У теорії штучного інтелекту знання – це сукупність інформації про світ, властивості об'єктів, закономірності процесів і явищ, а також правила використання їх для прийняття рішень. Головна відмінність ***знань*** від ***даних*** полягає в їх структурованості і активності.

Поява в базі нових фактів або встановлення нових зв'язків може стати джерелом змін в прийнятті рішень. Виникла необхідність якимось чином співвіднести звичні поняття «дані і бази даних» з поняттями «знання і бази знань». Безсумнівно, що дані і структура бази даних певною мірою відображають знання про предметну область і її структуру. Проте, є специфічні ознаки, що відрізняють знання від даних. Такими специфічними ознаками знань у зв'язку з представленням їх в комп'ютерах виділяють наступні чотири ознаки:

- внутрішня інтерпретованість;
- структурованість;
- зв’язність;
- активність.

Якщо звернутися до наборів даних, то деякі із зазначених ознак, властиві знанням, будуть справедливими і для них. Наприклад, перша ознака – ***інтерпретованість*** – явно проглядається у реляційній базі даних, де імена стовпців є атрибутами відносин, імена яких вказані в рядках. Внутрішня інтерпретованість передбачає можливість установки для елемента даних пов'язаної з ним системою імен. Система імен включає в себе індивідуальне ім'я, яке привласнене даній інформаційній одиниці. Наявність системи «надлишкових» імен дозволяє системі штучного інтелекту знати, що зберігається в її базі знань, а, отже, вміти відповідати на нечіткі питання про вміст бази знань.

Другу ознаку – ***структурованість*** – можна розглядати як властивість декомпозиції складних об'єктів на більш прості і встановлення зв'язків між простими об'єктами, що означає використання відносин «частина-ціле», «клас-підклас», «рід-вид» і т.д. Відносини подібного роду зустрічаються і в ієрархічних, і мережевих базах даних. Ці ж відносини можуть бути реалізовані і в реляційних (табличних) базах даних.

Для третьої ознаки знань – ***зв'язності*** – практично не можна знайти аналогів у звичайних базах даних. Як правило, наші знання пов'язані не тільки у сенсі структури. Вони відображають закономірності організації взаємодії фактів, процесів, явищ і причинно-наслідкові зв'язки між ними. Зв’язність характеризує можливість встановлення між інформаційними одиницями найрізноманітніших відносин (чітких, нечітких, бінарних, складових і ін.), які визначають семантику і прагматику зв'язків явищ і фактів, а також відносин, що визначають зміст даної системи в цілому.

Що стосується четвертої ознаки – ***активності***, то ситуація складається таким чином, що при використанні комп'ютерів – нові знання породжуються програмами (тобто програмно), а дані пасивно зберігаються в пам'яті. Людині властива пізнавальна активність, іншими словами, знання людини активні. І це принципово відрізняє знання від даних. Наприклад, виявлення суперечностей в знаннях стає спонукальною причиною їх подолання і появи нових знань. Таким самим стимулом активності є неповнота знань, що відображається в необхідності їх поповнення. Таким чином, головна відмінність *знань* від *даних* полягає в їх зв'язності і активності, а поява в базі

нових фактів, або встановлення нових зв'язків може стати джерелом змін у прийнятті рішень.

Знання, які використовуються для створення системи ШІ, які забезпечує її роботу, зберігаються, модифікуються і виробляються у ній, можуть бути визначені різноманітним чином. У теперішній час використовують три визначення. Знання – це:

- результат, отриманий пізнавальним чином;
- система суджень з основною і єдиною організацією, основана на об’єктивній закономірності;
- формалізована інформація, на яку посилаються або використовують у процесі логічного виводу.

Процес рішення задач за допомогою найпростішої моделі системи ШІ представлений на рис 2.5.

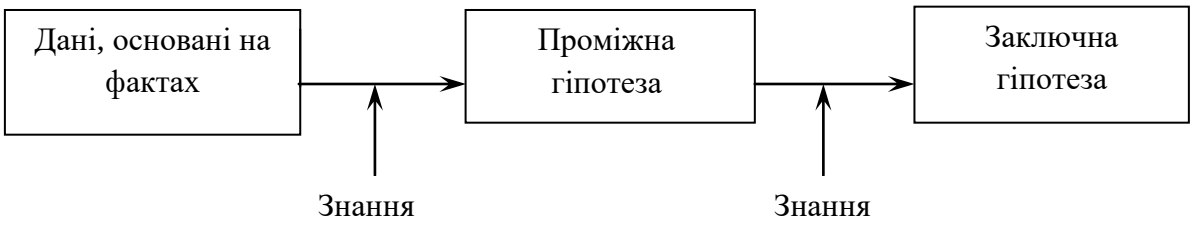

Рис. 2.5. Зв’язок між знаннями і висновком при рішенні інтелектуальної проблеми.

У представленій на рисунку постановці задачі, знання – це інформація, на яку посилаються, коли роблять різноманітні висновки на основі існуючих даних за допомогою ***логічних висновків***. Якщо такі дії виконуються на основі використання програмних засобів, то знання – це обов’язкова інформація представлена у певній формі.

Важливо те, що постановка і рішення будь-якої задачі, пов’язаної з обробкою даних і знань, завжди пов’язані з її «зануренням» у відповідну предметну область.

Предметна область — це частина реального світу, що розглядається у межах заздалегідь визначеного (використовуваного розробником) контексту. Зазвичай це множина всіх предметів, властивості яких і відношення між якими розглядаються у науковій теорії або галузі практичної діяльності спеціаліста. Подумки, предметна область представляється такою, що складається з реальних або абстрактних об’єктів, званих ***сутностями***. Так, наприклад, вирішуючи задачу складання розкладу обробки деталей на металорізальних верстатах, ми залучаємо у предметну область, з одного боку, такі сутності, як конкретні верстати, деталі, інтервали часу їх обробки, а з іншого боку – загальні поняття «верстат», «деталь», «тип верстату» і т.д. Об’єднана сукупність самої предметної області, її уявлень і сутностей, а також задач, що розв’язуються у цій області, визначається поняттям ***проблемна область.***

При цьому слід розуміти, що ***знання*** – це закономірності предметної області (принципи, зв'язки, закони), отримані в результаті практичної діяльності і професійного досвіду, що дозволяють фахівцям ставити і вирішувати завдання в цій галузі.

Важливо також зазначити, що при вирішенні задач в деякій предметній області знань, останню зручно розділити на дві великі категорії – ***факти*** і ***евристику***[2].

Перша категорія вказує, зазвичай, на добре відомі в даній галузі обставини, тому знання цієї категорії іноді називають текстовими, маючи на увазі достатню їх освітленість у спеціальній літературі або підручниках.

Друга категорія знань ґрунтується на власному досвіді фахівця у даній предметній області (т.зв. ***експерта***), і цей досвід накопичений у результаті багаторічної практики. У так званих експертних системах евристичні знання відіграють вирішальну роль у підвищенні ефективності систем. Іншими словами, в цю категорію входять такі знання, як «засоби зосередження», «засоби видалення непотрібних ідей», «способи використання нечіткої інформації» і т.д., що дозволяють з більшою ефективністю вирішувати завдання. Проте, через недостатню наукову обґрунтованість та відсутність вичерпних відомостей користуватися такими знаннями потрібно обачно.

Знання, крім того, можна розділити на ***факти*** (фактичні знання) і ***правила*** (знання для прийняття рішення).

Під фактами маються на увазі знання типу «А це А» і вони характерні для баз даних. Під правилами розуміються знання виду «ЯКЩО А – ТО Б». Крім них існують так звані метазнання (знання про знання). Поняття «метазнання» вказує на знання, які стосуються способів використання знань, і знання, які стосуються властивостей знань. Це поняття необхідно для управління базою знань і логічним висновком, виконання функцій ототожнення, навчання і т.д.

Дані та структури даних далеко не в повній мірі відображають особливості предметних областей. Хоча, взагалі кажучи, чітку грань між даними і знаннями провести можна не завжди, проте, відмінності між даними і знаннями існують, і ці відмінності привели до появи спеціальних формалізмів у вигляді моделей представлення знань в комп'ютерах, що відображають в тій чи іншій мірі основні ознаки, що характеризують знання.

Розвиток методів опису знань привів до створення широкого кола різноманітних моделей, що використовують різноманітні теорії і підходи.

Один з підходів до їх класифікації може виглядати наступним чином [4] (рис .2.6).

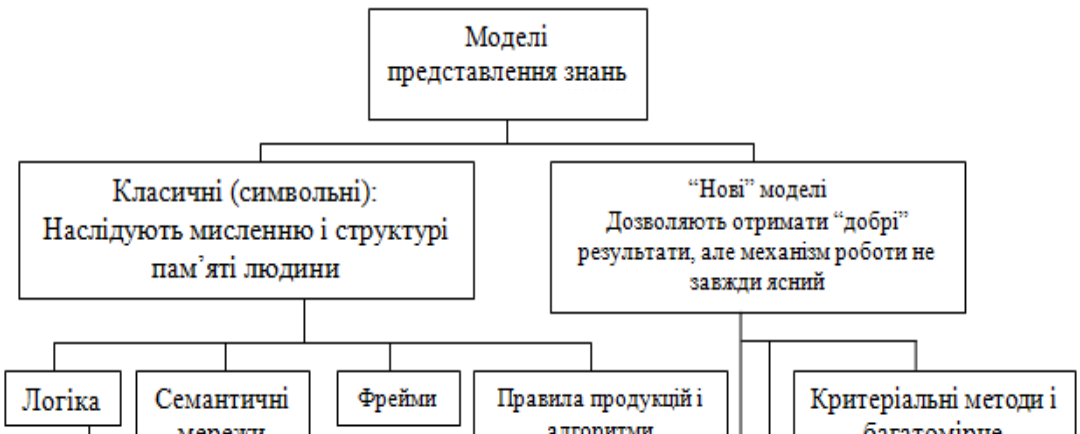

Рис.. 2.6. Класифікація найбільш розповсюджених моделей знань

На теперішній час достатньо широко використовуються, щонайменше, чотири види моделей і відповідно мов уявлення знань [9,11,14]:

1) моделі і мови семантичних мереж;

2) системи фреймів;

3) логічні моделі і мови;

4) продукційні системи.

Розглянемо докладніше ці підходи до опису і уявлення знань в комп'ютерах.

2.1. Семантичні[3] мережі

Під семантичною мережею розуміється інформаційна модель предметної області, що має вигляд орієнтованого графа, вершини якого відповідають об'єктам предметної області, а дуги (ребра) задають відносини між ними. Об'єктами можуть бути поняття, події, властивості і процеси.

Перші мережеві моделі з'явилися в 60-і роки минулого століття. Прикладами їх можуть служити RX-коди, синтагматичні ланцюги, і, нарешті, семантичні мережі.

В основі найбільш відомої семантичної моделі лежить поняття мережі, утвореної позначеними вершинами і дугами.

[3] Семантика, у вузькому сенсі слова – змістовна (смислова) сторона мовних одиниць, їх значення. Кожна мовна одиниця має свою семантику: найбільш елементарна семантика морфем, вона усвідомлюється в складі слів, що їх включають (і означає «людина, прихильна до чогось»: атеїст, антиглобаліст). Семантика, в широкому сенсі слова – аналіз відносин між мовними виразами і світом, реальним або уявним, а також саме це відношення (порівняйте: вираз типу семантика слова) і сукупність таких відносин (так, можна говорити про семантику деякої мови). Дане відношення полягає в тому, що мовні вирази (слова, словосполучення, пропозиції, тексти) означають те, що є в світі, – предмети, якості (або властивості), дії, способи вчинення дій, відносини, ситуації і їх послідовності.

Вершини мережі представляють деякі сутності (об'єкти[4], події[5], процеси[6], явища), а зв’язуючи їх дуги – відносини між цими сутностями. З цієї причини мову семантичних мереж іноді називають реляційною мовою відносин. Наклавши обмеження на опис вершин і дуг, можна отримати мережі різного виду.

Якщо вершини не мають власної внутрішньої структури, то відповідні мережі називають ***простими мережами***. Якщо вершини мають деяку власну структуру, то такі мережі називаються ***ієрархічними мережами***. На початковому етапі розробки систем ШІ використовувалися тільки прості мережі. В даний час, в більшості додатків, що використовують семантичні мережі, вони є ієрархічними.

Відносини в мережах можуть бути найрізноманітнішого типу, що дозволяє в достатній мірі забезпечити в семантичній мережі таку ознаку знань, як зв'язність.

У загальному випадку, це означає, що за допомогою семантичної мережі можна відобразити знання, представлені в текстах природною мовою.

Розглянемо, наприклад, наступну фразу [1]: «Рибак (***a1***) сів у човен (***a2***), переплив на протилежний берег (***a3***) і взяв кошик (***a4***) з рибою (***a5***). Ці об'єкти зв'язані наступними відносинами: «сів у» (***r1***), «переплив» (***r2***), «взяв» (***r3***) і «знаходиться» (***r4***). Мережа, що відповідає цьому тексту, показана на рис. 2.7.

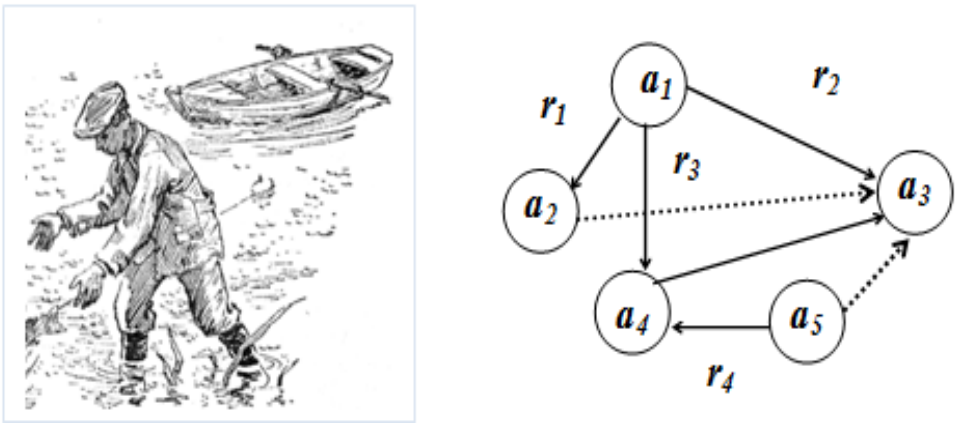

Рис.. 2.7. Уявлення об’єктів (*рибак, човен, берег, корзина, риба*) і подій (*сів, переплив, взяв*, з*находиться*) у виді семантичної мережі (приклад 1)

[4] Об'єкт – предмет або явище, що існують в реальній дійсності. Те, що має чітко окреслені межі. Відчутна сутність, яка має чітко визначальну поведінку.

[5] Подія – те, що має місце, відбувається, настає у довільній точці простору–часу; значна подія, явище чи інша діяльність, як факт суспільного або особистого життя; підмножина результатів експерименту.

[6] Стандарт «ISO 9000: 2000 Системи менеджменту якості» визначає процес як сукупність взаємозв'язаних і взаємодіючих дій, що перетворюють вхідні дані в вихідні.

Процес (інформатика) – програма у комп'ютерній системі, що виконується.

Процес (теорія організації) – стійка і цілеспрямована сукупність взаємопов'язаних дій, які за певною технологією перетворять входи у виходи для отримання заздалегідь визначених продуктів, результатів або послуг, які мають цінність для споживача.

Стрілки на рисунках вказують на значення, які повертаються після звернення.

Слід також зазначити, що наявність у конструкцій, які формуються імен фреймів і імен слотів означає, що значення, які зберігаються у фреймах, мають характер відсилань і тим самим внутрішньо інтерпретовані. Можливість розміщення в якості слотів наказів виклику тих чи інших процедур для виконання (так званих ***демонів)*** дозволяє активізувати програми на основі наявних знань.

Таким чином, фреймові мови задовольняють чотирма основними ознаками знань – інтерпретованості, структурованості, зв'язності і активності. Використання фреймів у фундаментальних науках дає можливість формування більш суворого понятійного апарату і комплексування звичайних моделей з фреймовими формалізмами. Для описових наук фрейми – це один з небагатьох способів дієвої формалізації для створення понятійного апарату.

Виходячи з логіки подій реального світу і прийнятого способу опису всіх можливих ситуацій, можна вважати даними і деякі інші відносини, явно не присутні в початковому тексті. Ці додаткові відносини показані на рис. 2.7 пунктиром. Розширений варіант тексту буде таким: «Рибак сів в човен і на човні переплив на інший берег. На іншому березі знаходилася риба. Риба знаходилася в кошику. Рибак взяв кошик з рибою».

Слід підкреслити одну важливу обставину. Як показали дослідження, проведені на прикладі мов індоєвропейської групи, є не більше 200-т різних відносин, що не зводяться один до одного. Комбінації цих базових відносин дозволяють виразити всі інші відносини, що фіксуються в текстах природною мовою.

Ця обставина лежить в основі формування моделей так званого ***ситуаційного управління***. Крім того, наявність кінцевої множини базових відносин дозволяє представляти в базах знань будь-яку предметну область, і, більш того, здійснювати автоматичну побудову семантичних мереж на основі будь-якого тексту.

Наприклад, знання про людину з прізвищем Іванов, що володіє автомобілем марки Рено червоного кольору, можуть бути відображені семантичною мережею, представленою на рис. 2.8.

Мережа містить також вершини і дуги, не згадані в попередньому реченні і відображають такі відносини, як «частина–ціле», зв'язки типу «це», «належить», «любить» і т.д.

Основна відмінність ієрархічних семантичних мереж від простих мереж полягає в можливості розділити мережу на підмережі-простори і встановлювати співвідношення не тільки між вершинами, а й між просторами. Всі вершини і дуги є елементами, принаймні, одного простору. Відзначимо, що поняття простору аналогічно поняттю дужок в математичній нотації.

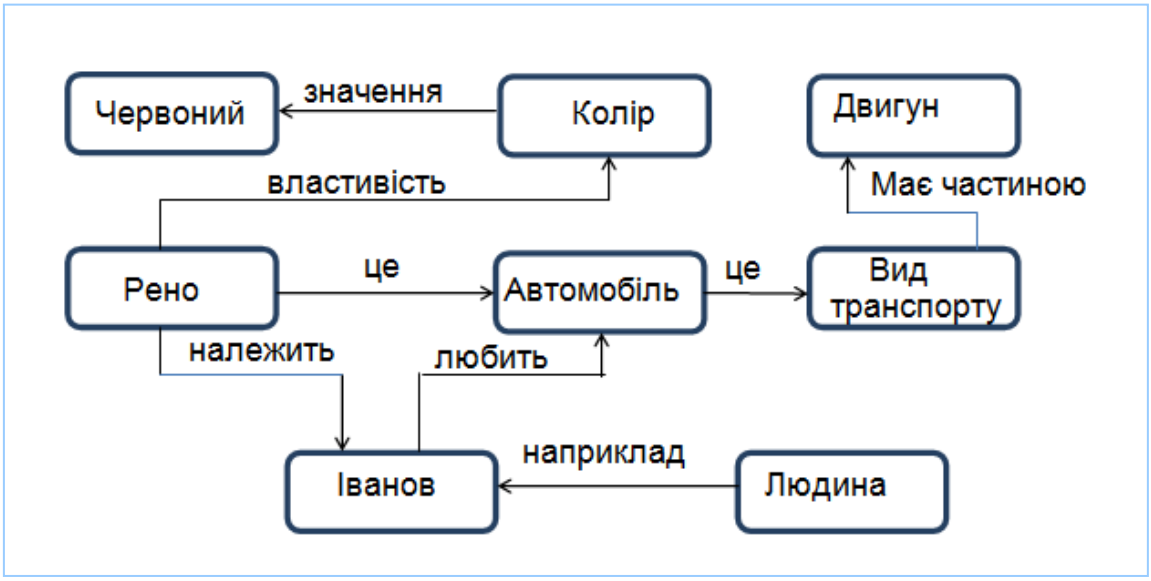

Рис.. 2.8. Семантична мережа (приклад 2)

Різні простори, існуючі в мережі, можуть бути впорядковані у вигляді дерева просторів, вершинам якого відповідають простори, а дугам – відносини «видимості».

На рис. 2.9 приведений приклад дерева просторів, відповідно до якого, наприклад, з простору-нащадка P_6 видимі всі вершини і дуги, що лежать в просторах-предках P_4, P_2 і P_0, а інші простори «невидимі».

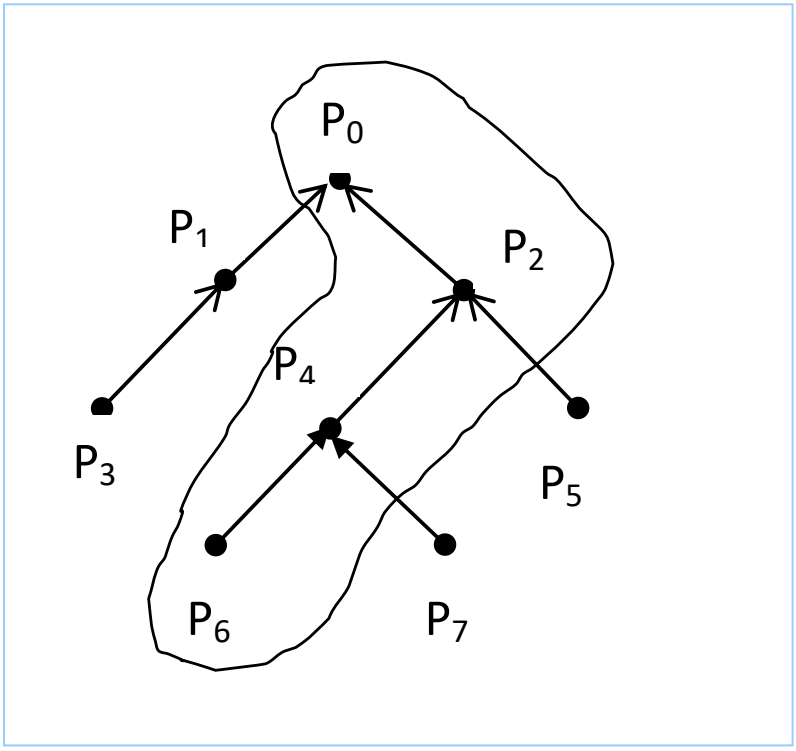

Рис.. 2.9. Простір станів.

Відношення «видимості» дозволяє групувати простори у впорядковані множини – ***перспективи***. Перспектива зазвичай використовується для обмеження мережевих сутностей, «видимих» деякою процедурою, яка працює з мережею.

Окремим випадком семантичних мереж є ***сценарії*** або ***однорідні семантичні мережі*** [8,11].

У таких мережах об'єкти пов'язані єдиним (*рос*. – единственным) відношенням суворого або несуворого порядку з різною семантикою.

Якщо, наприклад, об'єктами–поняттями будуть роботи (або окремі операції), а єдиним відношенням суворого порядку буде відношення слідування, то ми прийдемо до добре відомого мережевого графіку комплексу робіт з так званим «французьким представленням». Зрозуміло, що сценарії є зручним засобом складання планів.

Предметна область є множиною допустимих станів своїх компонентів. Представлене через загальні поняття і відносини між ними, ця множина утворює базу знань (БЗ) – у вигляді так званої інтенсіональної семантичної мережі. З іншого боку, в залежності від ситуації компоненти предметної області матимуть конкретні значення, властивості і характеристики. Всі ці конкретні дані про предметну область будуть відображатися в так званій екстенсіональній семантичній мережі, або базі даних (БД) мережевої структури.

Терміни ***«інтенсіональний»*** та ***«екстенсіональний»*** запозичені з семантики – науки про знакові системи.

Інтенсіонал – визначення або опис деякого поняття через його властивості. ***Екстенсіонал*** – набір конкретних фактів, відповідних даному поняттю.

На прикладі семантичної мережі загального вигляду можна встановити різницю між базою даних і базою знань (див. рис. 2.10).

Рис.. 2.10. Дані і знання в семантичній моделі.

У загальному випадку, ***інтенсіонал*** – це набір загальних понять і відносин, які характеризують безліч об'єктів, предметів, явищ.

Екстенсіоналом називають конкретні характеристики кожного елемента цієї множини понять і відносин. Наприклад, поняття «легкова машина» з родовидовими відносинами «кузов», «двигун» і «управління» буде інтенсіоналом по відношенню до множини екстенсіонала – марок легкових автомашин («вольво», «мерседес», «жигулі», «форд», «рено») з їх конкретними характеристиками. У свою чергу, якщо в якості загального поняття – інтенсіонала – виступають, наприклад, «жигулі», то екстенсіоналами можуть бути їх моделі, що випускаються (2101, 21301, 2303, 2309 і т.д.) з конкретними характеристиками. Таким чином, самі поняття інтенсіонала і екстенсіонала є відносними.

У комп'ютерах семантична мережа реалізується у вигляді унутрішньомашинної структури даних, яка використовується для представлення окремих слів і їх семантики. Наприклад [11], семантичну мережу «НОК – це найменше спільне кратне» (див. рис. 2.11) на основі результатів структурного аналізу можна переписати у вигляді структури даних семантичної мережі, як показано на рис. 2.12.

Тут С1, С2, ..., С9 – вузли семантичної мережі, які є покажчиками спискової структури.

Наведені скорочення для деяких граматичних термінів мають наступний сенс.

ТОК – друк відповідної семантики, причому деяке активне слово записується в початковій формі;

MODAL – час (оцінка) і відмінювання дієслова;

MOD – модифіковані слова;

POST – за допомогою союзу показує модифіковане слово;

AUX – допоміжний.

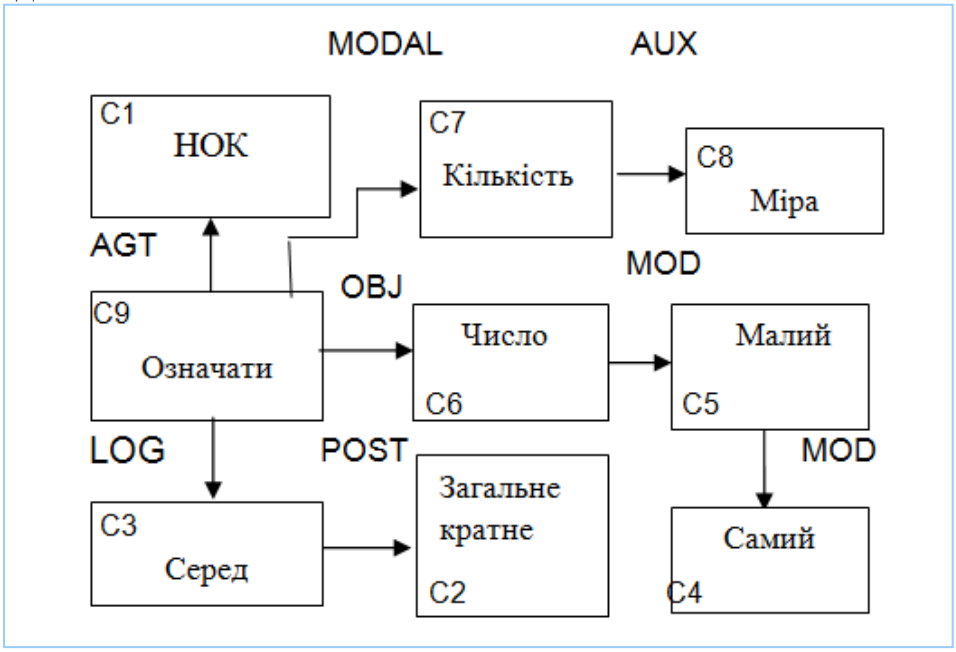

Рис. 2.11. Семантична мережа «НОК – це найменше спільне кратне»

С9	ТОК	ОЗНАЧАТИ	С4	ТОК	САМИЙ
	AGT	С1		MOD	НУЛЬ
	LOG	С3	С5	ТОК	МАЛИЙ
	OBJ	С6		MOD	С4
	MODAL	С7	С6	ТОК	ЧИСЛО
С1	ТОК	НОК		MOD	С5
	MOD	НУЛЬ	С7	PRES	КІЛЬКІСТЬ
С2	ТОК	ЗАГАЛЬНЕ КРАТНЕ		AUX	С8
	ТОК	НУЛЬ	С8	ТОК	МІРА
С3	ТОК	СЕРЕД			
	POST	С2			
	MOD	НУЛЬ			

Рис. 2.12. Структура даних семантичної мережі

Позитивна сторона представлення знань семантичними мережами полягає в тому, що це дуже простий і зрозумілий спосіб опису на підставі відносин між елементами (вузлами і дугами). Однак, зі збільшенням розмірів мережі, істотно збільшується час пошуку рішень в порівнянні зі способами, які не мають стратегії. Крім того, їм властива проблема гарантії придатності результатів виведення, що включає також проблему успадкування властивостей.

2.2. Фреймові моделі

Семантичні мережі, незважаючи на їх великі можливості, пов'язані з багатством наявних засобів, для відображення відносин поміж поняттями і об'єктами, разом з тим, мають і деякі недоліки. Реалізація моделей з довільною структурою і різними типами вершин вимагає великої різноманітності процедур обробки інформації, що ускладнює програмне забезпечення комп'ютерів. Це зумовило появу ряду специфічних типів семантичних мереж, таких як: синтагматичні ланцюги, сценарії, фрейми і т.д. Розглянемо докладніше фреймові уявлення.

Фрейм (*англ*. Frame – «каркас» або «рамка») – спосіб представлення знань у галузі штучного інтелекту, який представляє собою схему дій в реальній ситуації. Спочатку термін «фрейм» ввів Марвін Мінський в 70–і роки XX століття [15] для позначення структури знань при моделюванні сприйняття просторових сцен. Фрейм – це модель абстрактного образу, мінімальний можливий опис сутності будь-якого об'єкта, явища, події, ситуації або процесу. Розрізняють фрейми-зразки, фрейми-екземпляри, фрейми-структури, фрейми-ролі, фрейми-сценарії, фрейми-ситуації. При цьому, система зв'язаних фреймів може утворювати ***семантичну мережу***.

Під структурою фрейма розуміється спосіб використання схеми, типової послідовності дій або ситуативної модифікації фрейму. Фрейм, крім усього іншого, включає певне знання за замовчуванням, яке називається ***презумпцією***.

Зазвичай він має однорідну структуру, складається з імені та окремих одиниць, які називаються ***слотами*** (рис. 2.13)

ІМ'Я ФРЕЙМА
Ім'я 1-го слоту: значення 1-го слоту
Ім'я 2-го слоту: значення 2-го слоту
……
Ім'я *N*-го слоту: значення *N*-го слоту

Рис. 2.13. Структура фрейму

Слоти (*англ.* Slot – проріз, щіль) – це деякі незаповнені підструктури фрейму, після заповнення яких конкретними даними, фрейм буде представляти ту чи іншу ситуацію, явище чи об'єкт предметної області. При конкретизації фрейму йому і його слотам присвоюються конкретні імена і виконується заповнення слотів. Як значення слотів можуть виступати імена інших фреймів, що дає можливість будувати мережі фреймів.

Формально під фреймом розуміють структурний запис наступного виду:

$$[\ <\ f\ >, <\ V_1, g_1\ >, <\ V_2, g_2\ >, , <\ V_3, g_3\ >, \ldots\ldots, <\ V_n, g_n\ >\ ] \qquad (2.1)$$

Тут: f – ім'я фрейму, пара $<V_i\ , G_i>$ – це і – ий слот, де V_i – ім'я слота, а g_i – його значення. Значенням слота може бути практично все що завгодно (числа або математичні співвідношення, тексти на природній мові, програми, правила виведення або посилання на інші слоти даного фрейму або інших фреймів і т.д.).

Фрейми часто ділять на дві групи: ***фрейми–описи*** та ***рольові фрейми***. Розглянемо ряд прикладів.

Наприклад, фрейм-опис знань, щодо прибувших на базу вантажів кількох видів фруктів може бути записаний таким чином:

[< Фрукти >, < виноград, Болгарський 20 т >, < яблука, Джонатан **10 т >, < вишня, Володимирська 200 кг >].**

Прикладом рольового фрейму може служити набір знань про вантажі, що перевозяться:

[<Перевезти>, <що, прокат 300 т>, <звідки, Кривий Ріг>,< куди, Одесу >,< чим, залізничним транспортом >,< коли, в грудні 2017 року>].

У рольовому фреймі іменами слотів виступають питальні слова, відповіді на які є значеннями слотів.

Якщо в прикладах в загальному виразі для фрейма прибрати всі значення слотів, залишивши лише імена, то отримаємо конструкцію, звану ***протофреймом*** (прототипом фрейму).

Фрейми з конкретними значеннями називаються фреймами–екземплярами. Розглянемо приклади цих видів фреймів.

Приклад ***протофрейму***:

[<Список співробітників >,< прізвище (значення слоту 1) >, <рік народження (значення слоту 2) >, <спеціальність (значення слоту 3) >, < стаж (значення слоту 4) >].

Приклад ***фрейму – примірника***:

[<Список співробітників >,< прізвище (Попов–Сидоров–Іванов–Петров) >, <рік народження (1965–1968–1987–1958) >,< спеціальність (слюсар–токар–монтажник–сантехнік) >,< стаж (5 –21–32–23) >].

Фрейми мають властивість вкладеності. У цих випадках в якості значення слоту може виступати система імен слотів більш глибокого рівня. Властивість вкладеності, можливість мати в якості значень слотів посилання на інші фрейми і на інші слоти того ж самого фрейму забезпечують фреймовим мовам відповідність вимогам структурованості і зв'язності знань [11].

Слот фрейму може містити не тільки конкретне значення, а й ім'я процедури, що дозволяє обчислити його за заданим алгоритмом, а також одну або кілька продукцій (евристик), за допомогою яких це значення визначається. У слот може входити не одне, а кілька значень. Іноді цей слот включає компонент, званий ***фасетом***, який задає діапазон або перелік його можливих значень. Фасет вказує також граничні значення заповнювача слоту.

Крім конкретного значення в слоті можуть зберігатися процедури і правила, які викликаються при необхідності обчислення цього значення. Серед них виділяють ***процедури–демони*** і ***процедури–слуги***. Перші запускаються автоматично при виконанні деякої умови, а другі активізуються тільки за спеціальним запитом. Якщо, наприклад, фрейм, що описує людину, включає слоти <ДАТА НАРОДЖЕННЯ "і" ВІК>, і в першому з них знаходиться деяке значення, а в другому слоті може стояти ім'я процедури–демона, що обчислює вік за датою народження і поточною датою і активізується при кожній зміні поточної дати.

Таким чином, з урахуванням можливості успадкування структура даних фрейму може виглядати так (див. рис. 2.14).

Ім'я фрейму – це ідентифікатор, що присвоюється фрейму. Фрейм повинен мати унікальне ім'я в системі. Фрейм складається зі слотів.

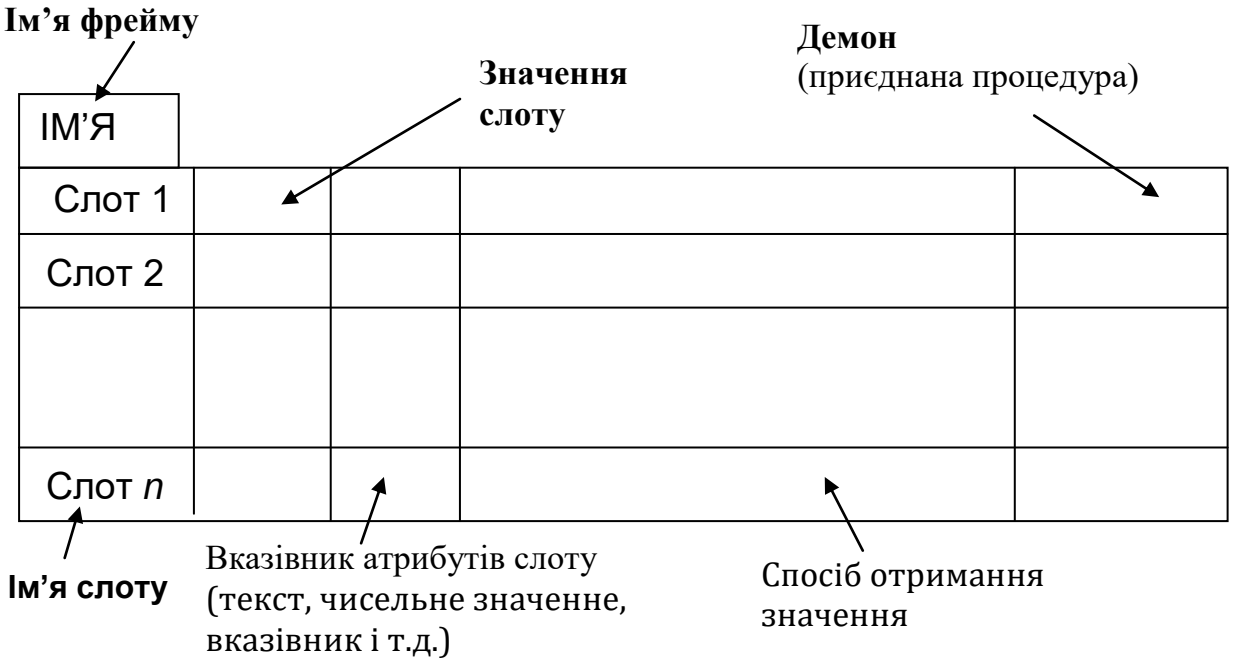

Рис. 2.14. Структура розміщення даних у фреймі

Слотів у фреймі може бути довільне число. Деякі з них визначаються самою системою для виконання специфічних функцій, а інші визначаються користувачем. В їх число входять слот «IS-A», що показує фрейм-батько даного фрейма, слот покажчиків дочірніх фреймів, який є списком покажчиків цих фреймів, слот для введення імені користувача, дати зміни, тексту коментаря та інші слоти. Кожен слот, в свою чергу, також представлений певною структурою даних.

Ім'я слоту – це ідентифікатор, що присвоюється слоту. При цьому, слот повинен мати унікальне ім'я у фреймі. Деякі слоти називаються системними і використовуються при редагуванні бази знань і управлінні виводом.

Покажчики наслідування стосуються тільки фреймових систем ієрархічного типу, основаних на відношеннях «абстрактне – конкретне». Вони показують, яку інформацію про атрибути слотів у фреймі верхнього рівня наслідують слоти з такими ж іменами у фреймах нижнього рівня. Типові покажчики наслідування приведені на рис. 2.15.

Покажчик з ім'ям ***O*** виконує одночасно функції покажчиків з іменами ***U*** і ***S***. Незважаючи на те, що в більшості систем допускається кілька варіантів вказівок на успадкування, існує чимало і таких, де допускається тільки один варіант. В такому випадку можна вважати, що використовується покажчик ***O*** значення за замовчуванням. Приклад використання покажчиків наведено на рис. 2.16.

- ***U*** (перша буква слова ***Unique***–унікальний) – кожний фрейм може мати різноманітні слоти з різними значеннями;
- ***S*** – всі слоти повинні мати однакові значення;
- ***R*** – значення слотів фрейму нижнього рівня повинні знаходитись в межах, вказаних значеннями слотів верхнього рівня;
- ***O*** – при відсутності вказівки значення слоту фрейму верхнього рівня стає значенням слоту фрейму нижнього рівня, але у випадку визначення нового значення слотів фреймів нижніх рівнів, вказуються у якості значень слотів.

Рис. 2.15. Основні покажчики успадкування

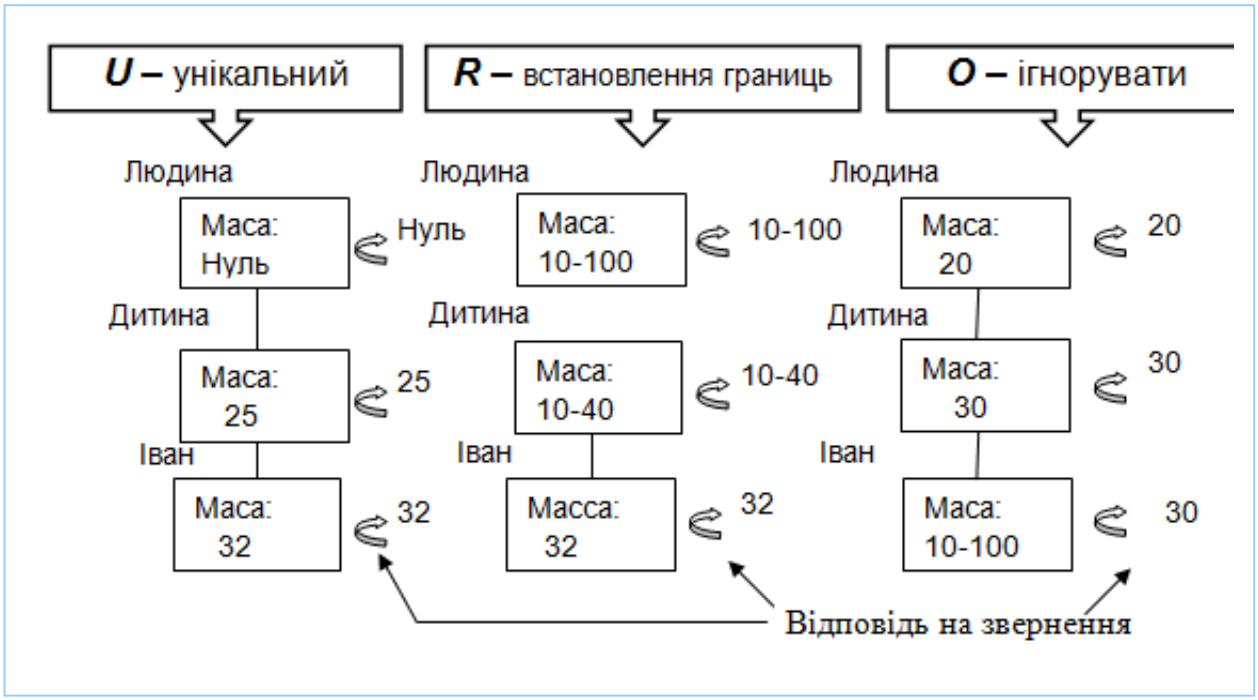

Рис. 2.16. Вказівки наслідування (*U*, *R*, *O*) і відповіді на звернення до значення слоту.

Стрілки на рисунках вказують на повернені після звернення значення.

Слід також зазначити, що наявність у конструкціях, що формуються імен фреймів і імен слотів означає, що значення, які зберігаються у фреймах, мають характер відсилань і тим самим внутрішньо інтерпретовані. Можливість розміщення у якості слотів наказів виклику тих чи інших процедур для виконання (так званих ***демонів***) дозволяє активізувати програми на основі існуючих знань.

Таким чином, фреймові мови задовольняють чотирьом основним ознакам знань – інтерпретованості, структурованості, зв'язності і активності. Використання фреймів у фундаментальних науках дає можливість

формування більш суворого понятійного апарату і комплексування звичайних моделей з фреймовими формалізмами. Для описових наук фрейми – це один з небагатьох способів діючої формалізації для створення понятійного апарату.

2.3. Логічні моделі знань.

Логічні моделі знань є основою людських міркувань і висновків, які, в свою чергу, можуть бути описані відповідними логічними обчисленнями[7]. До таких обчислень в першу чергу слід віднести сіллогістіку Аристотеля, а також прикладні обчислення висловлювань і предикатів, аксіоматика яких і використовується в якості логічних моделей знань.

Після більш ніж 2000-літнього незмінного стану сіллогістіка Аристотеля отримала розвиток і важливе практичне застосування саме в роботах з штучного інтелекту.

Логічні обчислення можуть бути представлені як формальні системи у вигляді четвірки:

$$\mathbf{M = < T , P , A , B >} \qquad (2.2)$$

Де: **T** – *множина базових елементів* різної природи, наприклад, слова з деякого обмеженого словника, літери деякого алфавіту, деталі дитячого конструктора, що входять до складу деякого набору і т.д. Важливо, що для множини **T** існує деякий спосіб визначення належності або неналежність довільного елемента ***x*** цієї множини. Процедура такої перевірки може бути будь-якою, але за кінцеве число кроків вона повинна давати позитивну або негативну відповідь на питання, чи є ***x*** елементом даної множини **T**. Позначимо цю процедуру ***П*** **(T)**.

Множина **P** є *множиною синтаксичних правил,* на основі яких будуються правильно побудовані формули. Наприклад, зі слів обмеженого словника будуються синтаксично правильні фрази $\mathbf{(X_1, X_2, ..., X_N)}$ або з деталей дитячого конструктора за допомогою гайок і болтів збираються нові конструкції $\mathbf{(X_1, X_2, ..., X_N)}$. Декларується існування процедури ***П*** **(P),** за допомогою якої за кінцеве число кроків можна отримати відповідь на питання, чи є яка-небудь з них, наприклад, сукупність $\mathbf{X_k}$, синтаксично правильною.

A – це *множина правильно побудованих формул* (ППФ), елементи якого називаються ***аксіомами.*** Як і для інших складових формальної системи, повинна існувати процедура ***П*** **(A),** за допомогою якої для будь-якої синтаксично правильної сукупності можна отримати відповідь на питання про приналежність її множині **A.**

[7]Логічні обчислення - теорія формальних логічних обчислень. Ця теорія інакше називається математичною або формальною логікою. Історично логічні обчислення були розроблені для теоретичної формалізації процесу доказів в різних теоріях.

Множина **В** є *множиною правил виведення,* які з множини **А** дозволяють отримувати нові правильно побудовані формули – ***теореми.*** До останніх знову можна застосовувати правила з **В.** Так формується множина виведених в даній формальній системі сукупностей. Якщо є деяка процедура ***П*** **(В),** за допомогою якої можна визначити для будь-якої синтаксично правильної сукупності, чи є вона виводимою, то відповідна формальна система називається ***вирішуваною.*** Це показує, що саме правила виведення є найбільш складною складовою логічної моделі.

Для знань, що входять в базу знань з використанням логічної моделі, можна вважати, що множина **А** утворює всі інформаційні одиниці, які введені в базу знань ззовні, а за допомогою правил виведення з них виводяться нові ***похідні*** знання. Іншими словами, формальна система являє собою ***генератор породження нових знань,*** які утворюють множину виведених в даній системі знань. Ця властивість логічних моделей робить їх привабливими для використання в базах знань. Вона дозволяє зберігати в базі лише ті знання, які утворюють множину аксіом **А,** а всі інші знання отримувати з них за правилами виведення.

Прикладами формальної системи є *числення висловів* і *числення предикатів,* які розглянуті в третьому і четвертому розділах.

2.4. Продукційні моделі

Продукційна модель знання – це модель, заснована на правилах, яка дозволяє представити знання у вигляді речень типу «***Якщо*** (умова), ***то*** (дія)». Є фрагментом семантичної мережі і заснована на тимчасових відносинах між станами об'єктів.

Продукційна модель має той недолік, що при накопиченні досить великого числа (порядку декількох сотень) продукцій вони починають внаслідок незворотності диз'юнкцій суперечити один одному. В цьому випадку розробники починають ускладнювати систему, включаючи до неї модулі нечіткого виведення чи інші засоби вирішення конфліктів: а) правила за пріоритетом; б) правила за глибиною; в) евристичні механізми винятків; г) евристичні механізми повернення і т.д.

У моделях даного типу використовуються деякі елементи описаних вище логічних і мережевих моделей. З логічних моделей запозичена ідея правил виведення, які тут називаються ***продукціями,*** а з мережевих моделей – опис знань у вигляді ***семантичної мережі.***

В результаті застосування правил виведення до фрагментів мережевого опису відбувається трансформація семантичної мережі за рахунок зміни її фрагментів, нарощування мережі і виключення з неї непотрібних фрагментів. Таким чином, в продукційних моделях процедурна інформація явно виділена і описується іншими засобами, ніж декларативна інформація. Замість логічного виводу, характерного для логічних моделей, в продукційних моделях з'являється ***вивід на знаннях.***

Продукції, з одного боку, близькі до логічним моделям, що дозволяє організувати на них ефективні процедури виведення, а з іншого боку, дозволяють більш наочно відображати знання, ніж класичні логічні моделі. У них відсутні жорсткі обмеження, характерні для логічних обчислень, що дає можливість змінювати інтерпретацію елементів продукції.

У загальному вигляді під ***продукцією*** розуміється вираз наступного вигляду:

(i) ; Q ; P ; A ⇨ B ; N (2.3).

Тут **i** – ім'я продукції, за допомогою якої дана продукція виділяється зі всієї множини продукцій.

Як ім'я може виступати деяка ***лексема,*** що відображає суть даної продукції (наприклад, «придбання книги», «набір коду замку»), або порядковий номер продукції в їх множині, що зберігається в пам'яті системи.

Елемент **Q** характеризує сферу застосування продукції. Такі сфери легко виділяються в ***когнітивних***[8] структурах людини. Наші знання як би «розкладені по поличках». На одній «поличці» зберігаються знання про те, як треба готувати їжу, на другий – як дістатися до роботи і т.д. Поділ знань на окремі сфери дозволяє економити час на пошук потрібних знань. Створення такого ж поділу на сфери в базі знань інтелектуальної системи доцільно і при використанні для представлення знань продукційних моделей.

Основним елементом продукції є її ядро: **A ⇨ У.** Інтерпретація ядра продукції може бути різною і залежить від того, що стоїть ліворуч і праворуч від знака ***секвенції*** **⇨.** Звичайне прочитання ядра продукції виглядає так: ЯКЩО **A,** ТО **B.** Більш складні конструкції ядра допускають в правій частині альтернативний вибір, наприклад, ЯКЩО **A,** ТО $\mathbf{B_1}$, ІНАКШЕ $\mathbf{B_2}$.

Секвенція може тлумачитися в звичайному логічному сенсі як знак логічного слідування елемента **B** з істинного **A** (якщо **A** не є істинним виразом, то про **B** нічого сказати не можна). Можливі й інші інтерпретації ядра продукції, наприклад, **A** описує деяку *умову,* необхідну для того, щоб можна було вчинити *дію* **B.**

Елемент **P** є *умовою застосовності ядра продукції.* Зазвичай **P** являє собою логічний вираз (як правило, предикат). Коли **P** приймає значення «істина», ядро продукції активізується. Якщо **P** помилково (*рос*. – ложно), то ядро продукції не може бути використано. Наприклад, якщо в продукції: «наявність грошей; якщо хочеш купити річ **X,** то заплати в касу її вартість і віддай чек продавцю », умова застосовності ядра помилкова, тобто якщо грошей немає, то застосувати ядро продукції неможливо.

Елемент **N** описує ***постумови продукції.*** Вони актуалізуються тільки в тому випадку, якщо ядро продукції реалізувалося.

Післяумови продукції описують дії і процедури, які необхідно виконати після реалізації ядра.

[8]Когнітивний - (лат. Cognitio - сприйняття, пізнання) відноситься до пізнання, до функцій мозку, які забезпечують формування понять, оперування ними і отримання вивідних знань. У психології термін «каганець» означає здатність до набуття знання і його переробці.

Наприклад, після покупки деякої речі в магазині необхідно в описанні товарів, наявних в цьому магазині, зменшити кількість речей такого типу на одиницю. Виконання постумови **N** може відбуватися не відразу після реалізації ядра продукції.

Якщо в пам'яті системи зберігається певний набір продукцій, то вони утворюють ***систему продукцій.*** В системі продукцій повинні бути задані спеціальні процедури управління продукціям, за допомогою яких відбувається актуалізація продукцій і вибір для виконання тієї чи іншої продукції.

Базова структура виробничої системи складається з трьох основних компонентів [11]. Перший з них – це набір правил, використовуваний в якості основи знань, тому його ще називають ***базою правил*** (рис.2.17). Другим компонентом є ***робоча пам'ять*** (або пам'ять для короткочасного зберігання), в якій зберігаються передумови, що стосуються конкретних завдань предметної області, і результати висновків, отримані на їх підставі. Третій компонент – це ***механізм логічного висновку,*** що використовує правила відповідно до вмісту робочої пам'яті

Для того щоб показати, як взаємодіють ці елементи, розглянемо нескладний приклад [8]. Дані, що записуються в робочу пам'ять, являють собою зразки (*рос*. – образцы) у вигляді набору символів, наприклад, «намір – відпочинок», «місце відпочинку – гори» і т.д. Правила, що записуються в базу правил, відображають вміст робочої пам'яті.

В умовній частині правил знаходяться або поодинокі зразки, або кілька умов, з'єднаних прийменником «І», а в заключній частині – зразки, додатково реєструються в робочій пам'яті.

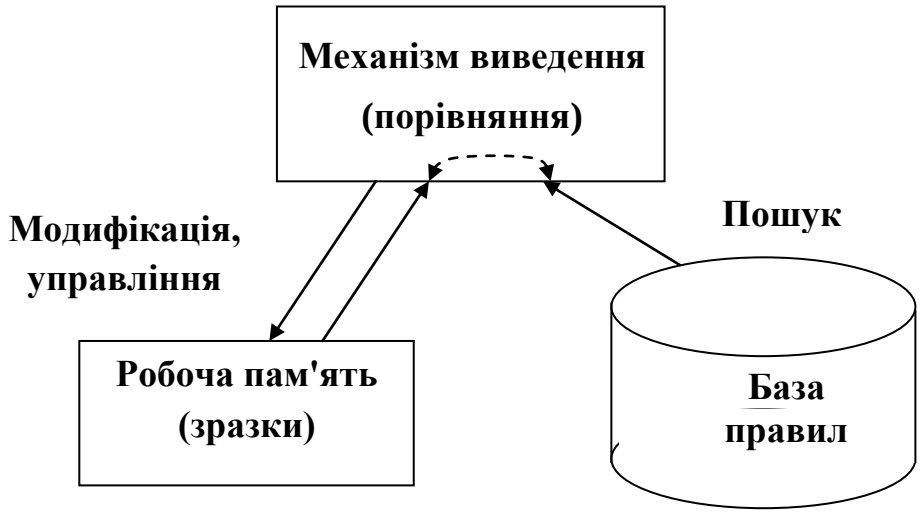

Рис. 2.17. Структура продукційної системи

Розглянемо конкретний випадок застосування продукційної системи. Нехай, наприклад, правила такі:

Правило 1. ЯКЩО «намір – відпочинок» І
«дорога вибоїста» ТО «використовувати джип».

Правило 2. ЯКЩО «місце відпочинку – гори» ТО «дорога вибоїста».

Після того як в робочу пам'ять записуються зразки «намір – відпочинок» і «місце відпочинку – гори», розглядається можливість застосування цих правил. Відомі два підходи для отримання логічного виводу в системі – ***прямий вивід*** і ***зворотний вивід.*** Розглянемо суть цих підходів.

При ***прямому виведенні*** відбувається робота з витягання попередньо записаних даних робочої пам'яті, застосування правил і доповнення даних, які завантажуються в робочу пам'ять. Спочатку механізм виведення зіставляє зразки в умовній частині зі зразками, що зберігаються в робочій пам'яті. Якщо все зразки є в робочій пам'яті, то умовна частина вважається істинною, в іншому випадку – неправдивою (*рос*. – ложной). В даному прикладі зразок «намір – відпочинок» існує в робочій пам'яті, а зразок «дорога вибоїста» відсутній, тому умовна частина правила 1 вважається помилковою. Умовна частина правила 2 є істинною. Оскільки в даному випадку існує тільки одне правило з істинною умовною частиною, то механізм виведення відразу ж виконує його заключну частину і зразок «дорога вибоїста» заноситься в робочу пам'ять. При спробі повторно застосувати ці правила виходить, що можна застосувати лише правило 1, оскільки правило 2 вже було застосовано і вибуло з числа кандидатів. До цього часу вміст робочої пам'яті було доповнено новим зразком – результатом застосування правила 2, тому умовна частина правила 1 стає справжньою, і вміст робочої пам'яті поповнюється зразком його заключній частині – «використовувати джип». У підсумку, правил, які можна було б застосовувати, не залишається, і система зупиняється.

Зворотний вивід – це спосіб виведення знань, при якому на підставі фактів, які потребують підтвердження, щоб виступати в ролі закінчення (*рос*. – заключения), досліджується можливість застосування правила, придатного для підтвердження. Припустимо, що кінцева мета – це «використовувати джип», і досліджується спочатку можливість застосування правила 1, що підтверджує цей факт. Оскільки зразок «намір – відпочинок» з умовною частини правила 1 вже занесений в робочу пам'ять, то для досягнення мети досить підтвердити факт «дорога вибоїста». Однак якщо взяти зразок «дорога вибоїста» за нову мету, то буде потрібно правило, яке підтверджує цей факт. Тому досліджується можливість застосування правила 2. Умовна частина цього правила в даний момент є істинною, тому правило 2 можна відразу ж поміняти, робоча пам'ять при цьому поповниться зразком «дорога вибоїста», і в результаті можливості застосування правила 1 підтверджується мета – «використовувати джип» .

У разі зворотного виведення умови зупинки системи очевидні: або досягається первинна мета, або закінчуються правила, які можуть бути застосовані для досягнення мети в ході виведення. Що стосується прямого виведення, то, як було зазначено вище, відсутності відповідних правил також є умовою зупинки. Однак система зупиняється і при виконанні деякої умови, якій задовольняє вміст робочої пам'яті, наприклад, шляхом перевірки появи

зразка «використовувати джип». Доказано, що для зворотних виводів характерна тенденція виключення з розгляду правил, які не мають прямого відношення до заданої мети, що дозволяє підвищити ефективність виведення.

Продукційні системи прості в застосуванні і завдяки гарній підготовленості засобів розробки дозволили створити велике число систем, заснованих на цій моделі знань. Продукційні системи, на відміну від фреймових і інших систем, не мають таких функцій, як, наприклад, встановлення високорівневих відносин між фреймами, але, з іншого боку, завдяки цьому спрощується проектування системи і її створення. В майбутньому, ймовірно, знайдуть застосування не тільки чисто продукційні системи, але і комбінації їх, наприклад, з фреймової системами.

3. Основи логіки висловлювань.

3.1. Введення в логіку висловлювань

Висловлення – це речення, що виражає судження. Якщо судження (результат міркування), що складає зміст (сенс) деякого ***висловлювання,*** істинно, то і про це ***висловлювання*** кажуть, що воно істинне.

Міркування – це ланцюжок взаємопов'язаних умовиводів. Залежно від характеру взаємозв'язку міркування буває ***індуктивним*** або ***дедуктивним.*** Якщо взаємозв'язок умовиводів побудований на ***індукції,*** то міркування індуктивне. Розглянемо ці поняття.

Індукція – це спосіб міркування, який починається від огляду окремих фактів і положень до загальних висновків, тобто це умовивід, що виконується на основі конкретних фактів і приводить до деякої гіпотези, тобто до ***утвердження*** узагальнюючих абстракцій, типових для всіх разом і кожного окремо фактів і передбачає одночасне обговорення критеріїв для виправдання достовірності отриманої гіпотези.

Для розуміння сенсу застосування індукції розглянемо приклад [13].

Побудуємо графіки двох рівнянь з двома змінними, наприклад:

$$x - 2y + 4 = 0$$
$$2x + y - 5 = 0.$$

Переконавшись в *тому,* що графіки цих рівнянь в декартовій системі координат є прямі лінії, ми на підставі індукції можемо зробити висновок, що графіками всякого рівняння виду $ax + by + c = 0$ в тій же системі координат буде пряма. Цей висновок вірний. Але щоб остаточно переконатися в його достовірності, необхідно або перебрати всі можливі координати x і y в поєднанні з усіма можливими значеннями коефіцієнтів, що нереально, або придумати якийсь більш дотепний і ефективний метод докази. Пошук ефективних методів докази достовірності міркування є обов'язковим розділом в структурі теорії будь–якої математичної логіки.

При дедуктивному міркуванні, яке інакше називається ***обчисленням,*** по–перше, взаємозв'язок умовиводів заснований на дедукції, по–друге, ланки ланцюга умовиводів додатково пов'язані ставленням логічного слідування.

Дедукція (висновок) – це спосіб міркування, при якому нове положення виводиться суто логічним шляхом від загальних положень до приватних висновків. Початком (посилками) дедукції є деяка сукупність узагальнюючих абстракцій[9], а кінцем (заключенням) – мета міркування, яке представляється як питання–факт або питання–абстракція мінімального масштабу. Ось кілька прикладів.

Приклад 1. Всі люди смертні. Сократ – людина. Отже, Сократ смертний (висновок).

Приклад 2. Будь–яке натуральне число, сума цифр якого ділиться на три, саме ділиться на три. Сума цифр числа 4635 ділиться на три. Отже, число 4635 ділиться на три.

Як і в випадку індуктивного міркування, тут також для того, щоб переконатися в достовірності міркування, необхідно перевірити, тобто довести, що сума цифр числа 4635 ділиться на три. Це простий випадок доказу, однак, він необхідний в будь–якому дедуктивному міркуванні. У цьому прикладі не піддається сумнівам перша вихідна посилка, істинність якої доказана в теорії чисел.

Приклад 3. Графіком рівняння виду $ax + by + c = 0$ в прямокутній декартовій системі координат є пряма лінія. Графіком рівняння $by + c = 0$ є пряма лінія, паралельна осі x. Отже, графіком прямої лінії, паралельної осі y є рівняння $ax + c = 0$.

Сутність відносини логічного слідування полягає в тому, що істинність висновку слідує тільки з одночасної істинності всіх і тільки всіх посилок при всіх можливих варіантах заміни абстракцій фактами. Кількість посилок N може бути великою (у всякому разі, більше десяти), а кожна з посилок є узагальнюючої абстракцією, істинність якої можна прояснити тільки після заміни її конкретним фактом, можливу кількість (Р) яких перевищує число десять, а іноді досягає сотень тисяч. З'ясувати одночасну істинність всіх посилок при всіх можливих варіантах заміни посилок фактами стає практично неможливо. Доказано, що ця задача має обчислювальну трудомісткість NP - повного завдання[10].

---

[9] Абстракція – принцип ігнорування другорядних аспектів предмету з метою виділення головних. Абстракція (при абстрагуванні) виділяє істотні характеристики деякого об’єкта, що відрізняють його від всіх інших видів об’єктів і, таким чином, чітко визначає його концептуальні межі з погляду спостерігача.

[10] NP-повна задача – (в теорії алгоритмів) завдання з відповіддю «так» або «ні» з класу NP, до якої можна звести будь-яку іншу задачу з цього класу за поліноміальний час (тобто за допомогою операцій, число яких не перевищує деякого полінома в залежності від розміру вихідних даних). Таким чином, NP-повні задачі утворюють в деякому сенсі підмножину «типових» задач в класі NP: якщо для якоїсь з них знайдений «поліноміально швидкий» алгоритм рішення, то і будь-яка інша задача з класу NP може бути вирішена так само «швидко». Прикладами таких задач є задача комівояжера, проблема розкраски графа і т.д.

В теорії алгоритмів **класом NP** (від англ. *non-deterministic polynomial* – недетермінований многочлен) називають безліч завдань розпізнавання, рішення яких при наявності деяких додаткових відомостей (так званого сертифіката рішення) можна «швидко» (за час, що не перевершує полінома від розміру даних) перевірити на машині Тьюринга.

У зв'язку з цим дедуктивний міркування завжди доповнюється формальними так званими ***правилами виведення***, які дозволяють шляхом послідовного їх застосування до самої дедукції крок за кроком спрощувати її без порушення відносини логічного слідування, доводячи в кінцевому підсумку, до очевидною схеми, істинність якої встановити не складає труднощів. Ці правила називають інструментарієм аналізу достовірності міркувань.

В інтелектуальній діяльності людини розрізняють логічне та аналітичне мислення. ***Логічне мислення*** – це володіння індуктивним і дедуктивним міркуваннями. ***Аналітичне мислення*** – це вміння:

1) абстрагувати існуючі (і які спостерігаються в реальному світі) явища, процеси, об'єкти;

2) встановлювати причинно-наслідкові зв'язки між об'єктами, процесами, явищами, їх властивостями і характеристиками;

3) структурувати складні процеси з метою їх пізнання на основі системного підходу;

4) визначати критерії для зіставлення віддзеркалюваних об'єктів, процесів, явищ їх властивостей і характеристик;

5) зіставляти об'єкти, процеси, явища, їх властивості та характеристики на основі заданих критеріїв;

6) визначати і абстрагувати операційне середовище, в якому функціонують об'єкти, явища, процеси;

7) давати кількісну оцінку якості функціонування процесів, об'єктів і систем об'єктів;

8) математично моделювати віддзеркалювані процеси, явища, об'єкти і їх системи.

Кожна з функцій аналітичного мислення може бути легко реалізована шляхом логічного мислення, тобто логічне мислення може бути використано в якості моделі аналітичного мислення.

За визначенням штучний інтелект моделює логічне та аналітичне мислення. Логічне мислення, в свою чергу, моделює аналітичне мислення. Отже, логічне мислення являє собою універсальну модель штучного інтелекту.

3.2. Математична логіка і її зв'язок з логічним мисленням.

Математична логіка представляє собою функціонально-повний набір формальних інструментальних засобів, призначених для моделювання та реалізації індуктивних і дедуктивних міркувань [7,12]. Дані засоби включають такі елементи (див. рис. 3.1):

1) ***формальні мови логіки***, призначені для постановки інтелектуальних завдань та представлення (моделювання) знань в аксіоматичних теоріях будь-яких предметних областей;

2) ***математичні структури*** для представлення (моделювання) систем знань, еквівалентних перетворень систем знань, маніпулювання знаннями і формулювання нових знань;

3) ***формальні правила виводу*** і алгоритми як інструментарій для обґрунтування достовірності міркувань, тобто докази істинності висновків в міркуванні.

У основах класичної математики відомо декілька логічних систем, але в якості фундаменту математичної логіки обґрунтовано прийнята одна з них – *логіка першого порядку*, що має ще й інші назви – *логіка предикатів*, *числення предикатів*. Тому для оволодіння основами штучного інтелекту необхідно глибоке вивчення теоретичних основ логіки предикатів, які викладені в наступному розділі посібника.

Рис. 3.1. Складові частини математичної логіки

Основна перевага використання логіки предикатів для представлення знань полягає в тому, що володіє добре зрозумілими математичними властивостями потужний механізм виведення може бути безпосередньо запрограмований. Крім цього, треба знати, чим доповнюють інші логічні системи логіку предикатів і в яких ситуаціях слід вдаватися до їхньої допомоги. Слід також знати про графічних уявленнях логіки предикатів – семантичних мережах та мережах фреймів, зокрема про їх переваги та недоліки в порівнянні з логікою предикатів.

Існує шість аспектів, які однозначно і функціонально повно визначають обчислення як математичну структуру:

1) алфавіт обчислення;

2) правила освіти мови в алфавіті – синтаксис мови;

3) правила присвоєння істиннісних значень формулами – семантика мови;

4) правила виведення в обчисленні, тобто правила, що визначають коректний (зберігає відношення логічного слідування) перехід від одних теорем до інших, з одного боку, і формалізують певні стандартні способи міркувань – з іншого;

5) правила еквівалентного перетворення формул обчислення;

6) алгоритми, які ефективно розпізнають достовірність міркувань.

Історично числення предикатів виросло з логіки висловлювань, що є складовою частиною логіки предикатів. Тому спочатку коротко розглянемо в зазначених аспектах *логіку висловлювань*, а потім перейдемо до *логіки предикатів*.

3.3. Основи логіки висловлювань

3.3.1.Алфавіт логіки висловлювань.

Елементами (символами) алфавіту є:

- висловлювання, що позначаються малими літерами;
- п'ять логічних зв'язок (табл.3.1)

Висловлення – це оповідальне (*рос.* – повествовательное) (стверджувальне) речення, про яке можна сказати істинне воно або помилково (тобто має значення *True* або *False* чи *Істинне* або *Помилково*).

Таблиця 3.1.

Елементи і символи логіки висловлювань

Назва зв'язки	Позначення	Тип	Інші позначення
Заперечення	¬	Унарний	‾ , ~ , **not , ні**
Кон'юнкція	∧	Бінарний	& , ● , ***and , i***
Диз'юнкція	∨	Бінарний	**/ , *or* , *або***
Імплікація	⊃	Бінарний	⇒ , → , ⇨
Еквівалентність	≡	Бінарний	⇔ , ↔

Наведемо приклади висловлювань [13]:

1. Йде дощ. Ця пропозиція є висловлюванням, і його можна замінити (ідентифікувати), наприклад, буквою *q*.
2. Дорога мокра. Це теж висловлювання, і його можна ідентифікувати буквою *r*.

Надалі процес ідентифікації (іменування) конкретного висловлювання певною рядкової буквою будемо позначати символом

⇌ (дорівнює за визначенням).

3. *p* ⇌Сніг білий. *H* ⇌Президент іде у відставку.
4. *e* ⇌Земля крутиться. *n* ⇌Довжина кола дорівнює її діаметру.

Слід звернути увагу на те, що в висловлюванні представляється взаємозв'язок (відношення) об’єкта, що віддзеркалюється з його властивостями або характеристиками, або будь-якими діями об'єкта, або розглядається взаємозв'язок двох або більшої кількості об'єктів, що віддзеркалюються. Це важливо для розуміння семантики терміна «***предикат***». Наприклад:

5. *k* ⇌ Сторона *a*, сторона *b* и сторона *c* є сторонами рівнобедреного трикутника *T*.

У логіці висловлювання діють дві угоди:

1) Висновок про істинність (*І*) або хибності (*Х*) конкретного висловлювання дає суб'єкт, що моделює реальний світ, при постановці і вирішенні деякої задачі. Семантика (смислове значення) віддзеркалюваного висловлювання і представлення про її істинність повністю визначається моделлю світу цього суб'єкта. Цілком зрозуміло, що різні суб'єкти можуть мати або формувати різні моделі реального світу.

2) В основу логіки висловлювань належить дедуктивне міркування, тому вона має й іншу назву – «***числення висловів***».

3.3.2. Правила утворення мови в алфавіті (синтаксис мови).

Для опису правил введемо поняття ***метасимвол***. *Метасимвол* – це не позначення, що належить мові, а управляючий елемент який дозволяє вводити поняття і властивості цієї мови, а також вказати порядок, в якому повинні застосовуватися правила мови. Метасимволи представляють собою особливі символи для вказівки певних інструкцій в регулярному виразі.

Введемо чотири метасимвола для ілюстрації заданих правил прикладами:

1) x 2) y 3) (4)) (3.1) .

Метасимволи x і y будуть служити для позначення формул, а дужки «(» та «)» – для вказівки порядку застосування правил.

Правила утворення мови в алфавіті наступні:

<u>*Базисне правило:*</u> будь–яке висловлювання є формула.

<u>*Правило індукційного кроку:*</u> якщо x і y – формули, то $\neg x$, $(x \wedge y), (x \vee y), (x \supset y), (x \equiv y)$ – теж формули.

<u>*Правило обмеження:*</u> формули можуть утворюватися тільки за правилами 1 і 2.

Інших правил немає.

Далі, необхідно визначитися, як метасимволи дужок вказують на порядок застосування правил. Для цього розглянемо приклад.

Нехай, $x \rightleftharpoons (p \wedge (q \vee r))$. При побудові формули x правило індукційного кроку застосовувалося двічі: перший раз – при побудові формули $(q \vee r)$ з формул q і r, а другий – при побудові заключній формули з формул p і $(q \vee r)$.

Зазначені правила утворення мови в алфавіті використовуються для представлення складових як завгодно складних висловлювань.

Формули мови діляться на ***атоми*** або ***атомарні формули*** і ***формули*** (без епітетів), до яких відносяться всі складові формули, тобто формули, утворені за допомогою зв'язок, а атоми – це неподільні (вихідні) висловлювання.

3.3.3. Правила присвоєння істиннісних значень формулами (семантика мови)

За визначенням ***атом*** може мати тільки два значення: або «істина» (I) або «хибність» (X). Кожне з цих значень називають істінностним. Правила присвоєння істиннісних значень формулами п'яти зв'язок (зв’язувань) представлені в табл. 3.2.

Таблиця 3.2.

Таблиця істинності

x	y	⌉x	x ∧ y	x ∨ y	x ⊃ y	x ≡ y
I	I	X	I	I	I	I
I	X	X	X	I	X	X
X	I	I	X	I	I	X
X	X	I	X	X	I	I

Введемо ряд нових понять, які знадобляться при розгляді наступних аспектів обчислення висловлювань.

1) «***інтерпретувати формулу***» – приписати їй одне з двох значень істинності «I» чи хибності «X».

2) «***інтерпретація для формули***» – це набір істиннісних значень всіх атомів, що входять в формулу, призначений для одночасної заміни ними самих атомів в цій формулі. Формула, що містить *K* різних висловлювань, допускає $2^{к}$ інтерпретацій.

Проілюструємо ці два поняття на прикладах інтерпретації деяких формул (табл. 3.3).

Таблиця 3.3.

Інтерпретація деяких формул

x y z	*y ⊃ z*	*x ⊃(y⊃ z)*	*x ∧ y*	*(x ∧ y)⊃ z*	*w*	⌉*w*
I I I	X	I	I	I	I	X
I I X	X	X	I	X	I	X
I X I	I	I	X	I	I	X
I X X	I	I	X	I	I	X
X I I	I	I	X	I	I	X
X I X	X	I	X	I	I	X
X X I	I	I	X	I	I	X
X X X	I	I	X	I	I	X

Смислові елементи таблиці такі.

– перші три стовпці кожного рядка є однією з можливих інтерпретацій.

– «семантика мови» – це повний набір правил інтерпретації формул, тобто це вся таблиця 3.2.

– «загальнозначимість формули» – це істинність формули при всіх можливих її інтерпретаціях, тобто *w* в табл. 3.3).

– «суперечливість формули» (нездійсненність)– це хибність формули при всіх можливих її інтерпретаціях – ⌉ *w* в табл.3.3

– «еквівалентність формул» – формули *x* і *y* еквівалентні, коли істинності значення *x* і *y* збігаються при кожній загальній інтерпретації для *x* і *y*.

– «літера» – це *атом* або його заперечення; *атом* в математичній логіці – найпростіший випадок формули, тобто формула, яку не можна роздрібнити на підформули.

– «диз'юнкція формул» – це формула *X*, утворена з початкових формул $F_1, F_2, ..., F_n$ за допомогою ***диз'юнктивної*** (і тільки) зв'язки:

$$X = F_1 \vee F_2 \vee ,... \vee F_n. \qquad (3.2)$$

– «кон'юнкція формул» – це формула *Y*, утворена з початкових формул $F_1, F_2, ..., F_n$ за допомогою ***кон'юнктівної*** (і тільки) зв'язки:

$$Y = F_1 \wedge F_2 \wedge ,... \wedge F_n. \qquad (3.3)$$

–«диз'юнкт»– це формула *Z*, утворена з початкових літер (і тільки літер) за допомогою ***диз'юнктивній*** (і тільки) зв'язки:

$$Z = A_1 \vee A_2 \vee A_3 \vee ,... \vee An. \qquad (3.4)$$

Еквівалентом диз'юнкта є множина літер, що входять до нього:

$$Z = A_1 \vee A_2 \vee ,... \vee A_m \equiv \{A_1 , A_2 ,..., A_m\} \qquad (3.5)$$

– «R – літерний диз'юнкт» – це диз'юнкт, в якому R літер.

– «одиничний диз'юнкт» – це диз'юнкт з однією літерою.

– «порожній диз'юнкт» – це диз'юнкт, в якому немає літер. Оскільки він не містить літер, які могли б бути істинними при деякій інтерпретації, то він завжди хибний.

– «область дії логічних зв'язок» – при бездужковому запису ця область впорядкована по спадаючій і відповідає послідовності $\equiv , \supset , \wedge , \vee , \neg$.

–«діз'юнктівная нормальна форма»– це формула:

$F = F_1 \vee F_2 \vee ,... \vee F_n$, где F_i – кон'юнкція літер.

–«кон'юнктивна нормальна форма»– це формула:

$F = F_1 \wedge F_2 \wedge ,... \wedge F_n$, где F_i – диз'юнкція літер.

– «виконувана формула» – формула виконувана (*рос*. – выполнима) тоді і тільки тоді, коли існує, принаймні, одна інтерпретація, при якій ця формула істинна. Ця інтерпретація називається ***моделлю формули***.

–«контрарна пара формул»– це множина { • A, A}

–«тавтологія»– загальнозначуща формула, справжня у всіх її інтерпретаціях. Диз'юнкт, що містить контрарну пару, є тавтологією.

–«кон'юнктивна нормальна форма (КНФ)» в булевій логіці – нормальна форма, в якій булева формула має вигляд кон'юнкції диз'юнкцій літералів. Кон'юнктивна нормальна форма зручна для автоматичного доказу теорем.

–«приведена КНФ (ПКНФ)»– це КНФ, з якої вилучені тавтології і повторення літер в межах одного і того ж диз'юнкт.

3.3.4. Правила виведення в численні висловів (стереотипи дедуктивного міркування).

Стереотипи вироблені за багато десятків років, а деякі за багато сотень років, і дозволяють здійснювати коректні, тобто без порушення відносини логічного слідування, переходи від одних теорем до інших з метою приведення структури міркування до канонічної формі (до приведеної КНФ - ПКНФ). Сутність канонічної форми буде розкрита в підпункті **3.3.5**, а зараз, до обговорення правил виведення, введемо ряд нових понять, без яких це обговорення неможливо.

Поняття «відношення логічного слідування».

Формула G є логічним наслідком формул $F_1, F_2, ..., F_n$ тоді і тільки тоді, коли для кожної інтерпретації I, в якій $(F_1 \wedge F_2 \wedge ,... \wedge F_n.)$ – істина, G також істина. Формули$F_1, F_2, ..., F_n$ називаються ***посилками***, а G – заключенням по відношенню до логічного слідування.

Далі зупинимося на понятті «необхідні і достатні умови логічного слідування (наслідку)».

Формула G тоді і тільки тоді є логічним наслідком $F_1, F_2, ..., F_n$, коли формула $((F_1 \wedge F_2 \wedge ,... \wedge F_n.) \supset G)$ **–** загальнозначима, або коли формула $(F_1 \wedge F_2 \wedge ,... \wedge F_n \wedge \neg G)$– суперечлива (*рос*. – противоречива).

Поняття «теорема в дедуктивному міркуванні».

Якщо формула G є логічним слідством формул $F_1, F_2, ..., F_n$, то формула $((F_1 \wedge F_2 \wedge ,... \wedge F_n) \supset G)$ називається теоремою, а G називається висновком теореми.

Поняття «доказ в дедуктивному міркуванні» (тобто в обчисленні).

Доказ – це аргументоване обґрунтування того, яким чином висновок (*рос*. – заключение) в теоремі логічно випливає з її посилок. Доказ представляється у вигляді впорядкованої послідовності (сліду) умовиводів, в результаті яких встановлюється істинне значення висновку.

Тепер зупинимося на особливостях застосування різних ***метасимволів*** при оформленні правил виведення в ***численнях висловлювань***.

Для позначення відношення логічного слідування введемо новий метасимвол ——— (горизонтальна риска). Будемо використовувати цей метасимвол для розділення теорем–посилок і теорем–заключень. Над рискою будемо записувати список теорем-посилок, під рискою – теорему-заключення. Зазначена форма запису теорем буде свідчити про те, що теорема-посилка є логічним наслідком теорем–заключень.

Метасимвол «**;**» (крапку з комою) будемо використовувати як роздільник в списку теорем.

Метасимвол «**,**» (кому) будемо використовувати як роздільник посилок всередині теореми, що імітує кон'юнктівний зв'язок.

Метасимвол $\square$ (прямокутник) будемо використовувати для позначення суперечливих формул.

Розглянемо наступний приклад використання відносини логічного слідування. Нехай завдане наступне висловлювання.

«Якщо барометр падає, то буде погана погода».

Запишемо його у наступному вигляді.

Барометр падає

Буде погана погода

Перелік ***правил виведення*** у данному випадку буде наступний:

1) $\dfrac{\Gamma \supset \Phi\ ;\ \Gamma \supset \Psi}{\Gamma \supset \Phi \wedge \Psi}$ 2) $\dfrac{\Gamma \supset \Phi \wedge \Psi}{\Gamma \supset \Phi}$ 3) $\dfrac{\Gamma \supset \Phi \wedge \Psi}{\Gamma \supset \Psi}$ 4) $\dfrac{\Gamma \supset \Phi}{\Gamma \supset \Psi \vee \Psi}$

5) $\dfrac{\Gamma \supset \Psi}{\Gamma \supset \Phi \vee \Psi}$ 6) $\dfrac{\Gamma,\Phi \supset \Psi;\ \Gamma,X \supset \Psi;\ \Gamma \supset \Phi \vee X}{\Gamma \supset \Psi}$ 7) $\dfrac{\Gamma,\Phi \supset \Psi}{\Gamma \supset \Psi \supset \Psi}$

8) $\dfrac{\Gamma \supset \Phi\ ;\Gamma \supset \Phi \supset \Psi}{\Gamma \supset \Psi}$ 9) $\dfrac{\Gamma, \neg\Phi \supset \square}{\Gamma \supset \Phi}$ 10) $\dfrac{\Gamma \supset \Phi;\ \Gamma \supset \neg\Phi}{\Gamma \supset \square}$

11) $\dfrac{\Gamma,\Phi,\Psi,\ \Gamma_1 \supset X}{\Gamma,\ \Psi,\Phi,\ \Gamma_1 \supset X}$ 12) $\dfrac{\Gamma \supset \Phi}{\Gamma,\ \Psi \supset \Phi}$

Правила 1, 2 та 3, експлуатуючи сутність кон'юнктивної зв'язки і дозволяють спростити модель теореми.

Правила 4 та 5, експлуатуючи сутність диз'юнктивної зв'язки, дозволяють в заключенні теореми вводити нові додаткові формули.

Правило 6 формує спосіб міркування «розбір» двох можливих випадків. Якщо при виконанні посилок Г справедливо Ф або Х, а Ψ справедливо при виконанні умов Г і Ф, а також при виконанні умов Г і Х, то Ψ завжди справедливо при виконанні посилок Г, що встановлюється шляхом розгляду двох можливих випадків:

а) виконані умови Г і Ф;

б) виконані умови Г і Х.

Правило 7 формалізує прийом еквівалентного переформулювання теореми, що дозволяє одну з посилок теореми поміщати у заключення у вигляді посилки.

Правило 8 – правило виводу (відділення), **modus ponens**, введене ще Аристотелем. Воно вказує, як можна звільнятися від посилки у заключенні.

Правило 9 – формалізує «міркування від противного». Нехай, умова Г і $\neg\Phi$ можуть одночасно виконуватися. Приходячи до протиріччя, робимо висновок, що із здійсненності Г завжди випливає здійсненність Ф.

Правило 10 – це правило «виявлення протиріччя».

Правило 11 –перестановка посилок не впливає на істинність заключення.

Правило 12 – додаючи зайву посилку, ми не порушуємо істинність заключення теореми.

Висновки: правила виведення 1 – 12 є функціонально повним набором правил, застосовуючи які до вихідних аксіом і доказаним теоремам, можна врешті–решт отримати доказ справжнього значення заключення в теоремі – цілі. Однак обов'язковими вимогами побудови доказу за допомогою правил виведення 1 – 12 є наступні:

1) вихідними посилками повинні бути тільки аксіоми і доказані теореми;

2) за допомогою правил виведення 1 – 12 (і тільки цих правил) можна будувати композиції аксіом і доказаних теорем, прагнучи у підсумку отримати заключення теореми-мети. Ці дві вимоги відповідають стереотипу класичного дедуктивного міркування, іноді званого прямою дедукцією, тобто міркування від істинних абстрактних посилок (аксіом і доказаних теорем) до істинного конкретного заключення. Розглянемо приклади недотримання зазначених вимог:

① а) Всі метали – елементи;

б) Бронза – метал

Бронза – елемент

Висновок є помилковим: бронза не є елементом. Тут порушений закон тотожності, який забороняє в процесі даного умовиводу в одне і те ж поняття вкладати різний зміст. У посилках а) і б) метал вжито не в однаковому розумінні. У посилці а) метал – це хімічний елемент, а в посилці б) метал – це речовина, що використовується в господарстві та виробництві.

② Бекон і Гоббс були єгиптянами

Бекон і Гоббс були ідеалістами

Деякі ідеалісти були єгиптянами

Висновок в умовиводі вірний, проте, обидві посилки хибні: Бекон і Гоббс були англійцями і матеріалістами (а не ідеалістами).

Практика показала, що спосіб доказу теорем прямою дедукцією надзвичайно неефективний з наступних причин. Структура композицій на проміжних етапах перетворень є випадковою в зв'язку з тим, що не існує критеріїв і керуючих впливів, які цілеспрямовано орієнтували б ці композиції до заданої мети. Тому не виключена можливість відходу в бік від мети, що найчастіше і буває на практиці. З цього випливає глобальний висновок про неефективність прямої дедукції при доказі теорем. У зв'язку з цим правила 1 – 12 найчастіше використовуються тільки для приведення структури теореми до канонічної формі.

3.3.5. Правила еквівалентних перетворень формул обчислення висловлювань.

Еквівалентні перетворення формул потрібні для того, щоб привести структуру цільової теореми до ***канонічної форми*** (КФ). КФ являє собою приведену нормальну форму з диз'юнктами замість діз'юнктівних формул і з деякими іншими принциповими особливостями, які будуть розглянуті пізніше.

Якщо в посилках і заключенні присутні формули, що не відповідають зазначеним вимогам, то вони повинні бути коректно замінені еквівалентними

формулами. Це дозволяє визначити правила еквівалентних перетворень (табл. 3.4), де використовуються два нових метасимвола.

$\boxtimes$ (конверт) – для позначення загальнозначущих формул;

= (равенство) – для позначення рівності формул.

Приклад: На підставі табл. 3.4 отримаємо діз'юнктівную форму для формули

$(P \vee \neg Q) \rightarrow R$

$(P \vee \neg Q) \rightarrow R = \neg (P \vee \neg Q) \vee R =$ (за правилом 2)

$= (\neg P \vee \neg (\neg Q)) \vee R =$ (за правилом 10а)

$= (\neg P \wedge Q) \vee R$ (за правилом 9).

Таблиця 3.4.

Правила еквивалентних перетворень

№ правила	Правила еквівалентних перетворень формул Обчислення висловлювань
1	$F \equiv G = (F \rightarrow G) \wedge (G \rightarrow F)$
2	$F \rightarrow G = \neg F \vee G$
3	а. $F \vee G = G \vee F$; б. $F \wedge G = G \wedge F$
4	а. $(F \vee G) \vee H = F \vee (G \vee H)$; б. $(F \wedge G) \wedge H = F \wedge (G \wedge H)$
5	а. $F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$; б. $F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$
6	а. $F \vee \square = F$; б. $F \wedge \boxtimes = F$
7	а. $F \vee \boxtimes = \boxtimes$; б. $F \wedge \square = \square$
8	а. $F \vee \neg F = \boxtimes$; б. $F \wedge \neg F = \square$
9	$\neg (\neg F) = F$
10	а. $\neg (F \vee G) = \neg F \wedge \neg G$; б. $\neg (F \wedge G) = \neg F \vee \neg G$

Інших правил еквівалентних перетворень не існує.

3.3.6. Алгоритми, що ефективно розпізнають достовірність міркувань.

Не будемо розглядати всю понад тисячолітню історію пошуку ефективних алгоритмів доказів теорем, і проробляти порівняльний аналіз всіх відомих алгоритмів. Досить навести лише ті алгоритми, які знайшли широке практичне застосування в штучному інтелекті.

Найбільш відомим алгоритмом називається ***метод резолюцій*** (***резолютивний вивід***). Він зародився в 1930 році (в роботах Ербрана), розвиток і становлення отримав в 1965 році (в роботах Робінсона), широке застосування отримав з початку 70–х років (в роботах групи А. Кольмрауера, яка застосовувала його на базі мови Пролог для доказу теорем) і використовується по теперішній час.

Відомо багато варіантів застосування методу резолюцій, але найвища його ефективність проявляється в наступних умовах:

1) коли цільова теорема перетворена в протилежну теорему;

2) коли цільова теорема представлена в структурі приведеної кон'юнктівной нормальної форми (ПКНФ) з наступною принциповою особливістю: диз'юнкт в ПКНФ є диз'юнктом Хорна.

ПКНФ, яка задовольняє 2-й вимозі, називається канонічною формою, що має вигляд:

$$(P_1 \vee P_2 \vee ...) \wedge (q_1 \vee q_2 \vee ...) \wedge ... \wedge (r_1 \vee r_2 \vee ...) \wedge \neg (C_1 \vee C_2 \vee ...), \qquad (3.6)$$

де, P_i, q_j, r_k – літери посилок, а C_t – літери заключення (поняття «протилежна теорема» і «диз'юнкт Хорна» будуть розглянуті пізніше);

3) коли при доказі цільової теореми користуються дедуктивним міркуванням, званим зворотньою дедукцією, яке докорінно відрізняється від прямої (класичної) дедукції (про зворотну дедукції буде сказано пізніше);

4) коли доводять суперечливість канонічної форми цільової теореми, використовуючи при цьому ту перевагу ПКНФ, що вона стає суперечливою, якщо хоч один з її диз'юнктів хибний. Це дає право вважати еквівалентом ПКНФ множини її диз'юнктів, а перевірка суперечливості ПКНФ зводиться до перевірки істинності кожного з диз'юнктів.

А) Суть методу резолюцій (правило 1).

Застосування методу резолюцій базується на фундаментальному методі дедукції: множина формул дедуктивного міркування нездійснена тоді і тільки тоді, хибність (Х) є логічним наслідком його (цієї множини).

Цю ж властивість можна сформулювати інакше:

Формула С є логічним наслідком кінцевої множини формул Е тоді і тільки тоді, коли, $E \wedge \neg C$ не здійснимо. Ця властивість дедуктивного міркування, що випливає з його визначення, широко відоме як принцип дедукції.

У формальному представленні воно виглядає так:

$$\{ H_1, H_2 ..., H_n \} \models C \equiv \{ H_1, H_2, ..., H_n, \neg C \} \models X, \qquad (3.7)$$

де метасимвол $\models$ – відношення логічного слідування.

H_i (i = 1, n) – посилка в дедуктивному міркуванні, С – висновок в міркуванні.

Ґрунтуючись на цій властивості, нездійсненність множини Е диз'юнктів канонічної форми цільової теореми дедуктивного міркування можна перевірити, породжуючи логічні наслідки з Е до тих пір, поки не отримаємо порожній диз'юнкт.

Для породжень логічних наслідків будемо використовувати наступну схему міркувань. Нехай А, В і Х – формули.

Припустимо, що дві формули $(A \vee X)$ і $(B \vee \neg X)$ – істинні. Якщо формула Х – теж істинна (І), то $\neg X$ – Х і тоді можна зробити висновок, що В –істинна.

Навпаки, якщо Х – хибна (помилкова), то можна зробити висновок, що А – хибна. Таким чином, в обох випадках $(A \vee B)$ – істинна.

Отримуємо правило $\{A \vee X, B \vee \neg X\} \models A \vee B$, яке можна записати також у вигляді:

$$\{\neg X \to A,\ X \to B\} \models A \vee B . \qquad (3.8)$$

У тому окремому випадку, коли X – висловлювання, A і B – диз'юнкт, це правило називається ***правилом резолюцій***.

Б) Сутність перетворення цільової теореми в протилежну теорему (правило 2).

У класичній математиці поняття «протилежна теорема» означає, що висновок протилежної теореми є запереченням укладення вихідної теореми, а посилки протилежної теореми є запереченням (інверсією) посилок вихідної теореми. Наприклад, $A \to B$ – вихідна (початкова) теорема, $\neg A \to \neg B$ – протилежна теорема. Так як заперечення загальнозначущої формули – суперечлива формула, то і доказувати слід суперечливість, а не общезначимість протилежної формули.

Таким чином, якщо вихідна цільова теорема має вигляд:

$$(D_1 \wedge D_2 \wedge ... \wedge D_n) \to C, \qquad (3.9)$$

де D_i – диз'юнкт Хорна, C – заключення теореми, то, перетворюючи формулу (1) за правилом (2), отримаємо нову формулу без імплікації:

$$(\neg (D_1 \wedge D_2 \wedge ... \wedge D_n) \vee C). \qquad (3.10)$$

Перетворюючи формулу (2) в протилежну формулу, отримаємо ПКНФ:

$$(D_1 \wedge D_2 \wedge ... \wedge D_n \wedge \neg C) \qquad (3.11)$$

ПКНФ (3) буде суперечлива, якщо який–небудь диз'юнкт D_i або $\neg C$ – хибний.

Звідси випливає просте правило перевірки формули (3.11) на суперечливість: слід перевірити кожен елемент множини диз'юнктів формули на його хибність. Найвищу ефективність такої перевірки дає метод резолюцій.

Таким чином, перетворення цільової теореми в протилежну теорему необхідне для отримання чистої ПКНФ, яка потребує доказу своєї суперечливості.

В)Диз'юнкти Хорна та їх призначення.

Диз'юнкти Хорна – це диз'юнкти, які містять не більше однієї позитивної літери.

Диз'юнкт $(\neg p \vee \neg q \vee \neg r \vee S)$ еквівалентний імплікації $(p \vee q \vee r) \to S$.

Загальнозначуща імплікація – це доказана теорема. Тому диз'юнкти Хорна використовуються в канонічній формі цільових теорем як еквіваленти посилок, що представляють собою доказані теореми. Ці посилки – доказані теореми – називаються ***правилами***. Правила широко використовуються при автоматичному доказі теорем.

Диз'юнкт, що зводиться до одиночної позитивної літери, представляє деякий ***факт***, тобто заключення, що не залежить ні від яких посилок. Факти також широко використовуються при доказі теорем.

Крім доказаних властивостей диз'юнктів Хорна, відзначимо ще дві їх якості. Перша полягає в тому, що при представленні посилок виключно диз'юнктами Хорна метод резолюцій обов'язково завершує доказ, чого може

не відбутися в разі нехорнівського представлення посилок, коли метод може зациклитися.

Друга полягає в тому, що обчислювальна трудомісткість доказу цільової теореми методом резолюцій мінімальна і не перевищує n^2, де n – кількість літер в канонічній формі з урахуванням повторень.

Г) Сутність зворотньої дедукції.

Основним недоліком прямий дедукції є те, що в ній не існує критеріїв і керуючих впливів, які направляли б слід доказу до заданої мети і не дозволили б доказу піти в сторону від мети. У зворотній дедукції ці недоліки усунуті.

Замість того щоб починати з посилок, зворотня дедукція починає з ув'язнення–мети і, застосовуючи правила–посилки (доказані теореми), підміняє поточні цілі новими до тих пір, поки ці нові цілі не стануть літерами-фактами. Тим самим процедура доказу стає керованою метою виведення на кожному кроці умовиводів, що не дозволить їй збитися з істинного шляху. Сам шлях перетворюється в критичний, тобто в найкоротший (приклади будуть приведені при численні предикатів, розділ 4).

Д) Правила доказу теорем методом резолюцій.

Вихідним об'єктом для доказу є повна множина диз'юнктів, що належать канонічній формі цільової теореми. Позначимо ці множини символом D. Впорядкуємо D за деякими критеріями, тим самим перетворивши його в нумерований список, який позначимо S.

Докази суперечливості списку S будемо виконувати за такою рекурсивної схемою.

Беремо цільової диз'юнкт, що стоїть в голові списку S, і співставляємо його з усіма подальшими диз'юнктами, що стоять після нього. На кожному кроці зіставлення пар диз'юнктів стежимо за появою контрарной пари літер $\{A, \neg A\}$. Якщо її немає, то переходимо до наступного диз'юнкту, не роблячи ніяких дій. Якщо ж з'являється контрарна пара літер в парі диз'юнктів, що співставляються, то, викресливши її і породивши резольвенту r, запишемо її в кінець списку S і дамо їй номер на одиницю більше номера останнього елемента у вихідному списку S. Резольвента r дописується в список за умови, що вона не є тавтологією і не поглинається якимось диз'юнктом зі списку S. У цих випадках вона просто ігнорується.

Після того як перший головний диз'юнкт списку S зіставлений з усіма подальшими диз'юнктами початкового списку S, а обчислені резольвенти поставлені в хвіст списку S, процедура попарного зіставлення елементів повторюється тепер вже для нового диз'юнкта. Таким новим диз'юнктом стає другий номер диз'юнкта з тепер вже розширеного списку S_p. Рекурсивно повторюються процедури викреслювання контрарних пар, породження резольвент, приписування резольвент в хвіст списку за аналогією з попереднім циклом. Але перший елемент списку Sp вже в зіставленні участь не бере, а в цьому зіставленні беруть участь ті резольвенти, які були породжені в першому циклі. Рекурсія з наступними головними елементами триває до тих пір, поки поточний список не перетворюється в порожній

диз'юнкт або в набір неспівставляємих диз'юнктів. Якщо з'являється порожній диз'юнкт, то теорема доказана, в іншому випадку – заключення хибне.

Порожній диз'юнкт позначається Λ і означає хибність (*false*), або тотожню нулю формулу.

Від критеріїв впорядкування списку S залежить ефективність доказу. Критерії упорядкування списків такі:

1) першим в список S записуються диз'юнкти заключення в будь–якому порядку;

2) у другу чергу записуються факти в будь-якому порядку;

3) у третю чергу записуються діз'юнкти-правила, але в порядку, що відповідає порядку фактів, тобто якщо на першому місці стоїть факт a_1, то правило, що включає факт a_1 в підсписок правил, теж має стояти на першому місці.

Розглянемо приклади доказу теорем методом резолюцій.

Приклад 1.

Приведемо доказ общезначимості правила 6, тобто правила розбору можливих випадків (див. п. 3.2.4). Потрібно доказати:

$\{ h, h\rightarrow (p\vee q) , p\rightarrow c , q\rightarrow c \} \rightarrow c$

Множина диз'юнктів, нездійсненність якого слід встановити, таке:

$\{ h , \neg h\vee p\vee q , \neg p\vee c, \neg q\vee c , \neg c \}$

Методом резолюцій порожній диз'юнкт отримаємо таким чином:

1) $\neg c$ – заперечення (*рос*. – отрицание) заключення
2) h – гіпотеза
3) $\neg h \vee p \vee q$ – гіпотеза
4) $\neg p \vee c$ – гіпотеза
5) $\neg q \vee c$ – гіпотеза
6) $\neg p$ – резольвента для 1 і 4
7) $\neg q$ – резольвента для 1 і 5
8) $p \vee q$ – резольвента для 2 і 3
9) q – резольвента для 6 і 8
10) p – резольвента для 7 і 8
11) Λ – резольвента для 7 і 9.

Приклад 2

Докажемо нездійсненність (*рос*. – невыполнимость) множини диз'юнктів.

$\{ p \vee q , p \vee r , \neg q \vee \neg r , \neg p \}$

1) $p \vee q$
2) $p \vee r$
3) $\neg q \vee \neg r$
4) $\neg p$
5) $p \vee \neg r$ – резольвента для 1 та 3

6) q – резольвента для 1 і 4
7) $p \vee \neg q$ – резольвента для 2 і 3
8) r – резольвента для 2 і 4
9) p – резольвента для 2 і 5
10) $\neg r$ – резольвента для 3 і 6
11) $\neg q$ – резольвента для 3 і 8
12) Λ – резольвента для 4 і 9.

4. Логіка предикатів

4.1.Введеня в логіку предикатів

Предикат (*лат.* Praedicatuum – сказане) – це присудок затвердження, тобто те, що висловлюється (затверджується або заперечується) в судженні про суб'єкта. Наприклад, у твердженні «ракета досягла Місяця» предикат – «досягла Місяця». У логіці висловлювань атом розглядається як неподільне ціле, структура і склад якого не піддаються зовнішньому аналізу в процесі міркувань. Однак є багато думок, які не можуть бути розглянуті простим способом. Наведемо приклади.

Приклад 1. Мовою логіки висловлювань не можна висловити той факт, що з пропозиції «щонайменше, один студент вирішив всі контрольні завдання» слідує висновок «кожну контрольну задачу вирішив, щонайменше, один студент». Кожне з цих висловлювань є висловлюванням, імплікація яких не є тавтологією.

Приклад 2. Нехай a, b, c – прямі на площині, || – символ паралельності прямих, тоді з посилок ***x: a*** || ***b*** і ***y: b*** || ***c*** треба зробити висновок ***z: a*** || ***c***. Однак в логіці висловлювань теорема $\boldsymbol{x} \wedge \boldsymbol{y} \rightarrow \boldsymbol{z}$ не є загальнозначущої формулою.

Приклад 3. ***P:*** Кожна людина смертна
Q: Конфуцій – людина
R: Конфуцій – смертний.

Заключення ***R*** теореми здається очевидним, проте, в логіці висловлювань формула $(P \wedge Q) \rightarrow R$ не є тавтологією.

Протиріччя в наведених прикладах випливають з відсутності можливості проникнути у внутрішню структуру кожного з висловлювань, побачити в цій структурі взаємозв'язок між об'єктами в кожному з них і, найголовніше, відобразити цей взаємозв'язок у вигляді формули.

Для ліквідації цих недоліків і призначена логіка предикатів, яка є розширенням логіки висловлювань, тобто разом з поняттями логіки висловлювань вона містить ряд інших понять, що підсилюють логічне мислення.

У логіку предикатів, крім символів логіки висловлювання, додатково введені символи розкритих об'єктів, символи властивостей і характеристик цих об'єктів, функціональні символи, а також символи відносин між

об'єктами, їх характеристиками і властивостями. Крім того, введені також узагальнюючі і конкретизують вираження «все» і «деякі» (так звані квантори), що дозволяють кількісно охарактеризувати зв'язку об'єктів, властивостей, характеристик і відносин. Доповнення логіки висловлювань зазначеними символами дозволяє явно задавати і управляти об'єктами, відносинами, властивостями і характеристиками об'єктів, тобто здійснювати зовнішнє втручання у внутрішню структуру висловлювань.

4.2. Алфавіт логіки предикатів.

Символами (елементами) алфавіту логіки предикатів є наступні.

1) Константи (індивідуальні символи). Зазвичай це імена об'єктів, найменування властивостей, характеристик, значення властивостей або характеристик.

2) Символи предметних змінних. Зазвичай, це малі літери x, y, z, можливо з індексами. Предметна змінна передбачає наявність області її визначення, тобто кінцевої множини значень даної змінної.

3) Функціональні символи. Зазвичай, це малі літери f, q, h або осмислені слова з малих літер, наприклад, плюс, мінус, помножити, батько, мати, дочка.

4) Предикатні символи. Зазвичай, це прописні (великі) букви P, Q, R, або осмислені слова, які складаються з великих літер, такі як БІЛЬШЕ, МЕНШЕ, ЛЮБИТЬ, МІСТИТЬ, СМЕРТЕН, ЛЮДИНА.

5) Зв'язки обчислення (*рос*. – исчисления) висловлювань ($\neg, \wedge, \vee, \supset, \equiv$).

6) Квантори спільності $\forall$ (для всіх) та існування $\exists$ (існує).

У мові предикатів міститься і мова висловлювань, так як предикат стає висловлюванням після заміни його предметних змінних їх конкретними значеннями.

4.3. Правила утворення мови в алфавіті (синтаксис)

Алфавіт логіки предикатів дозволяє визначити нові поняття, необхідні для мови цієї логіки.

Поняття «відношення». У логіці предикатів предикат виражає взаємний зв'язок поміж об'єктами, що відображаються, зі всіма своїми властивостями, характеристиками, діями, а також взаємозв'язок об'єктів, процесів, дій. Кількість зазначених предметів в одному відношенні не обмежується. Відношення може бути двомісним (тобто між двома предметами), тримісним, і взагалі n–місцевим. Істинне висловлювання є відношенням. Формально найпоширеніше двомісне відношення (бінарне) розглядається як множина пар елементів і позначається вектором <x, y>, де x називається першим елементом, а y – другим елементом координатою пари. Векторне представлення n-мірних відносин теж широко поширене.

Поняття «терм» – це будь–яка константа, предметна змінна і функція.

Поняття «функція» символізує дію, що ставить у відповідність списку констант одну певну константу. Будь-яка функція має визначену кількість аргументів. Якщо функціональний символ f має n аргументів, то f називається n-місним функціональним символом. Константа може розглядатися як функціональний символ без аргументів.

Позначення функції – f (t_1, t_2, ... t_n), де t_i – терми; при n = 0 функція позначається просто f.

Приклади:

1) плюс (t_1, t_2, t_3) означає $t_1 + t_2 + t_3 = t_\Sigma$

2) плюс (плюс (t_1, t_2, t_3, t_4), мінус (t_5, t_6), помножити (t_1, ..., t_n))

3) батько (батько (іван)), що визначає дідуся Івана.

Предикат – це узагальнення поняття «висловлювання», що полягає:

1) в абстрагуванні предметів, що беруть участь у взаємозв'язках цього висловлювання;

2) в забезпеченні можливості зовнішнього управління внутрішніми взаємозв'язками цього висловлювання за допомогою кванторних символів.

Будь–який предикат має певне число аргументів. Якщо предикатний символ P має n аргументів, то він називається n-місним предикатним символом.

Позначення предиката – P (t_1, ..., t_n), де t_i – терми; при n = 0 предикат позначають P замість P ().

Предикат символізує дію, що ставить у відповідність списку констант одне з істиннісних значень I (Істина) чи X (Хибність). Це означає, що предикат (як висловлювальна форма) після заміни в ньому аргументів – предметних змінних конкретними константами перетворюється в висловлювання, яке може бути або істинним (I), або хибним (X). Істинний предикат називається ***відношенням***.

Так само, як і в висловлюванні, в предикаті виражаються взаємозв'язки між різними предметами: між відображуваним об'єктом і його властивостями, характеристиками і діями, між парами або великою кількістю об'єктів, процесів, явищ.

Предикатний символ, як правило, представляє присудок (*рос*. – сказуемое) в розповідному (*рос*. – повествовательном) реченні, про який можна сказати, істинне воно або хибне. Але дуже часто в предикатному символі відбивається семантика взаємозв'язку предметів, що відбиваються.

Приклади:

1) ЛЮБИТЬ (Іван, Марія)

2) ЛЮБИТЬ (батько (Іван), Іван)

3) БІЛЬШЕ (x, 3), де x $\in$R – множина натуральних чисел.

4) БІЛЬШЕ (плюс (x, 1), x)

5) СТОЛИЦЯ (x, y)

де x $\in${Кишинів, Київ, Мінськ},

y $\in${Молдавія, Україна, Білорусія}

6) МІСТИТИ (x, y, v)

де x ∈{вуглець, сірка},
y ∈{сталь, кремній},
v ∈{множина дійсних чисел}
7) ПОСТАВКА (x, y, z, i, j, k, l, m),
де x | чого? | ∈{список комплектуючих},
y | кому? | ∈{список одержувачів},
z | хто? | ∈{список постачальників},
i | звідки? | ∈ { список джерел},
j | куди? | ∈ {список приймачів},
k | скільки? | ∈ {список кількості комплектуючих},
l | за ціною? | ∈ {список цін}?
m | транспортні витрати? | ∈ {список питомих вартостей перевезень}
8) МАТИ ВЛАСТИВОСТІ І ХАРАКТЕРИСТИКИ (z, x1, x2, x3, y1, y2, y3),
де z | об'єкт, що відображається? | ∈{список об'єктів, що відображаються },
x1 | колір? | ∈ {список квітів},
x2 | запах? | ∈ {список запахів},
x3 | смак? | ∈ {список смаків},
y1 | геометрична форма? | ∈ {список геометричних форм},
y2 | твердість? | ∈{список показників твердості},
y3 | питома вага? | ∈ {список питомих ваг}.

Поняття «атом» – це предикат, представлений у вигляді формули P (t_1, t_2, ..., t_n).

Синтаксис мови логіки предикатів визначається рекурсивно наступними правилами.

1) Атом є формула.

2) Якщо F і G – формули, то (¬F), (F ∨ G), (F ∧ G), (F→ G) і (F≡ G) – формули.

3) Якщо F – формула, а x – предметна змінна F, то $(\forall x)$ F і $(\exists x)$ F – формули.

4) Формули породжуються тільки кінцевим застосуванням правил 1–3.

Область дії квантора – це та формула, до якої він застосовується.

Наприклад, областю дії квантора існування у формулі $(\forall x)(\exists y)$ МЕНШЕ(x, y) є формула МЕНШЕ(x, y), а область дії квантора всезагальності є формула $(\exists y)$ МЕНШЕ(x, y).

У формулі $(\forall x)(Q(x) \to R(x))$ область дії квантора всезагальності є формула $(Q(x) \to R(x))$.

Входження змінної x в формулу називається пов'язаним тоді і тільки тоді, коли воно співпадає з входженням до кванторного комплексу $(\forall x)$ або $(\exists x)$, або знаходиться в області дії такого комплексу.

Входження змінної в формулу вільно тоді і тільки тоді, коли воно не є зв'язаним.

Наприклад, у формулі $(\forall x)P(x,y)$ змінна x зв'язана, тому що обидва входження x зв'язані. Однак, змінна y вільна, тому що єдине входження y вільно.

Змінна в формулі може бути вільною та зв'язаною, наприклад, змінна і вільна, і пов'язана в формулі $(\forall x)P(x,y)\wedge(\forall y)Q(y)$.

Поняття «семантика квантора загальності $\forall$». Квантор загальності $\forall$ встановлює правило інтерпретації формули F, до якої він застосовується. При кожному конкретному значенні змінної x (яку квантор $\forall$ зв'язує у формулі F), що належить області визначення змінної x, формула F, як функція від x, може приймати або тільки істинні значення, або тільки хибні, але поєднання І і Х неприпустимо на всій області визначення.

Поняття «семантика квантора існування $\exists$».

Квантор існування $\exists$ встановлює правило інтерпретації формули F, до якої він застосовується. Хоча б при одному конкретному значенні змінної x (яку $\exists$ зв'язує у формулі F), що належить області визначення x, формула F, як функція від x, обов'язково приймає істинне (або хибне) значення.

Наведемо приклад міркування, записаного мовою логіки предикатів.

1) кожна людина смертна.

2) Сократ – людина, отже, Сократ – смертний.

«x – є людина» позначимо через ЛЮДИНА(x),

«x – смертний», через СМЕРТНА(x).

Перше твердження представимо формулою:

$(\forall x)($ЛЮДИНА$(X)\rightarrow$СМЕРТНА$(x))$.

Твердження «Сократ – людина» – це формула ЛЮДИНА (Сократ), а твердження «Сократ – смертний» – це формула СМЕРТНА(Сократ).

В цілому міркування представляється формулою:

$(\forall x)($ЛЮДИНА$(x)\rightarrow$ СМЕРТНА $(x))\wedge$ЛЮДИНА (Сократ)

$\rightarrow$СМЕРТНА(Сократ).

4.4. Правила присвоєння істиннісних значень формулам (семантика мови).

Щоб визначити інтерпретацію для формули логіки предикатів слідує задати:

- загальну область визначення D щодо всіх предметних змінних, що входять до складу формули;
- значення констант;
- істинністні значення предикатів, що входять до складу формули;
- значення функцій, що входять до складу формули.

При цьому:

1) кожній константі відповідає деякий елемент з D;

2) кожному n–місному функціональному символу ставиться у відповідність відображення з D^n у D (зауважимо, що $D^n = \{(x_1,\ldots,x_n) / x_1 \in D,\ldots,x_n \in D\}$

3) кожному n–місному предикатному символу ставиться у відповідність відображення D^n у множину {I, X}.

Для кожної інтерпретації формули з області D формула може отримувати істиннісного значення I або X згідно з наступними правилами:

1. якщо задані значення формул G і H, то істиннісні значення формул $\neg G, ((G \wedge H),(G \vee H),(G \to H),(G \equiv H))$ відповідають таблиці 3.2 логіки висловлювань.

2. $(\forall x)$ G набуває значення I, якщо G набуває значення I для кожного x із D; інакше вона набуває значення X.

3. $(\exists x)G$ набуває значення I, якщо g набуває значення I хоч би для одного x з D; інакше вона набуває значення X.

4. Формула, що містить вільні змінні, не може отримати істиннісні значення. У логіці предикатів діє наступна угода: формула або не містить вільних змінних, або вільні змінні розглядаються як константи.

Приклад: Оцінимо формули

a) $(\exists x)(P(f(x)) \wedge Q(x, f(a)))$

b) $(\exists x)(P(x) \wedge Q(x,a))$

c) $(\forall x)(\exists y)(P(x) \wedge Q(x,y))$

У наступній інтерпретації $\mathbf{I_1}$:

$D = \{1,2\}, a = 1,\ f(1) = 2,\ f(2) = 1, P(1) = X,\ P(2) = \text{I},\ Q(1,1) = \text{I},$
$Q(1,2) = I,\ Q(2,1) = X,\ Q(2,2) = I.$

Для формули **a)** : якщо $x = 1$, то

$P(f(x) \wedge Q(x, f(a)) = P(f(1)) \wedge Q(1, f(a)) = P(2) \wedge Q(1, f(1)) =$
$P(2) \wedge Q(1,2) = I \wedge I = I\ ;$

якщо $x = 2$, то

$P(f(x) \wedge Q(x, f(a)) = P(f(2) \wedge Q(2, f(1)) = P(1) \wedge Q(2,1) =$
$X \wedge X = X.$

Так як в області визначення D існує елемент, а саме $x = 1$, такий, що $P(f(x) \wedge Q(x, f(a))$ істинна, то a) істинна при інтерпретації $\mathbf{I_1}$.

Для формули **b)**:

якщо $x = 1$, то

$P(x) \wedge Q(x,a) = P(1) \wedge Q(1,1) = \text{X} \wedge \text{I} = \text{X};$

якщо x=2, то

$P(x) \wedge Q(x,a) = P(2) \wedge Q(2,1) = \text{I} \wedge \text{X} = \text{I}.$

Так як в області визначення D не існує такого елемента, що $P(x) \wedge Q(x,a)$ істинна, то формула b) буде хибною при інтерпретації $\mathbf{I_1}$.

Для формули **c)**:

якщо $x = 1$, то $P(x) = P(1) =$ X. Отже, $P(x) \wedge Q(x, y) =$ X для $y = 1$ та для $y = 2$. Так як існує x, а саме $x = 1$, такий, що значення $(\exists y)(P(x) \wedge Q(x, y)$ хибне, то формула c) хибна при інтерпретації $\mathbf{I_1}$, тобто ця формула спростовується інтерпретацією $\mathbf{I_1}$. Визначимо ряд найважливіших понять щодо логіки предикатів.

Несуперечлива (здійснима) формула – формула G здійснима (несуперечлива) тоді і лише тоді, коли існує така інтерпретація $\mathbf{I_1}$, що G має значення I (Істина) в інтерпретації $\mathbf{I_1}$. Якщо формула G є I в інтерпретації $\mathbf{I_1}$, то $\mathbf{I_1}$ є моделлю формули G, та $\mathbf{I_1}$ задовольняє формулі G.

Суперечлива формула – коли не існує ніякої інтерпретації, яка б задовольняла G.

Загальнозначуща формула – коли не існує ніякої інтерпретації $\mathbf{I_1}$, яка б не задовольняла G.

Логічний наслідок – формула G є логічним наслідком формул $F_1, F_2, ..., F_n$ тоді і лише тоді, коли для кожної інтерпретації $\mathbf{I_1}$, якщо $F_1 \wedge F_2 \wedge ..... \wedge F_n$ істинна в $\mathbf{I_1}$, то G також є істиною в інтерпретації $\mathbf{I_1}$.

Зауваження: Оскільки в логіці предикатів кількість областей визначення предметних змінних ніяк не обмежується, тобто їх може бути нескінченна кількість, то є нескінченна кількість інтерпретацій формули. Отже, на відміну від логіки висловлювань, відсутня можливість доказу загальнозначущості або суперечності формули шляхом визначення її істинносних значень за усіма можливими інтерпретаціями, навіть у найпростіших випадках. Тому доказ теорем (достовірності міркувань) у логіці предикатів здійснюється тільки методом резолюцій, котрий має свої особливості.

4.5. Правила виводу у численні предикатів.

Логіка предикатів – це розділ ***символічної логіки***, що вивчає ***міркування*** і інші ***мовні контексти*** з урахуванням внутрішньої структури простих висловлювань, що входять в них, при цьому вирази мови трактуються функціонально, тобто як ***знаки*** деяких ***функцій*** або ж як ***знаки аргументів*** цих функцій [4,5].

Логіка предикатів, також, як і логіка висловлювань, заснована на дедуктивному міркуванні. Тому правила виводу логіки висловлювань, приведені в п. 3.3.4, рівнозначно діють і в логіці предикатів. Проте вони доповнюються правилами введення і видалення кванторів всезагальності і існування. Нехай E – множина формул.

Правила введення кванторів наступні:

для квантора $\forall$ $\dfrac{E \Rightarrow A(x)}{E \Rightarrow (\forall x)A(x)}$,

для квантора $\exists$ $\dfrac{E \Rightarrow A(x)}{E \Rightarrow (\exists x)A(x)}$.

Правила видалення кванторів наступні:

для квантора $\forall$ $\dfrac{E \Rightarrow (\forall x)A(x)}{E \Rightarrow A(x)}$,

для квантора $\exists$ $\dfrac{E \Rightarrow (\exists x)A(x)}{E \Rightarrow A(x)}$.

Так само, як і в логіці висловлювань, в логіці предикатів правила виведення застосовуються дуже рідко. Причини однакові й тому тут не повторюються.

В основному правила виведення використовуються для еквівалентних перетворень формульного представлення теорем з метою приведення їх до канонічної форми.

4.6. Правила еквівалентних перетворень формул обчислення предикатів.

У логіці першого порядку (логіці предикатів) умови ефективного застосування методу резолюцій для доказу теорем такі ж, як і в логіці висловлювань. Нагадуємо, що одна з цих умов – це представлення теорем в ПКНФ. Правила еквівалентних перетворень формул, введені в логіці висловлювань, рівнозначні і для логіки першого порядку. Однак присутність в формулах кванторів всезагальності та існування ускладнює застосування теорем до ПКНФ.

У зв'язку з цим додатково застосовується низка правил, що дозволяють виключити зазначені квантори з формул. Ці правила поділяються на дві групи:

1) Правила утворення випереджених (*рос*. – предварённых) нормальних форм (ВНФ);

2) Правила утворення скулемівських стандартних форм (ССФ). Розглянемо ці форми та правила їх утворення.

Формула F знаходиться у ***випередженій нормальній формі*** (ВНФ), тоді і лише тоді, коли вона має вигляд:

$(Q_1\ x_1)(Q_2 x_2)....(Q_n x_n)\ (M),$ де кожне $(Q_i x_i), i = \overline{1,n}$ це або $(\forall x_i)$, або $(\exists x_i)$, та (M) – це формула, яка не містить кванторів. $(Q_1\ x_1)(Q_2 x_2)....(Q_n x_n)$ називається префіксом, а (M) – матрицею формули F.

Наприклад, в формулі $(\forall x)(\forall y)(\exists z)(Q(x,y) \supset R(z))$

префікс $(\forall x)(\forall y)(\exists z)$ випереджує матрицю $(Q(x,y) \supset R(z)).$

Розглянемо правила еквівалентних перетворень формул, що містять квантори.

Нехай F це формула, що містить вільну змінну x .Будемо позначати цю формулу $F[x]$.. Нехай G – це формула, яка не містить змінної x. Нехай Q є квантором $\forall$ або квантором $\exists$. У цьому випадку правила наступні:

1a) $(Qx)F[x] \vee G \equiv (Qx)(F[x] \vee G)$;

1b) $(Qx)F[x] \wedge G \equiv (Qx)(F[x] \wedge G)$;

2a) $\neg((\forall x)P[x]) \equiv (\exists x)(\neg F[x])$;

2b) $\neg((\exists x)F[x]) \equiv (\forall x)(\neg F[x])$;

3a) $(\forall x)(F[x] \wedge (\forall x)H[x] \equiv (\forall x)(F[x] \wedge H[x])$;

3b) $(\forall x)(F[x] \vee (\forall x)H[x] \equiv (\forall x)(F[x] \vee H[x])$;

4a) $(Q_1x)F[x] \vee (Q_2x)H[x] \equiv (Q_1x)(Q_2z)(F[x] \vee H[z])$;

4b) $(Q_3x)F[x] \wedge (Q_4x)H[x] \equiv (Q_3x)(Q_4z)(F[x] \wedge H[z])$;

Використовуючи правила еквівалентних перетворень формул логіки висловлювань і вказані вісім правил, завжди можливо перетворити будь-яку формулу в ВНФ. Розглянемо приклад.

Приведемо формулу $(\forall x)P(x) \supset (\exists x)Q(x)$ до ВНФ. Використовуючи правило виключення зв'язки імплікації, отримаємо:

$(\forall x)P(x) \supset (\exists x)Q(x) \equiv \neg((\forall x)P(x) \vee (\exists x)Q(x)$.

За правилом 2а маємо:

$\neg((\forall x)P(x) \vee (\exists x)Q(x) \equiv (\exists x)(\neg P(x)) \vee (\exists x)Q(x)$.

Нарешті, використовуючи правило 3b, отримаємо:

$(\exists x)(\neg P(x)) \vee (\exists x)Q(x) \equiv (\exists x)(\neg P(x) \vee Q(x))$.

Формула в правій частині останнього співвідношення зображена у випередженій нормальній формі (ВНФ).

Скулеміська стандартна форма – це ВНФ, у префіксі якої відсутні квантори існування $\exists$, а матриця M є ПКНФ. З цього визначення стає очевидним, що скулемівське перетворення формул спрямовані на виключення кванторів існування з випереджених нормальних форм. Розглянемо ці правила перетворення.

Нехай формула F знаходиться у випередженій нормальній формі, $(Q_1 x_1)(Q_2 x_2)....(Q_n x_n)\,(M),$ де M є ПКНФ. Припустимо, що Q_r є квантором існування у префіксі $(Q_1 x_1)(Q_2 x_2)....(Q_n x_n)$, $1 \le r \le n$.

Якщо ніякий квантор загальності не знаходиться в префіксі лівіше Q_r, то виберемо нову константу c, що відрізняється від інших констант, що входять до M, замінимо всі x_r, що зустрічаються в M, на константу c та викреслимо $(Q_r x_r)$ із префікса. Якщо $Q_{s1}, Q_{s2}, ..., Q_{sm}$ – список усіх кванторів всезагальності, що зустрічаються лівіше Q_r, $1 \le s1 < s2 < ...sm < r,$ то виберемо новий $m-$місний функціональний символ f, який відрізняється від інших функціональних символів, замінимо всі x_r у M на $f(x_{s1}, x_{s2}, ..., x_{sm})$ і викреслимо $(Q_r x_r)$ із префікса.

Потім цей процес застосовуємо щодо всіх кванторів існування в префіксі; остання з отриманих формул є ССФ – скулемівська стандартна форма.

Константи та функції, які використовуються для заміни змінних квантора існування, називають скулемівськими константами і функціями. Розглянемо приклад.

Отримаємо ССФ для формули:

$(\exists x)(\forall y)(\forall z)(\exists u)(\forall v)(\exists w)P(x, y, z, u, v, w).$

Тут лівіше $(\exists x)$ немає ніяких кванторів загальності, лівіше $(\exists u)$ знаходяться $(\forall y)$ та $(\forall z)$, а лівіше $(\exists w)$ знаходяться $(\forall y)$, $(\forall z)$ та $(\forall v)$. Отже, замінимо змінну x на константу a, змінну u на двомісну функцію $f(y, z)$, змінну w на тримісну функцію $g(y, z, v)$. Таким чином, після вказаних замін і вилучення кванторів існування отримаємо наступну скулемівську стандартну форму для згаданої раніше початкової формули:

$(\forall y)(\forall z)(\forall v)P(a, y, z, f(y, z), v, g(y, z, v)).$

Розглянуті правила еквівалентних перетворень дають можливість перетворити будь-яку теорему логіки предикатів в скулемівській стандартній формі. Оскільки префікс в цій формі містить тільки квантори загальності, то це означає, наприклад, для $(\forall x)G,$ що форма набуває значення І, якщо G істинно для кожного x з області D (а в іншому випадку набуває значення Х (Хибно)), то це надає право розглядати G як просте висловлювання і квантор загальності, що зв'язує x, просто викреслити з префікса. Рівною мірою цей висновок відноситься і до кванторів всезагальності, що зв'язують інші змінні. Тому для доказу теорем в логіці предикатів можливо використовувати тільки матриці, що знаходяться в ПКНФ.

У логіці предикатів доказана також наступна теорема.

Нехай S – множина диз'юнктів, які зображують ПКНФ в скулемівській стандартній формі деякої формули (теореми) F. Тоді формула F суперечлива в тому і лише в тому випадку, коли множина S суперечлива.

Механізм застосування методу резолюцій, який використовувався для доказу теорем у логіці висловлювань, може бути застосований і в логіці предикатів. Однак при цьому виникають три істотних питання:

1) як знайти контрарні пари для диз'юнктів, що містять змінні?

2) як обчислити резольвенту з диз'юнктів, що містять змінні?

3) як отримати максимальну користь від зворотньої дедукції з метою підвищення ефективності методу резолюцій?

Відповіді на ці питання вносять деяку специфіку до алгоритму методу резолюцій.

4.7. Особливості використання метода резолюцій при доведенні теорем у логіці предикатів

Ідея Джона Алана Робінсона (1965 р) – автора метода резолюцій [4], – полягає в тому, щоб у логіці предикатів можна було, працювати, безпосередньо, з диз'юнктами, що містять змінні, не прибігаючи до попередньої заміни змінних константами з області визначення D. Ця ідея забезпечується використанням операцій ***підстановки*** та ***уніфікації***. Щоб визначити суть цих операцій, введемо кілька нових понять і визначень.

Поняття «***вираз***». Під виразом мається на увазі терм, безліч термів, безліч атомів, літера, диз'юнкт, безліч диз'юнктів. Коли у виразі не зустрічаються ніякі змінні, він називається ***основним виразом***: основний атом, основна літера, основний диз'юнкт, основний терм, що говорить про відсутність в них змінних.

Поняття «***підстановка***». Попередньо на прикладі з'ясуємо, в чому полягає проблема пошуку контрарних пар в логіці предикатів. Розглянемо наступні диз'юнкти:

$$C_1: \quad P(x) \vee Q(x)$$

$$C_2: \quad \neg P(f(x)) \vee R(x)$$

Слід відмітити, що в диз'юнкті C_1 поки немає ніякої літери, що була б контрарна якій-небудь літері в диз'юнкті C_2. Проте, підставивши у C_1 функцію $f(x)$ замість змінної x ,отримаємо:

$$C_1^*: \quad P(f(x)) \vee Q(f(x)).$$

Тепер літера $P(f(x))$ у C_1^* є контрарною літері $\neg P(f(x))$ в C_2.

Отже , можливо отримати резольвенту з C_1^* та C_2 :

$$C_3: \quad Q(f(x)) \vee R(x).$$

Визначення 1: підстановка – це кінцева множина виду $\{t_1 / v_1, \ldots\ldots, t_n / v_n\}$, де кожна v_i – це змінна, кожен елемент t_i – це терм, що відрізняється від v_i, а всі v_i – різні.

Визначення 2: Нехай $\Theta = \{t_1 / v_1, \ldots\ldots, t_n / v_n\}$ – підстановка, а E – вираз. Тоді $E\Theta$ – це вираз, отриманий з E заміною одночасно всіх входжень змінної v_i $(i = \overline{1,n})$ у виразі E на терм t_i. $E\Theta$ називається прикладом E.

Наприклад, нехай $\Theta = \{a / x, f(b) / y, c / z\}$ та $E = P(x, y, z).$ у цьому випадку підстановка дає $E\Theta = P(a, f(b), c).$

Поняття "***композиція підстановок***". Нехай дані підстановки $\Theta = \{t_1 / x_1,...,t_n / x_n\}$ та $\lambda = \{u_1 / y_1,...,u_m / y_m\}$. Тоді композиція Θ та λ є підстановкою $\Theta \circ \lambda$, яка виходить з множини $\{t_1\lambda / x_1,...,t_n\lambda / x_n, u_1 / y_1,...,u_m / y_m\}$ викреслюванням усіх елементів $t_j\lambda / x_j$, для яких $t_j\lambda = x_j$ й усіх елементів u_i / y_i таких, що $y_i \in \{x_1, x_2,...,x_n\}$.

Розглянемо приклад визначення композиції підстановок $\Theta = \{f(y) / x, z / y\}$ та $\lambda = \{a / x, b / y, y / z\}$.

Тоді $\Theta \circ \lambda = \{f(b) / x, y / y, a / x, b / y, y / z\}$. Однак, за визначенням y / y має бути викреслена; a / x та b / y також мають бути викреслені, оскільки x та y містяться серед змінних підстановки Θ. Як результат отримуємо: $\Theta \circ \lambda = \{f(x) / x, y / z\}$.

Поняття "***уніфікатор для множини виразів***". Підстановка Θ називається уніфікатором для множини виразів $\{E_1, E_2,...,E_n\}$ тоді й лише тоді, коли $E_1\Theta \equiv E_2\Theta \equiv ... \equiv E_n\Theta$. Вважається, що безліч виразів уніфікується, якщо для нього існує уніфікатор.

Уніфікатор σ для множини виразів $\{E_1, E_2,...,E_n\}$ називається найбільш загальним уніфікатором (НЗУ) тоді й лише тоді, коли для кожного уніфікатора Θ для цієї множини існує така підстановка λ, що $\Theta = \sigma \circ \lambda$. Нехай, наприклад, існує множина виразів $W = \{P(x, a, f(g(a))), P(z, y, fu))\}$. Тоді $\sigma = \{z / x, a / y, g(a) / u\}$ є найбільш загальний уніфікатор для W, а $\Theta = \{b / x, a / y, b / z, g(a) / u\}$ є найбільш загальним уніфікатором.

Опишемо по кроках ***алгоритм уніфікації***, який знаходить найбільш загальний уніфікатор (НЗУ), якщо множина $E = \{E_1, E_2,...,E_k\}$ уніфікується, і повідомляє про невдачу, якщо це не так [7]:

1) Встановити $k = 0, E_k = E$ і $\sigma_k = \varepsilon$, де ε – це порожня підстановка, що не містить елементів. Перейти до кроку 2.

2) Якщо E_k не є одноелементною множиною, то перейти до кроку 3. В інакшому випадку покласти $\sigma = \sigma_k$ и закінчити роботу.

3) Кожна з літер у E_k розглядається як ланцюг символів і виділяються перші підвирази літер, які не являться однаковими у всіх елементів E_k, тобто утворюється так звана множина неузгодженостей (рос. – рассогласований) типу $\{x_k, t_k\}$. Якщо у цій множині x_k – змінна, а t_k – терм, відмінний від x_k, то перейти до кроку 4. В противному випадку закінчити роботу з висновком: E не уніфіковано.

4) Нехай $\sigma_{k+1} = \sigma_k \circ \{t_k / x_k\}$ і $E_{k+1} = E_k\{t_k / x_k\}$.

5) Встановити $k := k+1$ и перейти до кроку 2.

Розглянемо роботу алгоритму знаходження НЗУ для множини $E = \{P(y, g(z), f(x)), P(a, x, f(d(y)))\}$. Кроки наступні:

1) $\sigma_0 = \varepsilon$ и $E_0 = E$.

2) Так як E_0 не є одноелементною множиною, то перейти до кроку 3.

3) Множина неузгодженостей $\{y, a\}$, тобто заміна $\{a / y\}$.

4) $\sigma_1 = \sigma_0 \circ \{a / y\} = \varepsilon \circ \{a / y\} = \{a / y\}$.
$E_1 = E_0\{a / y\} = \{P(a, g(z), f(x)), P(a, x, f(g(a)))\}$

5) Так як множина E_1 знову не одноелементна, то множина неузгодженостей буде $\{g(z), x\}$, тобто заміна $\{g(z) / x\}$.

6) $\sigma_2 = \sigma_1 \circ \{g(z) / x\} = \{a / y, g(z) / x\}$.
$E_2 = E_1\{g(z) / x\} = \{P(a, g(z), f(g(z))), P(a, g(z), f(g(a)))\}$.

7) Маємо $\{z, a\}$, тобто $\{a / z\}$.

8) $\sigma_3 = \sigma_2 \circ \{a / z\} = \{a / y, g(a) / x, a / z\}$
$E_3 = E_2\{a / z\} = \{P(a, g(a), f(g(a))), P(a, g(a), f(g(a)))\} =$
$= \{P(a, g(a), f(g(a)))\}$.

Так як отримана одноелементна множина, то шуканий найбільш загальний уніфікатор $\sigma = \sigma_3 = \{a / y, g(a) / x, a / z\}$.

Можна показати, що алгоритм уніфікації завжди завершується на кроці 2, якщо множина E уніфікована, і на кроці 3 – в інакшому випадку.

Поняття «***склейка диз'юнкта*** C». Якщо дві або більше літер (з однаковим знаком) диз'юнкта C мають найбільш загальний уніфікатор σ, то $C\sigma$ зветься склейкою C. Якщо $C\sigma$ – одиничний диз'юнкт, то склейка зветься одиничною склейкою.

Приклад: Нехай $C = P(x) \vee P(f(y)) \vee \neg Q(f(y))$.

Тоді перша і друга літери мають найбільш загальний уніфікатор $\sigma = \{f(y) / x\}$. Отже, $C\sigma = P(f(y)) \vee \neg Q(F(y))$ є склейка C.

Поняття «***бінарна резольвента***». Нехай C_1 и C_2 - два диз'юнкти (названі диз'юнктами-посилками), які не мають ніяких загальних змінних. Нехай також l_1 и l_2 - дві літери відповідно у C_1 і C_2. Якщо l_1 і $\neg l_2$ мають найбільш загальний уніфікатор σ, то диз'юнкт $(C_1 - l_1\sigma) \vee (C_2 - l_2\sigma)$ зветься бінарною резольвентою C_1 и C_2, а літери l_1 і l_2 називаються відрізаними літерами.

Приклад: Нехай $C_1 = P(x) \vee Q(x)$ і $\neg C_2 = P(a) \vee R(x)$. Так як змінна x входить в C_1 і C_2, то замінимо змінну x в C_2 на y, тобто $\neg$

$C_2 = P(a) \vee R(y)$. Обираємо $l_1 = P(x)$ і $l_2 = \neg P(a)$ Так як $l_2 = \neg P(a)$, то l_1 і l_2 мають найбільш загальний уніфікатор $\sigma = \{a / x\}$.

Отже,

$$(C_1\sigma - l_1\sigma) \vee (C_2\sigma - l_2\sigma) \equiv (\{P(a), Q(a)\} - \{P(a)\}) \vee (\{\neg P(a), R(y)\} -$$

$$\{\neg P(a)\}) \equiv \{Q(a)\} \vee \{R(y)\} = \{Q(a), R(y)\} = Q(a) \vee R(y).$$

Таким чином, $Q(a) \vee R(y)$– бінарна резольвента C_1 і C_2, а $P(x)$ і $\neg P(a)$ – відрізаємі літери.

Поняття «***резольвента логіки першого порядку***». Резольвентою диз'юнктів-посилок C_1 і C_2 є одна із наступних резольвент:

1) бінарна резольвента C_1 і C_2;
2) бінарна резольвента C_1 і склейки C_2;
3) бінарна резольвента C_2 і склейки C_1;
4) бінарна резольвента склейки C_1 і склейки C_2.

Приклад: Нехай $C_1 = P(f(g(a))) \vee \neg R(b)$,

$C_2 = P(x) \vee \neg P(f(y)) \vee \neg Q(y)$.

Тоді склейкою диз'юнкта C_2 є $C_2\sigma = C_2^{'} = \neg P(f(y)) \vee Q(y)$ і резольвентою для C_1 і $C_2^{'}$ буде $C = \neg R(b) \vee Q(g(a))$.

4.8. Доведення теорем методом резолюцій в логіці предикатів

Не існує чітких правил и рекомендацій, як представити у вигляді формули логіки предикатів той чи інший умовивід. Все це робиться інтуїтивно. А інтуїція знаходиться у пропорціональній залежності від майстерності у доведенні теорем методом резолюцій. Тому тільки через практику і навички можна придбати необхідну інтуїцію для моделювання умовиводів.

У якійсь мірі стартову інтуїцію можна придбати, ознайомившись з приведеними нижче прикладами, розглянутими у різних роботах [5,7,8,12,13]. У перших двох прикладах приводяться достатньо детальні пояснення, в інших – тільки по мірі необхідності.

Приклад 1. Деякі пацієнти люблять своїх лікарів. Ні один пацієнт не любить знахаря. Отже, жоден лікар не являється знахарем.

Введемо слідуючи позначення для предикатних символів: P – пацієнт, D – лікар, Z – знахар, L – любить.

Тоді перераховані нижче предикати будуть позначати:

$P(x)$ — x є пацієнт;

$D(x)$ — x є лікар;

$Z(x)$ — x є знахар;

$L(x,y)$ — x любить y.

Факти і висновок, наведені у розсудах, можуть бути представлені слідуючими формулами:

Факт 1 F_1 : $(\exists x)(P(x) \wedge (\forall y)(D(y) \supset L(x,y)))$.

Факт 2 F_2 : $(\forall x)(P(x) \supset (\forall y)(Z(y) \supset \neg L(x,y)))$.

Висновок G : $(\forall x)(D(x) \supset \neg Z(x))$.

Відповідно до умов ефективного доведення теорем методом резолюцій перетворимо факти F_1, F_2 і заперечення заключення $\neg G$ за правилами еквівалентних перетворень в наступні диз'юнкт:

(1) $P(a)$

(2) $D(y) \vee \neg L(a,y)$

} з F_1 Тут діє правило виключення квантору існування і правило виключення зв'язки імплікації.

(3) $\neg P(x) \vee \neg Z(y) \vee \neg L(x,y)$ из F_2

(4) $D(b)$

(5) $Z(b)$

} з $\neg G$ Тут діє правило 2а з 4.6., правило виключення квантору існування і правило виключення зв'язки імплікації.

Виконуючи уніфікації і склейки, отримаємо:

(6) $L(a,b)$ резольвента (2) і (4).

(7) $\neg Z(y) \vee \neg L(a,y)$ резольвента (1) і (3).

(8) $\neg L(a,b)$ резольвента (5) і (7).

(9) $\square$ резольвента (6) і (8).

Теорема доведена.

Приклад 2. Всі люди – тварини. Отже, голова людини є головою тварини.

Нехай є наступні предикати:

$L(x)$ — x є людина;

$A(x)$ — x є тварина;

$G(x,y)$ — x є головою y.

Необхідно довести теорему:

$$\frac{(\forall y)(L(y) \supset A(y))}{(\forall x)((\exists y)(L(y) \wedge G(x,y)) \supset (\exists z)(A(z) \wedge G(x,z)))}$$

Перетворення чисельника (теореми-посилки) дає диз'юнкт:

(1) $L(y) \vee A(y)$.

Для отримання інших диз'юнктів претворимо заперечення знаменника (теореми-заключення) наступним чином:

$$\neg (\forall x)((\exists y)(L(y) \wedge G(x,y)) \supset (\exists z)(A(z) \wedge G(x,z))) \equiv$$
$$\equiv \neg(\forall x)((\exists y)(L(y) \wedge G(x,y)) \vee (\exists z)(A(z) \wedge G(x,z))) \equiv$$
$$\equiv \neg(\forall x)((\forall y)(\neg L(y) \vee G(x,y)) \vee (\exists z)(A(z) \wedge G(x,z))) \equiv$$
$$\equiv \neg (\forall x)(\forall y)(\exists y)(\neg L(y) \vee \neg G(x,y) \vee A(z) \wedge G(x,z)) \equiv$$
$$\equiv (\exists x)(\exists y)(\forall z)(L(y) \wedge G(x,y) \wedge \neg A(z) \vee \neg G(x,y)).$$

Тоді (2) $L(b)$.

(3) $G(a,b)$.

(4) $A(z) \vee G(a,z)$.

Застосовуючи метод резолюцій, отримаємо:

(5) $L(b)$ з (1) і (2).

(6) $G(a,b)$ з (4) і (5).

(7) $\square$ из (3) і (6).

Теорема доведена.

Приклад 3. Посилки: митні чиновники обшукують кожного, хто в'їжджає в країну, окрім високопоставлених осіб. Деякі особи, сприяючі провозу наркотиків, в'їжджають в країну і обшукуються виключно людьми, які також сприяють провозу наркотиків. Ніхто з високопоставлених осіб не сприяв провозу наркотиків.

Висновок: деякі з митників сприяли провозу наркотиків.

Введемо наступні позначення для предикатів:

$E(x)$: x в'їжджав в країну;

$V(x)$: x був високопоставленою особою;

$S(x,y)$: y обшукував x ;

$C(x)$: x був митником;

$P(x)$: x сприяв провозу наркотиків.

Посилки представляються наступними формулами:

$(\forall x)(E(x) \wedge \neg V(x) \supset (\exists y)(S(x,y) \wedge C(y)))$,

$(\exists x)(P(x) \wedge E(x) \wedge (\forall y)(S(x,y) \supset P(y))$,

$(\forall x)(P(x) \supset \neg V(x)$,

а висновок теореми – формулою:

$(\exists x)(P(x) \wedge C(x))$.

Перетворюючи посилки в диз'юнкти, отримаємо:

(1) $\neg E(x) \vee V(x) \vee S(x, f(x))$.

(2) $\neg E(x) \vee V(x) \vee C(f(x))$.

(3) $P(a)$.

(4) $E(a)$.

(5) $\neg S(a, y) \vee P(y)$.

(6) $\neg P(x) \vee \quad V(x)$.

Заперечення заключення:

(7) $\neg P(x) \vee \neg C(x)$.

Доведення методом резолюцій виглядає наступим чином:

(8) $\neg V(a)$ із (3) і (6).

(9) $V(a) \vee C(f(a))$ із (2) і (4).

(10) $C(f(a))$ із (8) і (9).

(11) $V(a) \vee S(a, f(a))$ із (1) і (4).

(12) $S(a, f(a))$ із (8) і (11).

(13) $P(f(a))$ із (12) і (5).

(14) $\neg C(f(a))$ із (7) і (13).

(15) $\square$ із (10) і (14).

Висновок доведено.

Приклад 4. Існують студенти, які люблять всіх викладачів. Жоден зі студентів не любить невежд (невігласів). Отже, жоден з викладачів не являється невеждою.

Позначимо:

$C(x)$ — x є студент;

$P(x)$ — x є викладач;

$H(x)$ — x є невеждою;

$L(x, y)$ — x любить y.

На мові логіки предикатів після приведения до стандартного виду це запишеться так:

$$(\exists x)(C(x) \wedge (\forall y)(P(y) \supset L(x, y)))$$

$$\underline{(\forall x)(C(x) \supset (\forall y)(H(y) \supset \neg L(x, y)))}$$

$$(\forall y)(P(y) \supset \neg H(y))$$

Претворення двох теорем-посилок чисельника дає наступні диз'юнкти:

(1) $C(a)$.

(2) $\neg P(y) \vee L(a, y)$.

(3) $\neg C(x) \vee \neg H(y) \vee \neg L(x, y)$.

Після перетворення заперечення заключення із знаменника отримаємо:

$\neg(\forall y)(\neg P(y) \vee \neg H(y) \equiv (\exists y)(P(y) \wedge H(y))$, що дає диз'юнкти:

(4) $P(b)$.

(5) $H(b)$.

Шляхом уніфікації і склейок отримаємо:

(6) $L(a, b)$ із (2) і (4).

(7) $\neg H(y) \vee \neg L(a,y)$ із (1) і (3).

(8) $\neg L(a,b)$ із (5) і (7).

(9) $\square$ із (6) і (8).

Теорема доведена.

Приклад 5. Задача про мавпу і банан.

Мавпа хоче з'їсти банан, що підвішений до стелі кімнати. Зріст мавпи недостатній, щоб вона могла дотягнутись до банану. Однак вона може ходити по кімнаті, переносить стілець, який знаходиться у тій же кімнаті, може забратись на стілець и дістати банан. Показати порядок дій мавпи, при якому вона дістане банан.

Предикати тут такі:

$P(x,y,z,s)$ означає, що у стані s мавпа знаходиться у точці x, банан – у точці y, а стілець – у точці z;

$R(s)$ означає, що у стані s мавпа може дістати банан.

Функції, які беруть участь в описі задачі, наступні:

ходити(y,z,s)– стан, який створюється, якщо мавпа знаходилась спочатку у стані s і перейшла з точки x в точку z;

носити(y,z,s)– стан, який створюється, якщо мавпа знаходилась спочатку у стані s і перейшла з точки y в точку z, несучи з собою стілець;

підніматися(s)– стан, який створюється, якщо мавпа знаходилась у стані s і влізла на стілець.

Припускаємо, що спочатку мавпа знаходилась у точці a, банан – у точці b, стілець – у точці c і мавпа була у стані s_1.

Таким чином, маємо наступні аксіоми:

(1) $\neg P(x,y,z,s) \vee P(z,y,z,$**ходити**$(x,z,s))$.

(2) $\neg P(x,y,x,s) \vee P(y,y,y,$**носити**$(x,y,s))$.

(3) $\neg P(b,b,b,s) \vee R($**підніматися**$(s))$.

(4) $P(a,b,c,s_1)$.

Тут диз'юнкт (1) означає, що у будь-якому стані мавпа може перейти з точки x в точку z.

Диз'юнкт (2) означає, що якщо мавпа знаходиться біля стільця, який стоїть у точці x, то вона може перенести його у будь-яку точку y.

Диз'юнкт (3) означає, що якщо стілець і мавпа знаходяться під бананом, то мавпа може влізти на стілець і дістати банан.

Диз'юнкт (4) описує вихідну ситуацію.

Висновку теореми відповідає диз'юнкт

(5) $\neg R(s) \vee$ **відповідь**(s).

У цьому диз'юнкті предикат **відповідь** вимагає встановити порядок дій мавпи, відповідний стану мавпи з бананом.

Використовуючи диз'юнкти (1) — (5), виводимо наступні резольвенти:

(6) $\neg P(b,b,b,s) \vee$ **відповідь** (**підніматися** (s)) з (5) и (3).

(7) $\neg P(x,b,x,s) \vee$ **відповідь** (**підніматися** (**носити** (x,b,s)) з (6) и (2).

(8) $\neg P(x,b,z,s) \vee$ **відповідь** (**підніматися** (**носити** $z,b,$ **ходити** (x,z,s)))) з (7) и (1).

(9) **відповідь** (**підніматися** (**носити** $(c,b,$ **ходити** (a,c,s_1)))) з (8) и (4).

Диз'юнкт (9) дає відповідь. Його можна інтерпретувати як виконання наступних дій (починаючи з самої внутрішньої функції в диз'юнкті (9) і рухаючись назовні):

1. Мавпа йде з точки a в точку c.
2. Мавпа йде з точки c в точку b, несучи з собою стілець.
3. Мавпа влізає на стілець.

Після цих дій мавпа дістає банан.

4.9. Реалізація методу резолюцій у мові Пролог

Найвища ступінь розуміння алгоритму доказу теореми може бути досягнута у тому випадку, коли опис кожного кроку алгоритму ілюструється відповідними діями на конкретному прикладі, повнота якого дозволяє відобразити всі його особливості, реалізовані на деякій формальній мові. До таких мов відноситься декларативна мова Пролог [10]. Слід відмітити, що Пролог (*англ.* – Prolog) – є мовою і системою логічного програмування, що основана на мові предикатів математичної логіки диз'юнктів Хорна, який представляє собою підмножину логіки предикатів першого порядку

Розглянемо для початку роботу алгоритму доказу теорем методом резолюцій на прикладі відомої задачі про смертність Сократа [14].

У якості машинної реалізації використаємо одну з версій мови Пролог [11, 15,16], у якій допускається запис предикатів на російській мові.

Текст пролог-програми такий:

```
Domains                          % Розділ доменів
    имя=symbol
predicates                       % Розділ предикатів
    смертен(ім'я)
clauses                          % Розділ фактів і правил
    человек(сократ).             %факт про людину по імені Сократ
                                 % сократ – це константа
    смертен(X):—человек(X).      % правило «кожна людина смертна»
```

У програмі після знаку % до кінця рядку записується коментар. Слід також звернути увагу на обов'язкову наявність точки у кінці кожного факту і правила. Нарешті, декілька слів про правила у пролог-програмах. Кожне правило складається з голови (ліва половина правила), у якій поміщається визначений предикат, і тіла (права половина правила), у якому поміщається умовна частина правила.

Голова і тіло правила у програмі пов'язані знаком(—), який визначає істинність затвердження при виконанні умови. Цей знак в програмі можна замінити англійським словом **if** (якщо).

У правилах кома (,) відповідає знаку логічної операції І (логічне множення), а точка з комою (;) – знаку АБО (логічне додавання). Формування речень у мові Пролог обмежується тільки виразами Хорна, і тільки до них застосовують метод резолюцій. Слід нагадати, що вираз Хорна має вигляд:

$$\neg D_1 \vee \neg D_2 \vee ... \vee \neg D_n \vee D_m \ ,$$

де D_i – літерали, D_m - висновок теореми

Використовуючи зв'язку імплікації $\supset$, теорему можна переписати так:

$$D_1 \wedge D_2 \wedge ... \wedge D_n \supset D_m \ ,$$

що в синтаксису системи Пролог записується у вигляді:

$$D_m :- D_1, D_2, ..., D_n.$$

У приведеній вище Пролог-програмі представлені тільки посилки (посилки-факти і посилки-правила) і поки відсутні висновки теореми. Висновки теореми формулюються шляхом вказівки деякої мети.

Після введення програми у пролог-систему їй можна задавати питання, які стосуються описаних у програмі відношень. Поставлене системі питання є для неї метою, для досягнення якої система використовує інформацію розділу **clauses.**

Зазвичай система пропонує ввести мету з клавіатури одразу після запрошення, наприклад, у вигляді **? —** (або **Цель:**). У деяких пролог-системах це приклад так званої зовнішньої мети.

Питання про те, чи смертний Сократ, вводиться як заключення теореми:

? - смертен (сократ).

На це питання пролог-система відповість **yes** (Так).

Зазвичай, у пролог-системах з внутрішньої метою в програмі достатньо записати

goal смертен(сократ).

Розглянемо, як система отримує цю відповідь. Якщо висновок теореми не зможе використовувати тільки посилки-факти, то алгоритм прямує до посилок-правил (диз'юнктам Хорна) і підміняє поточні цілі новими до тих пір, поки ці нові цілі не виявляться простими посилками-фактами. Так як факт **смертен(сократ)** відсутній, то перше (і єдине) відповідне правило – це

смертен(X):— человек(X).

Це правило реалізує відношення **смертен**. Предикатний символ правила можна порівняти з предикатним символом заключення теореми.

Так як висновок **смертен(сократ**), то значення змінної **X** має бути приписано наступним чином:

X=сократ

Відбулася уніфікація {сократ / X} предиката **смертен(X)** правила і **смертен(сократ)** заключення теореми. Тоді початкова мета **смертен (сократ)** замінюється новою метою **человек(сократ)**, так як цей предикат є резольвентою вихідного укладання теореми і голови правила після їх уніфікації. Така мета негайно досягається, оскільки зустрічається в посилках теореми в якості факту. Цей крок завершує обчислення, так як отримана нова резольвента – порожній диз'юнкт склейкою заперечення укладання теореми (як того вимагає метод резолюцій) і поточної мети.

Всі кроки досягнення мети **смертен(сократ)** представлені на рис. 4.1 у вигляді дерева логічного висновку. Вершини дерева відповідають цілям або списками цілей, які потрібно досягти.

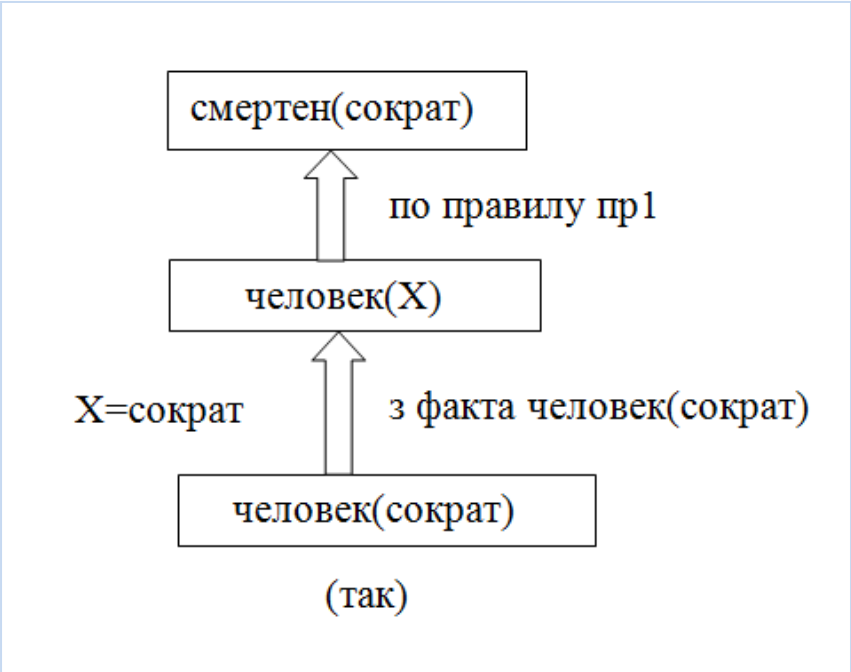

Рис. 4.1. Дерево логічного виводу

Дуги між вершинами відповідають застосуванню альтернативних пропозицій програми, які перетворять ціль, що відповідає одній вершині, в ціль, що відповідає іншій вершині. Коренева (верхня) ціль досягається тоді, коли знаходиться шлях від кореня дерева (верхньої вершини) до листа, поміченого міткою «так». Лист позначається міткою «так», якщо він є посилкою-фактом.

У другому прикладі розглянемо завдання про родинні стосунки [8, 9]. Рисунок 4.2 відображає стосунки батьків і дітей в деякій сім'ї, причому з тієї причини, що в мові Пролог константи розпочинаються з рядкової букви, усі імена на малюнку так і представлені.

Той факт, що Том є батьком Боба, можна записати на Пролозі так: **родитель(том,боб).**

Тут ми вибрали **родитель** в якості відношення (предиката), а **том** і **боб** є аргументами цього відношення. У пролог-програмі в розділі **domains** аргументи предиката **родитель** і нижче описаного предиката **предок** описані

як атоми символьного типу (**symbol**). У розділі **clauses** програми безпосередньо по дереву родинних стосунків записані усі наявні факти. Тут же записано два правила для визначення предиката **предок**, пояснення яких будуть дані пізніше.

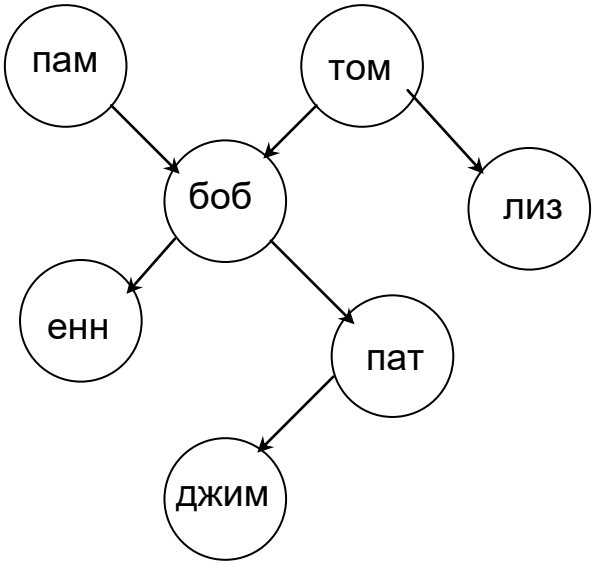

Рис. 4.2. Дерево родинних стосунків.

Текст Пролог-програми:

```
domains                          % Розділ доменов
имя=symbol
predicates                       % Розділ предикатів
родитель(имя,имя)
    предок(имя,имя)
clauses                          % Розділ фактів и правил
родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,енн).
родитель(боб,пат).
родитель(пат,джим).
предок(X,Z) : —                  %Правило пр1 : X– предок нащадка Z
      родитель(X,Z).
предок(X,Z) : —                  %Правило пр2 : X – предок нащадка Z
      родитель(X,Y),
      предок(Y,Z).
```

Якщо в завданні деяке відношення можна визначити декількома способами, то правил також буде декілька. Так, в цій програмі відношення **предок** визначено двома правилами. Перше правило визначає безпосередніх (найближчих) предків, а друге – віддалених. Деякий X є предком деякого Z, якщо між X і Z існує ланцюжок людей, пов'язаних між собою відношенням батько-дитина, як показано на рис. 4.3. У нашому прикладі на рис. 4.2 Том - найближчий предок Лиз і віддалений предок Джима.

Перше правило легко будується з рис. 4.3а:
Для всіх X и Z
X – предок Z, якщо
X – родитель Z ,
що відповідає в програмі запису :
предок(X,Z) : — родитель(X,Z).

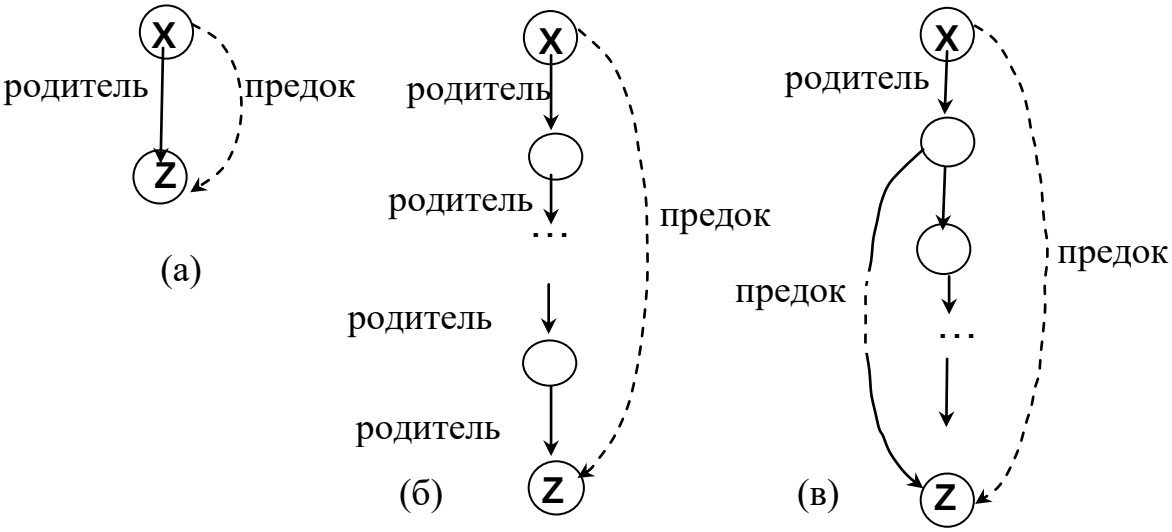

Рис. 4.3. Приклад відношення **предок**:

Друге правило для предиката **предок** дещо складніше, оскільки побудова ланцюжка стосунків **родитель** (див. рис. 4.3б) може викликати деякі труднощі. Якщо довжина ланцюжка строго фіксована, то можна побудувати правило, записавши в тілі через коми усю послідовність стосунків, наприклад, так:

(а) X – найближчий **предок** Z; (б) X – віддалений **предок** Z;
(в) рекурсивне формулювання відношення **предок**.

предок(X,Z) : — родитель(X,Y_1),
родитель(Y_1,Y_2),
родитель(Y_2,Y_3),
родитель(Y_3,Z).

Таке правило можна буде використати тільки для цього ланцюжка і не більше. Існує, проте, коректне і елегантне формулювання відношення **предок**, яка працюватиме для предків довільної віддаленості. Ключова ідея тут, – визначити відношення предок через нього самого. Таке визначення називається ***рекурсивним***. Рис. 4.3в ілюструє цю ідею.

Для всіх X и Y,
X – предок Z, якщо
існує Y такий, що

(1) X – родитель Y и
(2) Y – предок Z .

Пропозиція Прологу, що має той же сенс, записується так:

предок(X,Z) : —
родитель(X,Y),
предок(Y,Z).

Питання про те, чи являється Боб батьком Пат, має бути введений так:

? — **родитель(боб, пат).**

Знайшовши відповідний факт в програмі, система відповість:

Так

На питання

? **— родитель (лиз, пат)** система відповість:

Ні,

оскільки в програмі нічого не говориться про те, чи являється Лиз батьком Пат.

Як вже вказувалося, мета для програми може бути внутрішньою. Тоді вона вказується в розділі мети (**goal**), який записується перед (або після) розділу **clauses** і обов'язково завершується точкою. Наприклад, останнє питання системі з внутрішньою метою запишеться в програмі так:

goal
родитель(лиз,пат).

Система здатна відповісти і на цікавіші питання. Наприклад, «хто є батьком Лиз?»:

? — **родитель(X,лиз).**

На це питання система не просто відповість «так» або «ні». Вона скаже нам, яким має бути значення X (раніше невідоме), щоб наведене вище твердження було істинним, а також вкаже кількість рішень. Тому ми отримаємо відповідь:

X=том 1 решен.

Питання «Хто діти Боба?» можна передати системі в такій формі:

? — **родитель(боб, X).**

У цьому випадку буде отримана відповідь:

X = енн
X = пат 2 вирішене.

Програмі можна задавати і більш складні питання, скажімо, «Хто є батьком батька Джима?». Оскільки в програмі немає предиката типу **родитель_родитель**, питання можна скласти з двох простих запитань, використовуючи зв'язку I (Істина), наприклад, так:

? — **родитель(X, Y), родитель(Y, джим)**

Відповідь буде: **X = боб**
Y = пат 1 решен.

Це приклад складеного запиту.

Розглянемо роботу алгоритму доведення теорем в пролог-системі, якщо, наприклад, потрібно спробувати досягти мети:

? — **предок(тому, пат).**

Спочатку алгоритм пробує знайти таку посилку в теоремі, з якої негайно слідує згаданий висновок. Очевидно, єдиними підходящими для цього посилками-правилами є правила пр1 і пр2. Ці два правила реалізують відношення **предок**. Предикатні символи цих правил можна порівняти з предикатним символом заключення теореми. Спочатку алгоритм пробує посилку-правило пр1, що стоїть в програмі першим:

предок (X, Z): — родитель(X, Z). % пр1

Оскільки заключення – **предок(том, пат)**, то значення змінним повинні бути приписані наступним чином:

X = том, **Z** = пат

Відбулася уніфікація {том / X, пат / Y} предикатів **предок**(X, Z) правила пр1 і **предок** (**том**,**пат**) заключення теореми. Тоді вихідна мета предок (**том**, **пат**) замінюється новою метою **родитель**(**том**,**пат**), так як цей предикат є резольвентою вихідного укладання теореми і голови правила пр1 після їх уніфікації.

У посилках теореми немає правила, предикатний символ якого можна було б порівняти з новою метою **родитель**(**том**,**пат**), тому така мета виявляється неуспішною. Це говорить про те, що резольвенту – порожній диз'юнкт (□) вивести з початкової множини диз'юнктів при цьому варіанті міркування не можна. Тепер алгоритм робить повернення до вихідної мети, тобто до заключення теореми, щоб спробувати другий варіант виведення мети верхнього рівня **предок(том,пат)**. Таким чином, пробується правило пр2

предок (X, Z): -
родитель(X, Y),
предок(Y, Z).

Як і раніше, змінним приписуються значення **X**=том, **Z**=пат. У цей момент змінній **Y** ще не приписано ніякого значення. Вихідна мета **предок(том,пат)** замінюється двома цілями **родитель(том,Y)** і **предок(Y, пат)**, які є резольвентою пр2 і вихідної мети – заключення теореми, отриманої після уніфікації предикатів, що співставляються.

Маючи тепер перед собою дві мети, алгоритм намагається досягти їх в тому порядку, в якому вони записані. Досягти першої з них легко, оскільки вона відповідає факту з посилок. Провівши уніфікацію предиката **родитель (том,Y)** і предиката **родитель(том,боб)**, який стоїть в посилках першим, алгоритм змінної **Y** приписує значення **боб**. Тим самим досягається перша мета, а друга перетворюється на мету **предок(боб,пат).**

Для досягнення цієї мети знову застосовується правило пр1. Зауважимо, що це (друге) застосування правила пр1 ніяк не пов'язане з його першим застосуванням. Тому алгоритм використовує нову множину змінних правила щоразу, як воно застосовується. Щоб вказати це, ми перейменуємо змінні правила пр1 для нового його застосування в такий спосіб:

предок (X′, Z′): — родитель(X′, Z′).

Голова цього правила збігається з предикативним символом поточної мети **предок(боб,пат)**, тому виконується уніфікація {**боб / X′**, **пат / Z′**).

Поточна мета замінюється метою **родитель(боб,пат)** – тобто на нову резольвенту. Така мета негайно досягається, оскільки зустрічається в посилках теореми в якості факту. Цей крок завершує обчислення, так як отримана нова резольвента - порожній диз'юнкт склеюванням заперечення укладання теореми (як того вимагає метод резолюцій) і поточної мети.

Всі кроки досягнення мети **предок(том,пат)** представлені на рис. 4.4 у вигляді дерева логічного висновку. Як вже зазначалося при розгляді мал. 4.1, вершини дерева відповідають цілям або спискам цілей, які потрібно досягти.

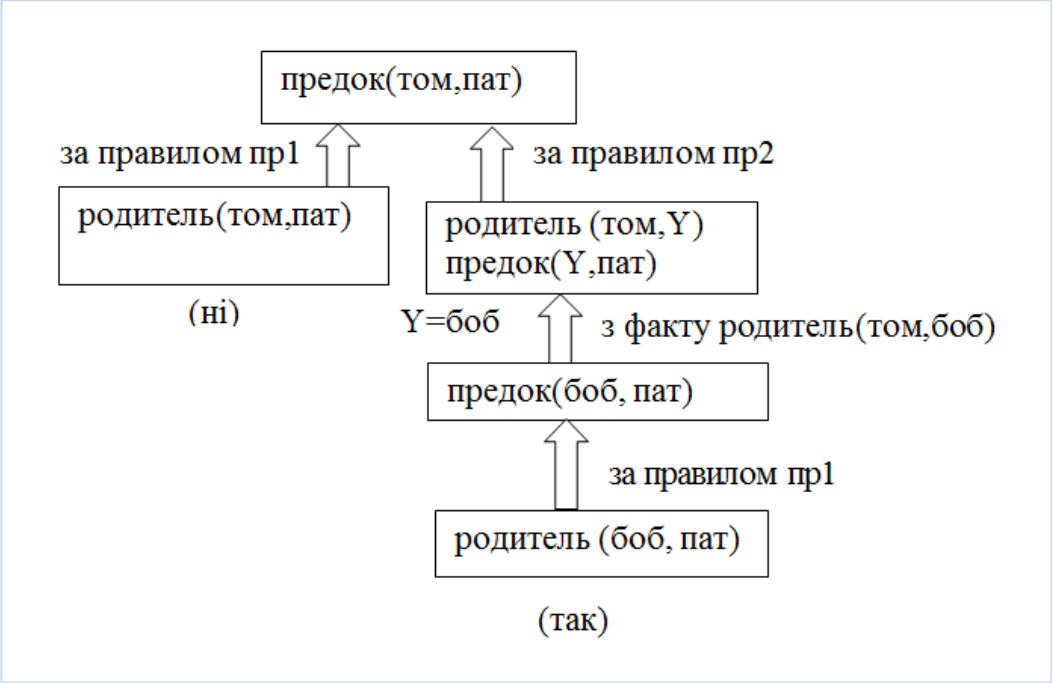

Рис. 4.4. Дерево логічного виводу на запит предок(том,пат)

Коренева (верхня) мета досягається тоді, коли знаходиться шлях від кореня дерева (верхньої вершини) до листа, позначеного міткою «так». Лист позначений міткою «так», якщо він є посилкою-фактом.

Виконання пролог-програми полягає в пошуку таких шляхів. У процесі такого пошуку система може входити і в гілки, що призводять до неуспіху. У той момент, коли вона виявляє, що гілка не приводить до успіху, відбувається повернення до попередньої вершини (в пролог-системі це **backtrack**, відкат).

Далі йде спроба застосувати до неї альтернативне рішення. Очевидно, що описаний алгоритм повністю формалізований, може виконуватися автоматично без участі людини. Отже, на запитання:

? — **предок(том, пат)**

пролог-система видає відповідь

Так.

5. Нечіткі знання

При спробах формалізувати людські знання дослідники незабаром зіткнулися з проблемою, що ускладнювала використання традиційного математичного апарату для їх опису. Існує цілий клас описів, що оперують якісними характеристиками об'єктів (багато, мало, сильний, дуже сильний і т.д.). Ці характеристики зазвичай розмиті і не можуть бути однозначно інтерпретовані, проте містять важливу інформацію (наприклад, «Однією з можливих ознак грипу є висока температура»).

Крім того, в задачах, розв'язуваних інтелектуальними системами, часто доводиться користуватися неточними знаннями, які не можуть бути інтерпретовані як повністю правдиві (істинні) чи неправдиві (хибні) (відповідають логічним значенням true / false або 0 / 1). Звісно, що існують знання, достовірність яких виражається деякою проміжної цифрою, наприклад 0.7.

Як, не руйнуючи властивості розмитості і неточності, представляти подібні знання формально? Для вирішення таких проблем американський математик Лотфі Заде запропонував формальний апарат нечіткої (fuzzy) алгебри і нечіткої логіки [17].

На даний час, нечітка логіка (англ. *Fuzzy logic*) – це розділ математики, що є узагальненням класичної логіки і теорії множин, який базується на понятті нечіткої безлічі, як об'єкта з функцією приналежності елемента до безлічі, що приймає будь-які значення в інтервалі [0, 1]. На основі цього поняття вводяться різні логічні операції над нечіткими множинами і формулюється поняття *лінгвістичної змінної*, як значення якої виступають нечіткі множини.

Пізніше цей напрямок одержав широке поширення [18, 19] і поклав початок однієї з гілок ШІ під назвою – м'які обчислення (soft computing).

Поняття *лінгвістичної змінної* ввів Л. Заде, як одне з головних понять в нечіткій логіці.

Лінгвістична змінна (ЛЗ) – це змінна, значення якої визначається набором вербальних (тобто словесних) характеристик деякої властивості. Наприклад, ЛЗ «зріст» визначається через набір {карликовий, низький, середній, високий, дуже високий}.

Нечіткі системи засновані на правилах продукційного типу, проте в якості посилки і укладання в правилі використовуються лінгвістичні змінні, що дозволяє уникнути обмежень, властивих класичним продукційним правилам.

5. 1 Основні поняття теорії нечітких множин

Точні значення вхідних змінних перетворюються на значення лінгвістичних змінних за допомогою застосування деяких положень теорії нечітких множин, а саме – за допомогою певних функцій приналежності.

Значення лінгвістичної змінної визначаються через так звані *нечіткі множини* (НМ), які в свою чергу визначені на деякому базовому наборі значень або базовій числовій шкалі, що має розмірність. Кожне значення ЛЗ визначається як нечітка множина (наприклад, НМ «низький зріст»).

Конкретне визначення ступеня приналежності можливо тільки при роботі з експертами. Під час обговорення питання про НМ лінгвістичної змінної цікаво прикинути, скільки всього НМ в змінної необхідно для досить точного уявлення фізичної величини. В даний час склалася думка, що для більшості застосувань досить 3-7 НМ на кожну змінну. Мінімальне значення числа НМ цілком виправдано. Таке визначення містить два екстремальних значення (мінімальне і максимальне) і середнє. Для більшості застосувань цього цілком достатньо. Що стосується максимальної кількості НМ, то вона не обмежена і залежить цілком від програми і необхідної точності опису системи. Число ж 7 обумовлено ємністю короткочасної пам'яті людини, в якій, за сучасними уявленнями, може зберігатися до семи одиниць інформації.

Нечітка множина визначається через деяку базову шкалу B і функцію приналежності НМ – $\mu(x_i)$, $x_i \in B$, приймаючу значення на інтервалі [0...1]. Отже, нечітка множина B – це сукупність пар виду $(x_i, \mu(x_i))$, де $x_i \in B$. Часто зустрічається і такий запис:

$$B = \sum_{i=1}^{n} \frac{x_i}{\mu \cdot x_i} \qquad (5.1)$$

де x_i. – i-е значення базової шкали.

Вигляд функції приналежності може бути абсолютно довільним. Зараз сформувалося поняття про так звані стандартні функції приналежності (див. рис. 5.1).

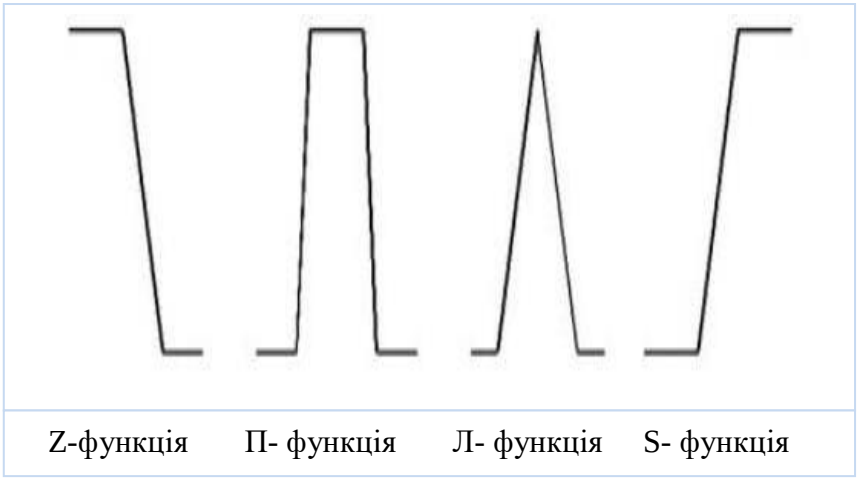

Рис. 5.1. Стандартні функції приналежності

Можна вказати основні правила визначення функцій приналежності і дати якусь подобу алгоритму по формалізації завдання в термінах нечіткої логіки.

Крок 1. Для кожного терма взятої лінгвістичної змінної знайти числове значення або діапазон значень, найкращим чином характеризують даний терм. Так як це значення або значення є «прототипом» нашого терма, то для них вибирається одиничне значення функції приналежності.

Крок 2. Після визначення значень з одиничною приналежністю необхідно визначити значення параметра з приналежністю «0» до даного терму. Це значення може бути вибрано як значення з приналежністю «1» до іншого терму з числа визначених раніше.

Крок 3. Після визначення екстремальних значень потрібно визначити проміжні значення. Для них вибираються П- або Л-функції з числа стандартних функцій приналежності.

Крок 4. Для значень, що відповідають екстремальним значенням параметра, вибираються S- або Z-функції приналежності.

Стандартні функції приналежності легко застосовні до вирішення більшості завдань. Однак якщо доведеться вирішувати специфічне завдання, можна вибрати і більш відповідну форму функції приналежності, при цьому можна добитися кращих результатів роботи системи, ніж при використанні функцій стандартного вигляду.

Як правило, функція приналежності визначає суб'єктивну ступінь впевненості експерта в тому, що дане конкретне значення базової шкали відповідає обумовленій НМ. Цю функцію не варто плутати з ймовірністю, що носить об'єктивний характер і підкоряється іншим математичним залежностям. Наприклад, для двох експертів визначення нечіткої множини «висока» для ЛЗ «ціна автомобіля» в умовних одиницях може істотно відрізнятися в залежності від їх соціального і фінансового становища.

Наприклад:

«Висока_ціна_автомобіля_1» = {50000/1+25000/0.8+10000/0.6+ 5000/0.4}.
«Висока_ціна_автомобіля_2» = {25000/1+10000/0.8 + 5000/0.7 + 3000/0.4}

Розглянемо наступний приклад.

Нехай перед нами стоїть завдання інтерпретації значень ЛЗ «*вік*», таких як «*молодий*» вік, «*похилий*» вік або «*перехідний*» вік. Визначимо «*вік*» як ЛЗ (рис. 5.2). Тоді «*молодий*», «*похилий*», «*перехідний*» будуть значеннями цієї лінгвістичної змінної. Більш повний базовий набір значень ЛП «*вік*» наступний:

В = {немовлячий, дитячий, юний, молодий, зрілий, похилий, старечий}.

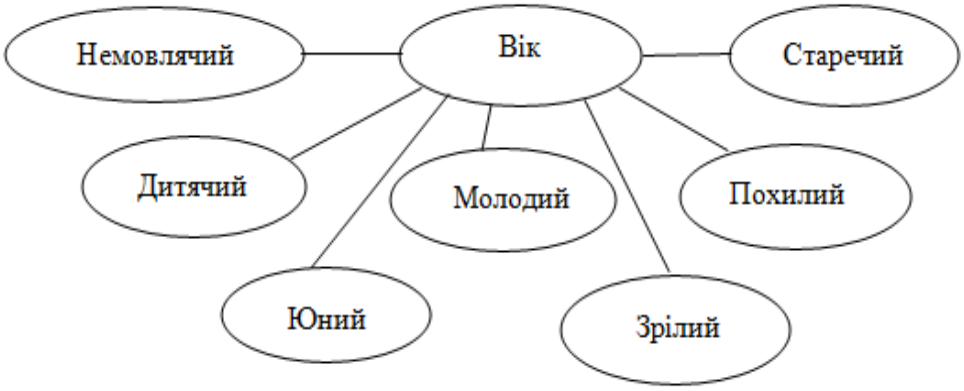

Рис. 5.2. Лінгвістична змінна «вік» і нечіткі множини, що визначають її значення

Для ЛЗ «вік» базова шкала – це числова шкала від 0 до 120, що позначає кількість прожитих років, а функція приналежності визначає, наскільки ми впевнені в тому, що дану кількість років можна віднести до цієї категорії віку. На рис. 5.3 відображено, як одні і ті ж значення базової шкали можуть брати участь у визначенні різних НМ.

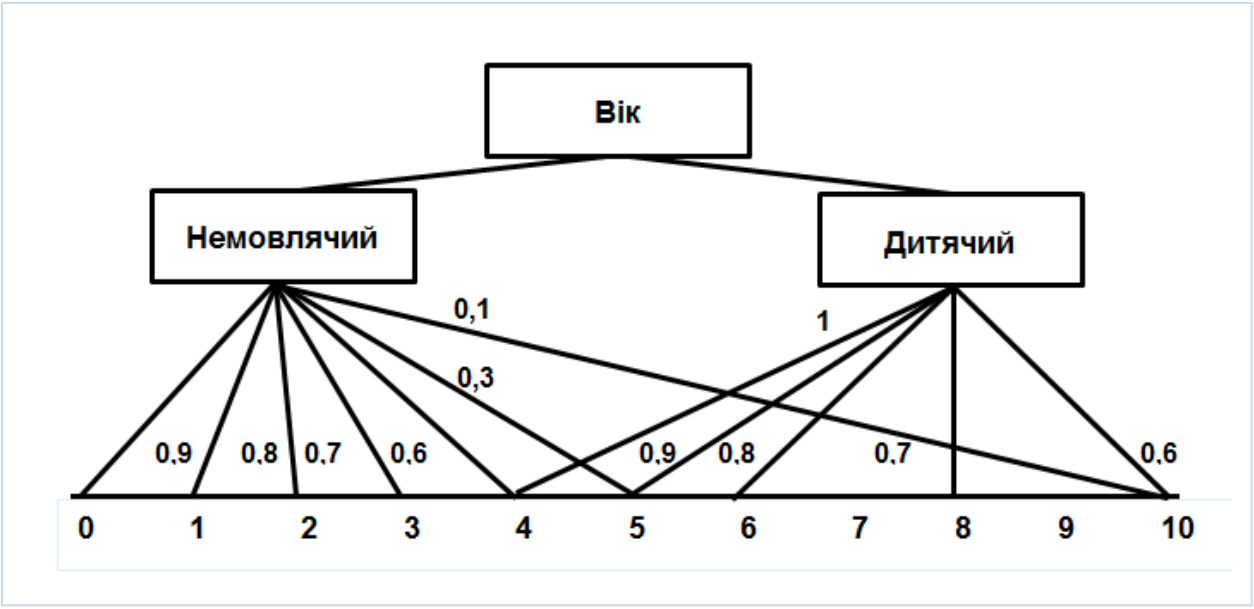

Рис. 5.3. Формування нечітких множин для базового набору значень ЛЗ «вік»

Наприклад, визначити значення НМ «дитячий вік» можна так:

$$немовлячий = \left\{ \frac{0,5}{1} + \frac{1}{0,9} + \frac{2}{0,8} + \frac{3}{0,7} + \frac{4}{0,6} + \frac{5}{0,3} + \frac{10}{0,1} \right\} \qquad (5.2)$$

Рисунок 5.4 ілюструє оцінку НМ якимсь усередненим експертом, який дитину до півроку з високим ступенем впевненості відносить до немовлят

(μ = 1). Діти до чотирьох років зараховуються до немовлят теж, але з меншим ступенем впевненості (0.5 < μ <0.9), а в десять років дитину називають так тільки в дуже рідкісних випадках – наприклад, для дев'яносторічної бабусі і 15 років може вважатися дитинством.

Таким чином, нечіткі множини дозволяють при визначенні понять враховувати суб'єктивні думки окремих індивідуумів (експертів).

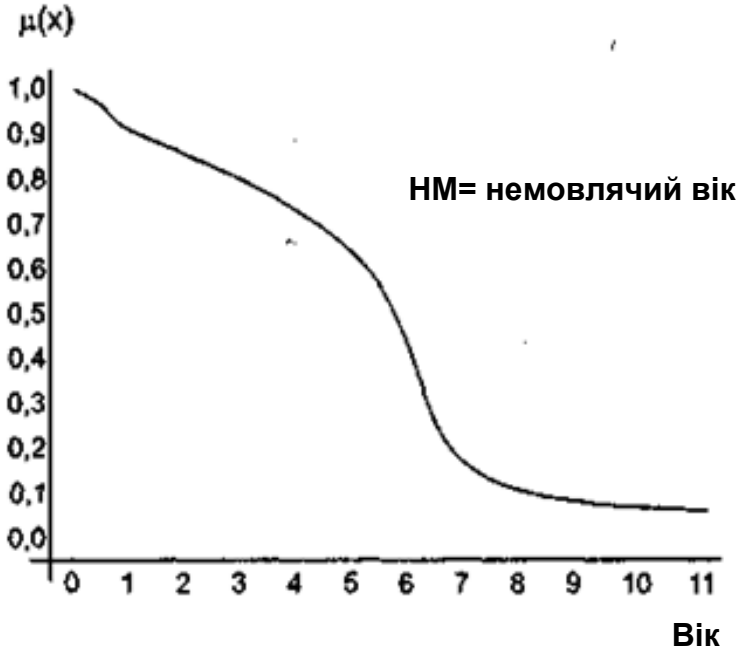

Рис. 5.4. Графік функції приналежності нечіткої множини «немовлячий вік»

5.2. Операції з нечіткими знаннями

Після визначення всіх НМ і їх функцій розподілу приступають до розробки бази знань на основі нечітких правил. Більшість нечітких систем використовують продукційні правила для опису залежностей між лінгвістичними змінними. Типове продукційне правило складається з антецедента (частина ЯКЩО ...) і консеквента (частина ТО ...).

Антецедентом (від лат. *Antecedens* – попередній) – в умовному вислові «Якщо А, то В» – є вислів «А»; а вислів «В» називається консеквентом. Наприклад, в умовному вислові «Якщо зараз день, то світло» антецедентом являється вислів «зараз день». Інколи термін «антецедент» використовується в поширеному сенсі для об означення посилки, підстави, причини, умови і т.п.

Антецедент може містити більше однієї посилки. В цьому випадку ЛП об'єднуються за допомогою логічних зв'язок І чи АБО.

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

проблемою невизначеності деяких характеристик. Для обліку цих невизначеностей розроблені спеціальні методи:

5.2.1. Метод «нечіткої логіки»

Зазначимо для прикладу правила визначення декількох операцій (нечітка логіка Заде)

1. Дві нечітких множини A і B називаються рівними, якщо:

$$\mu_A(u_i) = \mu_B(u_i), \quad u_i \in u. \tag{5.3}$$

2. Нечітка множина A міститься в B (A $\subset$ B), якщо:

$$\mu_A(u_i) \le \mu_B(u_i), u_i \in u. \tag{5.4}$$

3. Доповнення нечітких множин A-$\overline{A}$:

$$\overline{A} = \{\mu_{\overline{A}}(u_i) = 1 - \mu_A(u_i)\}, u_i \in U. \tag{5.5}$$

4. Об'єднання A і B - $A \cup B$ (операція АБО), де результат:

$$C = \sum_{i=1}^{n} \{\max\{\mu_A(u_i), \mu_B(u_i)\}/u_i\} \tag{5.6}$$

5. Перетин $A \cap B = C$ (операція I), де:

$$C = \{\mu_C(u_i) = \min(\mu_A(u_i), \mu_B(u_i))\} \tag{5.7}$$

6. Алгебраїчне множення:

$$A * B = C = \{\mu_C(u_i) = \mu_A(u_i) * \mu_B(u_i)\} \tag{5.8}$$

Слід зазначити, що класичний імовірнісний підхід по-іншому визначає операцію об'єднання:

$$\mu(x) = \mu 1(x) + \mu 2(x) - \mu 1(x) * \mu 2(x) \tag{5.9}$$

Посилення або ослаблення лінгвістичних понять досягається введенням спеціальних квантифікаторов. Наприклад, якщо поняття «старечий вік» *A* визначається як:

$$A = \left\{\frac{60}{0.6} + \frac{70}{0.8} + \frac{80}{0.9} + \frac{90}{1}\right\} \tag{5.10}$$

то поняття «дуже старий вік» визначиться як

$$con(A) = A^2 = \sum \frac{x_i}{\mu_i^2} \tag{5.11}$$

тобто «дуже старий вік» дорівнює:

$$A = \left\{\frac{60}{0.36} + \frac{70}{0.64} + \frac{80}{0.81} + \frac{90}{1}\right\} \tag{5.12}$$

5.2.2. Метод «коефіцієнта впевненості»

Метод «***коефіцієнта впевненості***» (КВ) – вперше був запропонований американським вченим Шортліффом **[**2, 3**]**. Він ввів поняття коефіцієнта впевненості для вимірювання ступеня наближення до якогось висновку h, що є результатом свідоцтва e на цей момент. Процедури, що виконуються при обчисленні КВ звуться «схемою Шортліффа». В загальному вигляді коефіцієнт впевненості може бути записаний таким чином:

$$КВ[h: e] = MD [h: e] - MHD [h: e] \qquad (5.13)$$

Тут КВ – коефіцієнт впевненості, MD – міра довіри, h – гіпотеза, e - свідоцтво, MHD – міра недовіри.

Значення КВ = [-1; 1], причому:

-1 – абсолютна неправда (хибність);

+1 – абсолютна істина;

0 – абсолютне незнання.

Ще раз нагадаємо, що MD і MHD – [0; 1] – не імовірнісні характеристики, а думка експерта. Для гіпотези h при двох свідченнях e1 і e2:

$$MD [h:e1;e2] = MD [h:e1] + MD [h:e1] (1 - MD [h:e2]) \qquad (5.14)$$

Ефективність другого свідоцтва на гіпотезу h при завданні свідоцтва e1 призводить до зміщення міри довіри в сторону повної визначеності на відстань, що залежить від e2. Ця формула має дві властивості:

1. Симетрична щодо e1 і e2.

2. У міру накопичення свідоцтв міра довіри рухається до визначеності.

Правила можуть забезпечуватися коефіцієнтом ослаблення числом 0-1, що показує надійність цього правила.

Основний недолік методу - дуже важко відрізнити випадок суперечливих свідчень від випадку недостатньої інформації.

5.2.3. «Байєсівський підхід» (інша назва - оцінка конкуруючих гіпотез)

Імовірність здійснення деякої ***гіпотези*** H при наявності певних підтверджуючих ***свідчень*** E обчислюється на основі апріорної ймовірності цієї гіпотези без підтверджуючих свідоцтв та ймовірності здійснення свідоцтв за умов, що гіпотеза вірна або невірна за відомою формулою Байеса.

Розглянемо приклад. Нехай P(h) – апріорна ймовірність того, що пацієнт захворів на грип. Необхідно дізнатися, чи дійсно даний пацієнт хворіє на грип. Відомо, що у нього висока температура. P(h:e) – ймовірність того, що якщо у пацієнта грип, то у нього висока температура.

Формула Байєса дозволяє уточнити апостеріорну умовну ймовірність гіпотези *H* при наявності свідоцтва *E*.

$$P(H : E) = \frac{P(H \,\&\, E)}{P(E)} \qquad (5.15)$$

За допомогою нескладних перетворень цю формулу можна привести до наступного вигляду:

$$P(H:E)=\frac{p^{+}p}{p^{+}p+p^{-}(1-p)} \qquad (5.16)$$

Тут справа присутні легко визначені вихідні апріорні ймовірності:

$$p=P(H),\ p^{+}=P(E:H),\ p^{-}=P(E:не\ H) \qquad (5.17)$$

Для функціонування системи необхідна ще одна формула, яка перераховує умовну ймовірність при відсутності свідоцтва (негативну відповідь користувача), тому що в деяких випадках це може призводити до збільшення умовної (апостеріорної) ймовірності гіпотези:

$$P(H:неE)=\frac{(1-p^{+})p}{1-p^{+}p+p^{-}(1-p)} \qquad (5.18)$$

Передбачається, що свідоцтва статистично незалежні, тому при розгляді чергового свідоцтва як апріорної ймовірності гіпотези приймається її поточна апостеріорна ймовірність.

Оскільки база знань (БЗ) системи на нечіткій логіці має імовірнісний характер, то користувачу надана можливість визначати наявність свідоцтва E також з деякою невизначеністю, висловлюючи свою відповідь q, наприклад, по десятибальній системі: -5 (тверде НІ), 0 (не знаю), 5 (тверде ТАК). При цьому приймається, що залежність умовної ймовірності гіпотези від величини q має кусочно-лінійний характер (див. рис. 5.5.)

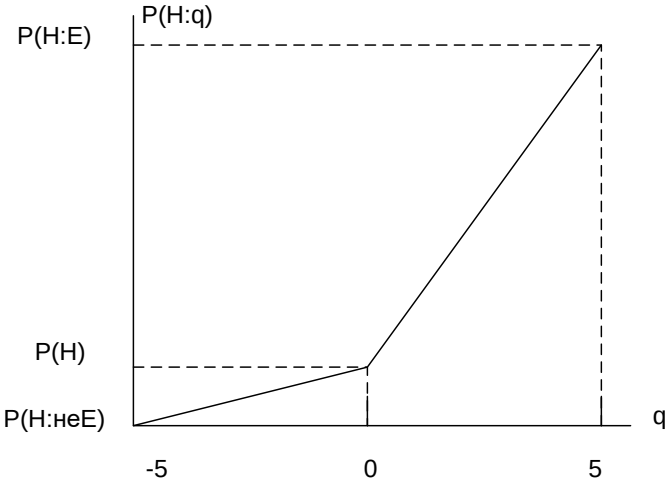

Рис. 5.5. Залежність умовної імовірності гіпотези від величини q

Тут $P(H:E), P(H:неE)$ визначаються формулами (5.16), (5.17), в яких $p=pt\equiv P(H)-$ поточна апріорна ймовірність гіпотези стосовно решти свідчень.

Відповідно до цього графіку після відповіді користувача шукана умовна (нова поточна апріорна) ймовірність гіпотези коригується за такими формулами:

$$P(H:q) = P(H) + (P(H:E) - P(H))\frac{q}{5}, \quad 0 \le q \le 5$$
$$P(H:q) = P(H:неE) + (P(H) - P(H:неE))\frac{(5+q)}{5}, \quad -5 \le q < 0 \qquad (5.19)$$

З програмістської точки зору іноді зручніше користуватися характеристикою, яка називається «*шансом*».

Перетворення ймовірності в «*шанс*» виконується наступним чином:

$$O = P / (1\text{-}P) \qquad (5.20)$$

Наприклад, якщо ймовірність одужання P = 0.3, то шанс одужати:

$$O = 0.3 / (1\text{-}0.3) = 0.43. \qquad (5.21)$$

Зворотне обчислення: P = O / (1 + O).

У ряді систем використовують так зване *відношення правдоподібності* (ВП) або ВП (h):

$$ВП(h) = P(h:e) / P(h:\neg e)$$
$$ВП(e) = P(e:h) / P(e:\neg h) \qquad (5.22)$$

Наприклад, це відношення ймовірності отримати позитивний результат діагностичного тесту у хворих до ймовірності отримати позитивний результат тесту у здорових осіб.

Тоді шанс після досвіду:

$$O'(h) = O(h) * ВП(h: e) \qquad (5.23)$$

Відношення правдоподібності ВП>0 завжди, причому ВП>1 – свідоцтво на користь гіпотези, а ВП<1 – проти гіпотези.

Значення ВП = 1 – говорить про те, що свідчення не впливають на ВП гіпотези. Іноді можуть зустрітися ВП = 0 або ∞. Тоді необхідно підібрати дані так, щоб уникнути цих величин.

ВП показує, наскільки більш імовірною стає гіпотеза при наявності свідоцтв, ніж при їх відсутності. Якщо свідоцтва ненадійні, то:

$$ОП' = ОП * ВС + (1\text{-}ВС) \qquad (5.24)$$

Тут ВС – вага свідоцтва – ймовірність того, що свідчення надійно.

Таким чином, основними перевагами байєсівського підходу є наступні можливості:

1) допускається комбінування декількох незалежних джерел даних;

2) легко коригується формула для розрахунку після появи свідчень.

Для виведення на нечітких множинах використовуються вищевказані спеціальні відносини і операції над ними. Одним з перших застосувань теорії НМ стало використання коефіцієнтів впевненості для виведення рекомендацій медичної експертної системи MYCIN [20].

Цей метод використовує кілька евристичних прийомів. Він став прикладом обробки нечітких знань, що вплинули на розробку наступних систем. Нечіткі експертні системи, крім своєї основної переваги – кращої адаптації до умов реального світу, мають ще дві переваги в порівнянні з традиційними.

По-перше, вони вільні від т.зв. «циклічних блокувань» при побудові висновків. По-друге, різні бази нечітких правил можна з легкістю об'єднувати, що рідко вдається в звичайних експертних системах. В даний час в більшість інструментальних засобів розробки систем, заснованих на знаннях, включені елементи роботи з НМ, крім того, розроблені спеціальні програмні засоби реалізації так званого нечіткого виведення, наприклад «оболонка» FuzzyCLIPS.

У більш широкому розумінні нечітке управління – це методологія створення систем управління, в яких відображення між реальними вхідними даними і вихідними параметрами представлено за допомогою нечітких правил. Нечітке управління виявилося дуже успішним в таких комерційних продуктах, як автоматичні коробки передач, відеокамери і електричні бритви. Критики цього підходу стверджують, що такі додатки виявилися успішними тому, що в них використовуються невеликі бази правил, логічні висновки не формують ланцюжки, а для підвищення продуктивності системи може здійснюватися настройка параметрів. І дійсно, той факт, що правила функціонування цих систем реалізовані за допомогою нечітких операторів, може не бути ключовим фактором їх успіху; секрет полягає в тому, щоб застосовувався лаконічний і інтуїтивно зрозумілий спосіб завдання гладко інтерпольованої функції з реальними значеннями.

6. Стратегії управління висновком

Серед відомих моделей подання знань, розглянутих в розділі 2, найбільшого поширення набула продукційна модель. При використанні продукційної моделі база знань системи штучного інтелекту складається з набору правил (ядер продукцій). Програма, що управляє перебором правил, називається машиною виведення [1]. Розглянемо більш докладно задачі і роботу машини виведення продукційної системи.

Будь-яка експертна система продукційного типу повинна містити три основні компоненти: базу правил, робочу пам'ять і механізм виведення [8].

База правил (БП) – формалізовані за допомогою правил продукцій знання про конкретну предметну область.

Робоча пам'ять (РП) – область пам'яті, в якій зберігається безліч фактів, що описують поточну ситуацію, і всі пари атрибут-значення, які були встановлені до певного моменту. Вміст РП в процесі виконання завдання змінюється зазвичай, збільшуючись в обсязі в міру застосування правил. Іншими словами, РП – це динамічна частина бази знань, вміст якої залежить від оточення розв'язуваної задачі. У найпростіших експертних систем (ЕС) збережені в РП факти не змінюються в процесі виконання завдання, однак існують системи, в яких допускається зміна і видалення фактів з РП. Це системи, що працюють в умовах неповноти інформації.

Механізм виведення виконує дві основні функції:

- перегляд існуючих в робочій пам'яті фактів і правил з БП, а також додавання в РП нових фактів;

- визначення порядку перегляду і застосування правил. Порядок може бути прямим або зворотним.

Прямий порядок – від фактів до висновків. В експертних системах з прямими висновками відомих фактів відшукується висновок, який слідує з цих фактів. Якщо такий висновок вдається знайти, він заноситься в робочу пам'ять. Прямі висновки часто застосовуються в системах діагностики, їх називають висновками, керованими даними.

Зворотний порядок виведення – від висновків до фактів. У системах зі зворотним висновком спочатку висувається деяка гіпотеза про кінцеве судження, а потім механізм виведення намагається знайти в робочій пам'яті факти, які могли б підтвердити або спростувати висунуту гіпотезу. Процес відшукання необхідних фактів може включати досить велике число кроків, при цьому можливо висування нових гіпотез (цілей). Зворотні висновки управляються цілями.

У переважній більшості систем, заснованих на знаннях, механізм виведення являє собою невелику за обсягом програму і включає два компонента – один реалізує власне вивід (компонента виведення), інший керує цим процесом (компонента управління). Дія компонента виведення заснована на застосуванні правила, званого modus ponens (правило виводу). Правило виводу «модус поненс», звичайно називане «правилом відділення» або «гіпотетичним силогізмом», дозволяє від твердження умовного висловлювання і твердження його основи (антецедента) перейти до твердженню слідства (консеквента).

Правило modus ponens (використовується в логіці предикатів і являється, по суті, одним з методів правил виводу), з двох виразів A і $A \rightarrow B$ виводить новий вираз B.

Іншими словами, якщо відомо, що істинно твердження A і висловлювання «A» тягне (імплікує) висловлювання «B» (тобто існує правило виду «ЯКЩО A, ТО B»), тоді твердження B також істинно.

Розглянемо наступний приклад. Нехай істинно твердження «барометр падає» і існує правило «Якщо барометр падає, то можливий дощ». Тоді твердження «можливий дощ» також істинно.

Правила спрацьовують, коли знаходяться факти, що задовольняють їх лівій частині: якщо істинна посилка, то повинен бути істинним і висновок.

Розглянемо приклад виведення рішення в логічної моделі на основі правила виведення – modus ponens.

Розглянемо твердження:

- «Сократ – людина»;
- «Людина –- це жива істота»;
- «Всі живі істоти смертні».

Потрібно довести твердження «Сократ смертний».

Рішення.

Крок 1. Уявімо висловлювання в предикативний формі:

УТВЕРЖДЕНИЕ	ПРЕДИКАТНАЯ ФОРМА
«Человек – это живое существо»	∀ (***X***)(Человек(***X***)→ Живое_существо(***X***))
«Сократ – человек»	Человек(Сократ)
«Все живые существа смертны»	∀ (***Y***)(Живое_существо(***Y***)→ Смертно(***Y***))

Крок 2. На основі правила виведення (modus ponens) і підстановки (Сократ/X) в першому предикаті отримаємо твердження:

«Сократ – это живое существо»

Крок 3. На основі правила виведення (modus ponens) і підстановки (Сократ/Y) в третьому предикаті отримаємо твердження:

«Сократ – смертен»

Компонент виведення повинен функціонувати навіть при нестачі інформації. Отримане рішення може і не бути точним, проте система не повинна зупинятися через те, що відсутня будь-яка частина вхідної інформації.

Таким чином, керуючий компонент визначає порядок застосування правил і виконує чотири функції.

1. Зіставлення – зразок (антецедент) правила зіставляється з наявними в РП фактами.
2. Дозвіл конфліктного набору – вибір одного з декількох правил в тому випадку, якщо їх можна застосувати одночасно.
3. Вибір – якщо в конкретній ситуації може бути застосовано відразу кілька правил, то з них вибирається одне, найбільш підходяще по заданому критерію (вирішення конфлікту).
4. Спрацьовування правила – якщо зразок правила при зіставленні збігся з будь-якими фактами з робочої пам'яті, то правило спрацьовує і при цьому воно відзначається в БП.
5. Дія – робоча пам'ять піддається зміні шляхом додавання в заключення правила, що спрацювало. Якщо в правій частині правила міститься вказівка на будь-яку дію, то воно виконується (як, наприклад, в системах забезпечення безпеки інформації).

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

В одному циклі може спрацювати тільки одне правило. Якщо кілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, яке спрацьовує в даному циклі. Цикл роботи інтерпретатора схематично представлений на рис. 6.1.

Інформація з робочої пам'яті послідовно зіставляється з посилками правил для виявлення успішного зіставлення. Сукупність відібраних правил складає так звану конфліктну множину. Для вирішення конфлікту інтерпретатор має критерій, за допомогою якого він обирає єдине правило, після чого воно спрацьовує. Це виражається в занесенні фактів, що утворюють висновок правила, в робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо ж у заключення правила входить назва якої-небудь дії, то вона виконується. Робота машини висновку залежить тільки від стану робочої пам’яті і від складу бази знань.

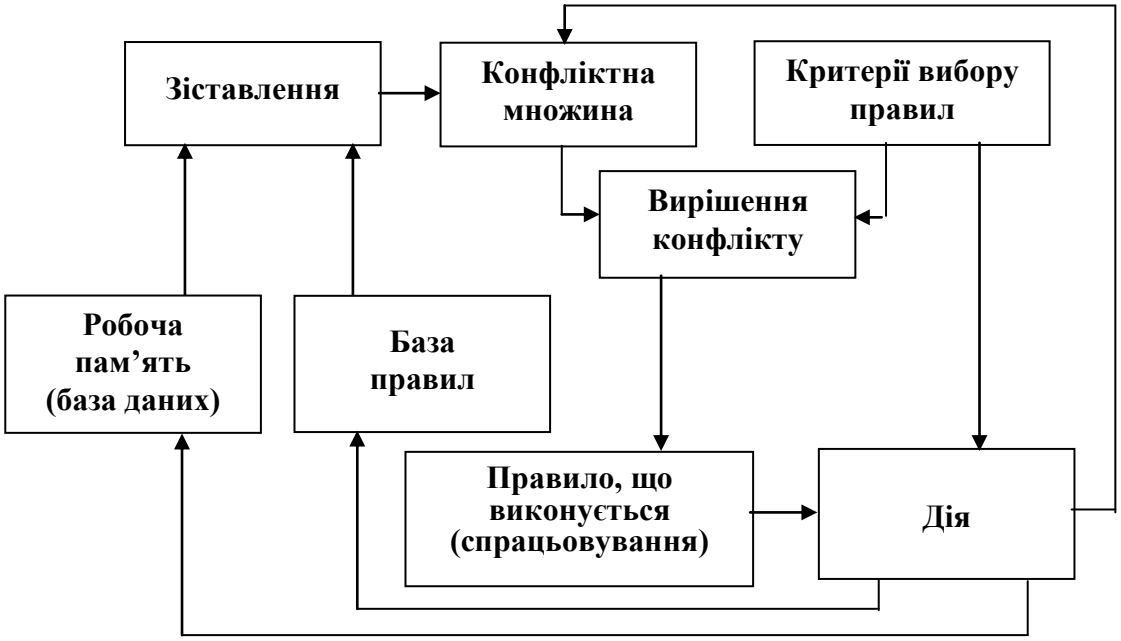

Рис. 6.1. Цикл роботи інтерпретатора правил виводу

На практиці зазвичай враховується історія роботи, тобто поведінка механізму виводу у попередніх циклах. Інформація про поведінку механізму виводу запам’ятовується в пам’яті станів (рис. 6.2). Зазвичай пам’ять станів має протокол роботи системи.

Від обраного методу пошуку, тобто стратегії виводу, буде залежати порядок застосування і спрацьовування правил. Процедура вибору зводиться до визначення направлення пошуку і способу його здійснення. Процедури, що реалізують пошук, зазвичай «зашиті» в механізм виводу, тому в більшості інтелектуальних систем інженери знань не мають до них доступу і, отже, не можуть в них нічого змінити за власним бажанням.

При розробці стратегії управління виводом важливо визначити два питання:

1. Яку точку у просторі станів прийняти у якості вихідної? Від вибору цієї точки залежить і метод здійснення пошуку – у прямому або зворотному напрямку.

2. Якими методами можна підвищити ефективність пошуку рішення? Ці методи визначаються обраною стратегією перебору – в глибину, в ширину, за підзадачами або якось інакше.

Простір станів – це граф, вершини якого відповідають ситуаціям, що зустрічаються в задачі («проблемні ситуації»), а рішення задачі зводиться до пошуку шляху в цьому графі.

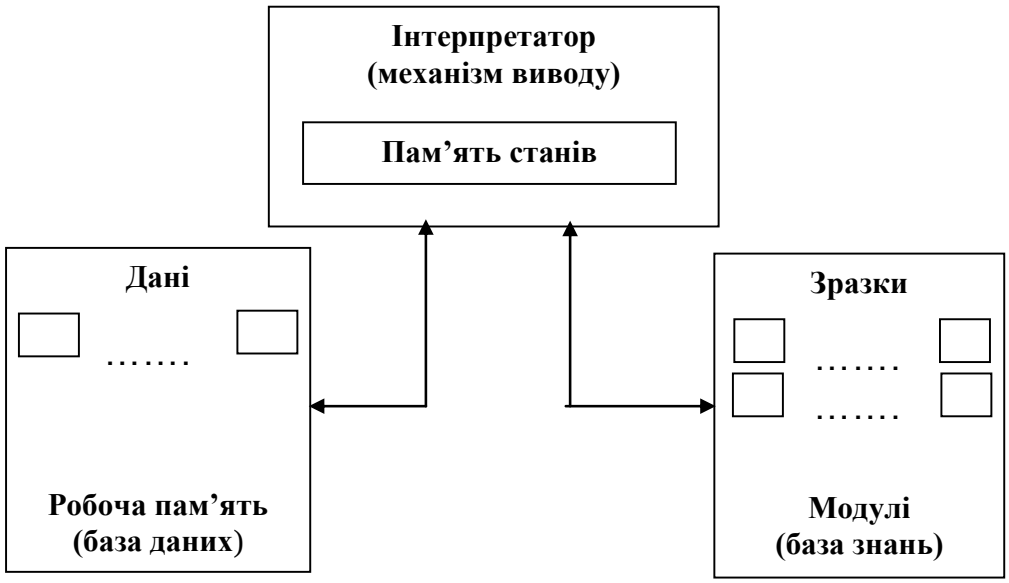

Рис. 6.2. Схема функціонування інтерпретатора правил

6.1. Прямий і зворотній вивід

При зворотному порядку виводу на початку висувається деяка гіпотеза, а потім механізм виводу як би повертається назад, переходячи до фактів, намагаючись знайти ті, які підтверджують гіпотезу (мал. 6.3, ліва частина).

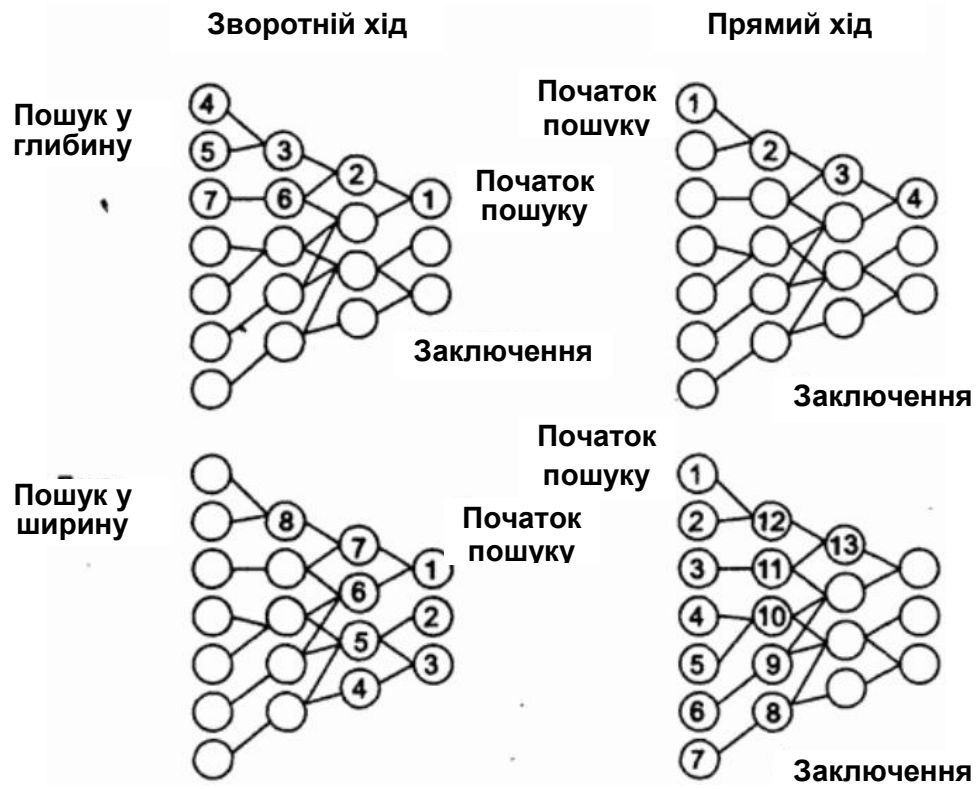

Рис. 6.3. Стратегії виводу

Якщо гіпотеза виявилась правильною, то обирається наступна гіпотеза, деталізуюча першу і яка являється по відношенню до неї підметою. Далі відшукуються факти, підтверджуючі істинність підпорядкованої гіпотези. Вивід, при якому пошук доказу починається з цільового затвердження. З'ясовуються умови, при яких цільове твердження є виведеним. Ці умови приймаються за нові цільові затвердження і процес пошуку триває. В.З. (Висновок Зворотний) закінчується, коли всі чергові умови виявляються аксіомами або процес умов обривається не приводячи до аксіом. Висновок такого типу називається керованим цілями, або керованим консеквентами. Зворотний пошук застосовується в тих випадках, коли цілі відомі і їх порівняно небагато.

У системах з прямим висновком відомих фактів відшукується висновок, який з цих фактів слідує (див. мал. 6.3, права частина).

Якщо такий висновок вдається знайти, то він заноситься в робочу пам'ять. Висновок, що веде від вихідних аксіом до цільового вираження.

При В.П. (Висновок Прямий) через неоднозначність вибору застосовних аксіом і правил виведення утворюється дерево рішень і процес знаходження ланцюжка, що веде від вихідних аксіом до цільового вираження, є переборним.

Стандартною процедурою, використовуваної при обході дерева рішень, є процедура повернення – бектрекінг. Прямий висновок часто називають висновком, керованим даними, або висновком, керованим антецедентами.

Існують системи, в яких висновок ґрунтується на поєднанні згаданих вище методів - зворотного і обмеженого прямого. Такий комбінований метод отримав назву циклічного.

Приклад 1.

Є фрагмент бази знань виробничої системи, що має два правила:

П1. Якщо «відпочинок - влітку» і «людина - активна», то «їхати в гори».

П2. Якщо «любить сонце», то «відпочинок влітку».

Припустимо, в систему надійшли факти – «людина активна» і «любить сонце».

ПРЯМИЙ ВИСНОВОК – виходячи з фактичних даних, отримати рекомендацію.

1-й прохід.

Крок 1. Пробуємо П1, не працює (не вистачає даних «відпочинок – влітку»). Крок 1. Пробуємо П2, працює, в базу надходить факт «відпочинок – влітку».

2-й прохід.

Крок 1. Пробуємо П1, працює, активується мета «їхати в гори», яка і виступає як порада, яку дає експертна система.

ЗВОРОТНИЙ ВИСНОВОК – підтвердити обрану мета за допомогою наявних правил і даних.

1-й прохід.

Крок 1. Мета – «їхати в гори»: пробуємо П1 даних «відпочинок – влітку» немає, вони стають новою метою і шукається правило, де мета – в лівій частині.

Крок 1. Мета «відпочинок – влітку»: правило П2 підтверджує мету і активує її.

2-й прохід.

Крок 1. Пробуємо П1, підтверджується шукана мета.

6.2. Суть основних стратегій управління виводом

В інтелектуальних системах прийняття рішень, база знань яка налічує сотні правил, бажаним є використання стратегії управління виводом, що дозволяє мінімізувати час пошук рішення і, тим самим, підвищити ефективність виведення. До числа таких стратегій відносяться [1-4]:

1. пошук в глибину;
2. пошук в ширину;
3. розбиття на під задачі;
4. альфа-бета алгоритм;
5. стратегія простоти / складності;
6. LEX-стратегія;
7. MEA-стратегія.

Розглянемо найпростіший приклад завдань – побудувати осмислене слово з деякого набору букв (кирилиця – літери к, о, т). На кожному рівні додаємо по букві (рис. 6.4).

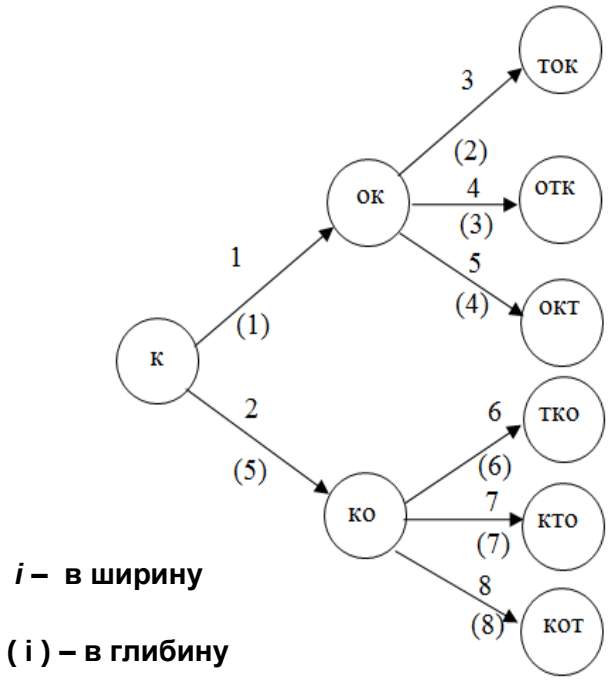

Рис. 6.4. Стратегії пошуку в ширину і в глибину

По мірі збільшення числа рівнів на графі (тут – включення в набір інших букв) спостерігається експоненціальне зростання числа вузлів, – так званий комбінаторний вибух. Тому актуальною є задача вибору відповідної стратегії пошуку. Розглянемо їх.

При пошуку в глибину у якості чергової підцілі обирається та, яка відповідає наступному, більш детальному рівню опису завдання. Правила, обрані в список заявок на підставі даних, які були включені в робочу пам'ять порівняно недавно, розташовуються в цьому списку раніше правил, при виборі яких використані більш старі дані. Наприклад, діагностуюча система, зробивши на основі відомих симптомів припущення про наявність певного захворювання, буде продовжувати запитувати уточнюючі ознаки і симптоми цієї хвороби до тих пір, поки повністю не підтвердить або НЕ спростує висунуту гіпотезу. На графі це відповідає побудові нащадків вузла, а потім – повернення до сусіднього вузла на тому ж рівні графа.

Алгоритм пошуку в глибину може швидше знайти рішення, особливо, якщо при його виконанні використовуються евристики для вибору чергової гілки (на пошук поєднання «окт» потрібно 4 кроку). Але цей алгоритм може ніколи не закінчитися, якщо простір станів нескінченний.

При пошуку в ширину, правила, обрані в списку заявок на підставі даних, які були включені в робочу пам'ять порівняно давно, розташовуються в цьому списку раніше правил, при виборі яких використані більш свіжі дані. Наприклад, діагностуюча система спочатку проаналізує всі симптоми, що знаходяться на одному рівні простору станів, навіть якщо вони належать до різних захворювань, і лише потім перейде до симптомів наступного рівня детальності. На графі це відповідає побудові всіх сусідів вузла на тому ж рівні, а потім будуються його нащадки. Алгоритм пошуку в ширину відшукує рішення, шлях до якого на графі – найкоротший, якщо таке існує, тобто він знаходить найкоротший шлях між початковим станом і рішенням. Алгоритми, що мають такі властивості, називаються розв’язними (*рос*. – разрешимыми).

Розбиття на підзадачі – має на увазі виділення підзадач, вирішення яких розглядається як досягнення проміжних цілей на шляху до кінцевої мети. Прикладом, що підтверджує ефективність стратегії розбиття на підзадачі, є пошук несправностей в комп'ютері. Спочатку виявляється підсистема, яка відмовила у роботі (живлення, енергоспоживання, пам'яті і т.д.), що значно звужує простір пошуку. Якщо вдається правильно зрозуміти сутність завдання і оптимально розбити її на систему ієрархічно пов'язаних цілей-підцілей, то можна домогтися того, що шлях до її вирішення в просторі пошуку буде мінімальним. Альфа-бета алгоритм дозволяє зменшити простір станів шляхом видалення гілок, неперспективних для успішного пошуку. Тому проглядаються тільки ті вершини, в які можна потрапити в результаті наступного кроку, після чого неперспективні напрямки виключаються. Альфа-бета алгоритм знайшов широке застосування в основному в системах, орієнтованих на різні ігри, наприклад в шахових програмах.

Стратегія простоти / складності визначається кількістю операцій перевірки, які потрібно виконати при аналізі умов даного правила. Перевага віддається більш простим, або, навпаки, більш складним правилам.

LEX-стратегія передбачає спочатку видалення зі списку заявок всіх правил, які вже були раніше використані. Решта правила з рівним значенням нерівності потім відсортовується за показниками «новизни» використовуваних даних. Якщо виявиться, що два правила використовують дані однаковою «свіжості», то перевага віддається тому правилу, яке залучає до аналізу передумов більше даних.

MEA-стратегія багато в чому аналогічна попередній, але при аналізі новизни беруться до уваги тільки перші умови в передумовах правил. Якщо виявиться, що в списку заявок виявилися два претендента з рівними показниками, то для вибору між ними застосовується механізм LEX-стратегії. MEA – це абревіатура однією з перших методик вирішення завдань штучного інтелекту шляхом побудови зворотного ланцюжка міркувань Mean-Ends Analysis (засіб-аналіз результату).

В сучасних системах штучного інтелекту (СШІ) використовуються частіше LEX і MEA стратегії, причому LEX показала себе як хороша стратегія загального застосування, в той час як MEA є ефективною стратегією при вирішенні більш специфічних завдань, таких як планування.

Крім розглянутих вище, відома також так звана евристична стратегія пошуку, в якій до алгоритму пошуку підключаються додаткові знання про завдання. Проста форма евристичного пошуку – «сходження на гору». Тут в процесі пошуку використовується деяка оціночна функція, за допомогою якої можна грубо оцінити, наскільки «хорошим» є поточний стан. Ось цей алгоритм:

1) Перебуваючи в даній точці простору станів, застосувати правила породження нової множини можливих рішень, наприклад, безліч ходів шахових фігур, допустимих в даній позиції.

2) Якщо одне з нових станів є вирішенням проблеми, то припинити процес. Інакше – перейти в той стан, який характеризується найвищим значенням оціночної функції і повернутися до кроку 1.

До труднощів в реалізації алгоритму сходження можна віднести такі.

- Як задати оціночну функцію?
- Як діяти, коли всі можливі ходи однаково хороші або погані (при сходженні – «вихід на плато»)?
- А якщо є локальні максимуми, з яких можливий тільки спуск, тобто «погіршення» стану (взяти ферзя і після цього програти)?

Інша форма евристичного методу – «спочатку найкращий». Тут порівнюються не тільки ті стани, в які можливий перехід з поточного, але і всі, до яких можна «дістати» (окинути поглядом якомога більшу ділянку простору станів і бути готовим, якщо буде потреба, повернутися туди, де ми вже були і піти іншим шляхом).

Різновид цієї форми - «перший найкращий», при якому мінімізується функція $f(n) = g(n) + h(n)$, де:

g(n) – відстань на графі від вузла *n* до початкового стану простору пошуку;

h(n) – відстань на графі від вузла *n* до кінцевого стану простору пошуку.

Висновки по стратегіям управління виводом.

1) Проблему будь-якої складності, в принципі, можна звести до проблеми пошуку в просторі станів, якщо тільки вдається її формалізувати в термінах початкового стану, кінцевого стану і операцій переходу в просторі станів.

2) Пошук в просторі станів повинен направлятися певним чином представленими знаннями про конкретну предметну область.

Підхід на основі стратегій пошуку рішень в продукційних ЕС відомий досить давно. Вельми популярна на початку 90-х років ЕС GURU (ІНТЕР-ЕКСПЕРТ) також використовувала подібні механізми управління стратегіями пошуку. Можливість зміни стратегії в результаті виконання завдання програмним чином і накопичення досвіду, які стратегії дають кращі результати для певних класів задач, дозволяє отримати ефективні механізми пошуку рішень на основі продукцій.

Підводячи підсумки, слід зазначити, що існують різні методи пошуку рішень в семантичних мережах, наприклад, метод обходу семантичної мережі – мультіпарсінг. Даний метод оригінальний тим, що дозволяє паралельно «вести» по графу кілька маркерів і, тим самим, розпаралелювати процес пошуку інформації в семантичній мережі, що збільшує швидкість пошуку. Ці методи використовуються, як правило, при поданні тексту у вигляді об'єктно-орієнтованої семантичної мережі і тут не розглядаються.

Пошук в мережах фреймів, заснований на прецедентах висновок (Case-based Reasoning – CBR), правдоподібні міркування (plausible reasoning), методи пошуку на основі нечіткої логіки та інші методи пошуку рішень ШІ також слід розглядати окремо. Їх рекомендується вивчити самостійно.

7. Агентні технології в розподіленому аналізі даних

7.1. Поняття програмного агента; його завдання, структура, властивості

Одним із наслідків розвитку ідеї організації розподілених обчислень за рахунок перенесення виконуваного коду стало зростання інтересу до так званих програмних агентів і технологій їх використання.

Е. Таненбаум запропонував наступне визначення [1]. ***Програмний агент*** – це автономний процес, здатний реагувати на середовище виконання і викликати зміни в середовищі виконання, можливо, в кооперації з користувачами або іншими агентами. При цьому агент сам піддається впливу з боку середовища [21 – 23].

Е. Таненбаум також ввів класифікацію агентів, в якій виділяються наступні основні типи.

Стаціонарні та мобільні агенти. Мобільні агенти, на відміну від стаціонарних, здатні переміщатися від одного вузла обчислювальної середовища (ВС) до іншого.

Кооперативні та конкуруючі. Кооперативний агент - здатний об'єднуватися з іншими агентами для вирішення загального завдання. Конкуруючий - здатний конкурувати з іншими агентами з метою захисту інтересів свого власника (наприклад, торгові агенти на біржі).

Агенти можуть застосовуватися при вирішенні наступних завдань:

- мобільні обчислення: міграція агентів може підтримуватися не тільки між постійно приєднаними до мережі вузлами, а й між мобільними платформами, що підключаються до постійної мережі на деякі проміжки часу і можливо по низькошвидкісних каналах. Клієнт приєднується до постійної мережі на короткий проміжок часу з мобільної платформи, відправляє агента для виконання завдання і від'єднується; потім клієнт під'єднується до іншої точки мережі і забирає результати роботи агента. Другий варіант - сервер, на який повинен переміститися агент, під'єднується до мережі, а потім від'єднується. У цьому випадку агент повинен вміти переміститися на такий тимчасово приєднується сервер і повернутися в постійну мережу.

– завдання управління інформацією;

– пошук інформації (море інформації – одна людина не в змозі знайти необхідну йому інформацію і проаналізувати її - використання агента, який мандрує по мережі в пошуках інформації, найкраще задовольняє потреби людини); пошукові агенти містять відомості про різні інформаційних джерелах (включаючи тип інформації, спосіб доступу до неї, а також такі характеристики інформаційного джерела, як надійність і точність даних);

– відбір (обробка) інформації. З усіх даних, що приходять до клієнта, вибирають тільки ті дані, які можуть бути цікаві клієнту. Використовуються в комбінації з пошуковими агентами (спочатку пошук, потім - відбір);

– моніторинг даних. Повідомлення користувача про зміни в різних джерелах даних в реальному часі (наприклад, мобільний агент переміщається на обчислювальний вузол, на якому розташоване джерело даних; це ефективніше, ніж використовувати статичного агента, що посилає запити джерела даних);

– універсальний доступ до даних. Агенти – посередники для роботи з різними джерелами даних, що мають механізми для взаємодії один з одним (наприклад, агент створює кілька агентів, кожен з яких працює зі своїм джерелом даних).

С. Франклін і А. Грессер в 1996 році запропонували наступне узагальнене визначення агента:

Автономний агент– це система, що знаходиться всередині оточення і є його частиною, яка сприймає це оточення (його сигнали) і впливає на оточення для виконання власної програми дій.

Можна виділити наступні основні складові автономного агента (рис. 7.1):

1. ***Сенсори*:** блоки агента, щоб забезпечити отримання інформації про навколишнє середовище та інших агентів;

2. ***Актуатори*:** блоки агента, що забезпечують вплив на навколишнє середовище.

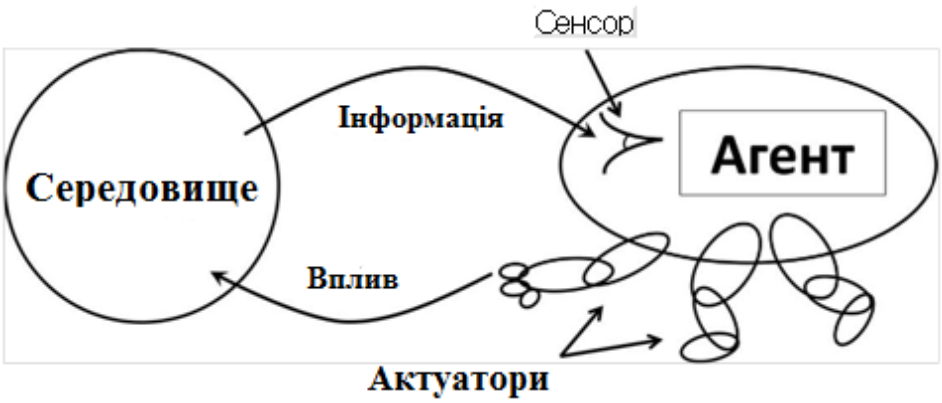

Рис. 7.1. Автономний агент.

При роботі простий автономний агент керується стандартним набором правил «Якщо-То» (мал. 7.2)

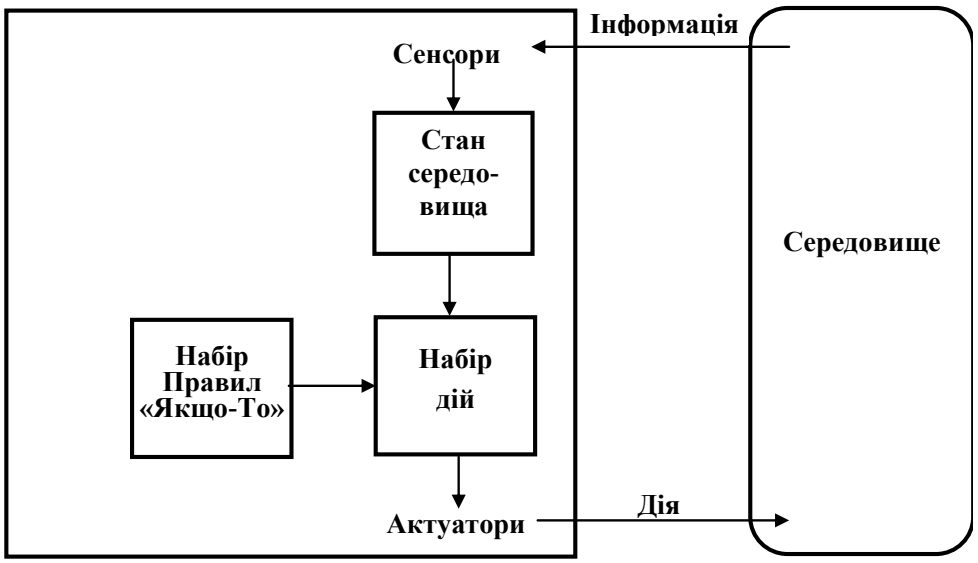

Рис. 7.2. Структура автономного агента.

Автономний агент повинен мати наступні властивості:

- реактивність;
- автономність;
- цілеспрямованість;
- комунікативність.

Різні автори не зовсім однаково трактують перераховані властивості. Спробуємо пояснити їх докладніше.

1. Властивість реактивності означає, що агент часом відповідає на зміни в оточенні. Агент має сенсори, за допомогою яких отримує інформацію від оточення. Сенсори можуть бути самими різними. Це можуть бути мікрофони, які сприймають акустичні сигнали і перетворюють їх в

електричні, відеокарти захоплення зображень, клавіатура комп'ютера або загальна область пам'яті, в яку оточення поміщає дані і з якої програмний агент бере дані для обчислень. Не всі зміни оточення стають відомими (доступними) сенсорам агента. Це цілком природно. Адже і людина не сприймає звуки частотою понад 30 кГц, радіохвилі і т.д. Таким чином, оточення не є повністю контрольоване для агента. Аналогічно, агент впливає на оточення, шляхом різноманітних виконавчих механізмів, включаючи загальну пам'ять. Зрозуміло, що ступінь впливу як і ступінь сприйняття є обмеженою. Агент може перевести оточення з деякого стану в деяке інше, але не з будь-якого в будь-який.

2. Властивість автономності означає, що агент є самоврядним, сам контролює свої дії. Програмний агент, що знаходиться на деякому сервері, має можливість «самозапуску». Він не вимагає від користувача будь-яких спеціальних дій щодо забезпечення його старту (подібно до того, як ми "натискаємо" два рази по іконці деякого файлу).

3. Властивість цілеспрямованості означає, що у агента є певна мета і його поведінка (дія на оточення) підпорядковано цій меті, а не є простим відгуком на сигнали з оточення. Інакше кажучи, агент є керуючою системою, а не керованим об'єктом.

4. Властивість комунікативності означає, що агент спілкується з іншими агентами (включаючи людей), використовуючи для цього певну мову. Це не обов'язково єдина мова для всіх агентів. Досить, щоб у пари агентів, які спілкуються була спільна мова. Мова може бути складна як, наприклад, природна мова. Але може бути і примітивна: обмін числами або короткими словами. Якщо багатослівні фрази складної мови несуть всю інформацію, як правило, в собі, то слова простої мови припускають "замовчування": обидві сторони діалогу "знають", про що йде мова (як у відомому анекдоті про занумерованих анекдотах).

5. В окрему категорію інтелектуальних агентів виділяють автономних агентів, що володіють властивістю навченості. Властивість навченості означає, що агент може коригувати свою поведінку, грунтуючись на попередньому досвіді. Це не просто накопичення в пам'яті параметрів оточення, тобто використання історичних даних, але зіставлення історії власних дій з історією їх впливу на оточення, і зміна в зв'язку з цим своєї програми дій.

6. Одна з найголовніших особливостей агента - це інтелектуальність. Інтелектуальний агент володіє певними знаннями про себе і про навколишнє середовище, і на основі цих знань він здатний визначати свою поведінку (мал. 7.3). Інтелектуальні агенти є основною сферою інтересів агентної технології. Важливе також середовище існування агента: це може бути як реальний світ, так і віртуальний, що стає важливим у зв'язку з широким розповсюдженням мережі Internet.

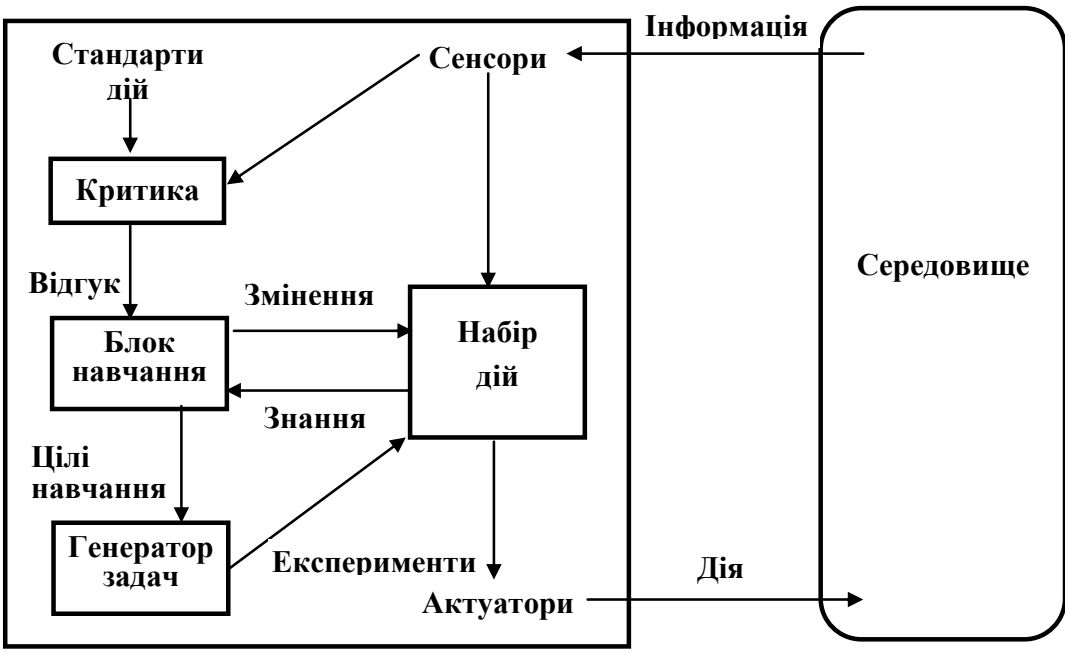

Рис. 7.3. Структура інтелектуального агента.

Від агентів вимагають здатність до навчання і навіть саме навчання [21, 22]. Здатність планувати свої дії ділить агентів на регулюючі та плануючі.

Плануючі властивості. Якщо вміння планувати не передбачено (регулюючий тип), то агент буде постійно переоцінювати ситуацію і відновлювати свій вплив на навколишнє середовище. Той, агент, який планує, може запланувати кілька дій на різні проміжки часу. При цьому агент може моделювати розвиток ситуації, що дає можливість більш адекватно реагувати на поточні ситуації. При цьому агент повинен брати до уваги не тільки свої дії і реакцію на них, а й зберігати моделі об'єктів і агентів навколишнього середовища для прогнозування їх можливих дій і реакцій.

7.2 Мультиагентні системи

Сукупність декількох агентів, що працюють спільно, а отже, мають властивістю спілкування, називаються багатоагентними або мультиагентними системами (МАС, *англ*. Multi-agent system).

Мультиагентні системи можуть бути використані для вирішення таких проблем, які складно або неможливо вирішити за допомогою одного агента або монолітної системи.

У мультиагентной системі необов'язково всі агенти взаємодіють (спілкуються) між собою. У крайньому випадку, спілкування немає взагалі. Такі системи назвемо дискретними мультиагентними системами. Другий крайній випадок - кожен агент спілкується з кожним. Таку систему назвемо повно мультиагентною системою.

Мультиагентна система, що діє як єдиний агент, повинна характеризуватися і деякої загальної для всіх субагентов метою і координацією дій по досягненню цієї мети.

Оскільки зустрічаються і інші ситуації, коли агенти не пов'язані настільки тісно, то такі системи можна назвати товариствами агентів. Відсутність єдиної мети, однак, не заперечує можливого групової поведінки агентів. Але воно є, скоріше, епізодичним, ніж систематичним.

Важливою відмінністю мультиагентної системи від програми або одного агента є те, що входять в систему програмні агенти (деякі), які були спроектовані спеціально для цієї системи.

Може бути, це - повторно використовувані агенти, або агенти, розроблені для вирішення більш універсальних завдань. У цих випадках агенти мають власні цілі, що не збігаються повністю з цілями системи (організації), але сумісні з ними. Тим не менш, вони можуть бути корисними один одному для вирішення поставлених перед ними завдань і, тому, дуже важливим для них з цієї точки зору є властивість комунікативності.

З організаційної точки зору існують спільні цілі всієї спільноти, і ці загальні цілі виражаються, перш за все, в ролях (які грають агенти) і нормах взаємодії.

Дослідження в області систем підтримки прийняття рішень (DSS – Decision Support System) в останні роки все більше переходять від створення систем у вигляді традиційного «ящика з інструментами» (toolbox) до парадигми співробітництва та інтеграції незалежних додатків. Швидко зростаюча область досліджень інтелектуальних агентів і мультиагентних систем пропонує можливості створення більш ефективних систем на основі єдиного підходу.

7.3. Агентні платформи

Агентна платформа – програмна оболонка, яка може створювати, інтерпретувати, запускати, переміщати і знищувати агенти [22].

Як "повітря" для агенту, агентна платформа забезпечує йому середовище для виконання. Також як і агент, агентна платформа асоціюється з повноваженнями, які визначають організацію чи персону, від імені яких працює система.

Агентна система однозначно ідентифікується ім'ям та адресою (рис. 7.4).

До складу платформи входять, як мінімум, одне місце та інтерфейс з'єднання – ***комунікаційна інфраструктура*** (КІ).

Місце забезпечує середовище виконання агентів на комп'ютері. У ньому можуть одночасно знаходитись декілька агентів. КІ реалізує службу зв'язку, службу імен та службу безпеки.

На одній машині можуть розміщуватись декілька агентних систем. Тип агентної системи описує сукупність параметрів агенту. Наприклад, якщо типом агентної системи є "аглети", то це означає, що агентна система створена компанією IBM, підтримує мову Java як мову реалізації агентів, та використовує Java Object Serialization для перетворення агентів в послідовну форму. Адміністратор мережевого регіону визначає служби комунікації для внутрішньорегіональних та міжрегіональних взаємозв'язків.

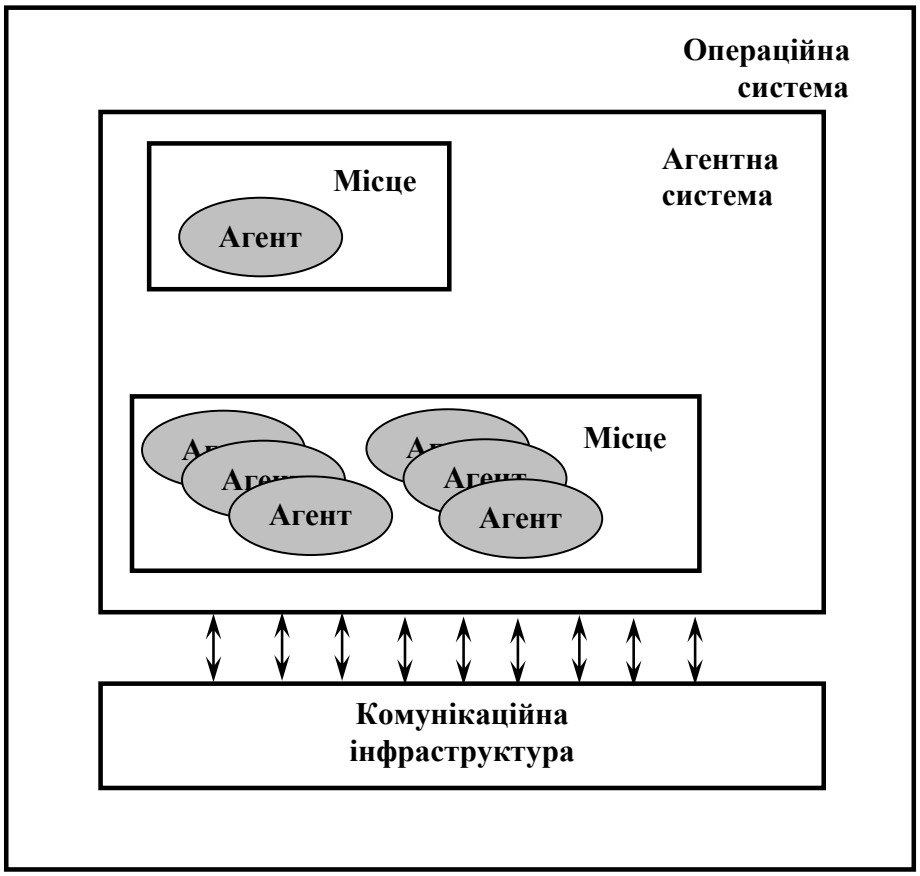

Рис. 7.4. Агентна система

Комунікаційна інфраструктура забезпечує транспортні служби зв'язку (наприклад TCP/IP), службу імен та службу безпеки для агентних систем.

У теперішній час існує декілька десятків МАС. Більшість із них (Gypsy, JADE, Ajanta, JATLite та ін.) розроблені в університетах з метою дослідження цієї технології. Деякі системи (такі як ASDK, Xelopes, JAFMAS та ін.) існують на рівні бібліотек, надаючи програмісту тільки базові класи для реалізації основних компонентів: агентів, платформ, механізмів взаємозв'язку та безпеки. На їх основі розробляються самостійні системи, наприклад, MagNet, E-Commercia. За останній час з'являються й комерційні системи, такі як Gossib фірми Tryllian, Bee-gent та Plangeny корпорації Toshiba. На жаль, документація до них недоступна.

7.4. Безпека в системах мобільних агентів

Агентні розподілені обчислювальні системи знайшли певну нішу у сфері РВС, але їх повсюдному поширенню перешкоджає сам принцип їх роботи: навряд чи розсудливий адміністратор обчислювальної мережі дозволить мандрувати комп'ютерами користувачів автономним інтелектуальним постійно змінюваним самонавчальним програмним системам, які можуть мігрувати та виконуватись на будь-якому обчислювальному вузлі. У зв'язку з цим виникає цілий ряд серйозних проблем, пов'язаних з безпекою агентних платформ.

З однієї сторони, агенти можуть нести приватну інформацію для забезпечення власної роботи. Наприклад, агент у системі електронної комерції може містити номер кредитної картки й паспортні дані користувача для того, щоб від його імені здійснювати операції.

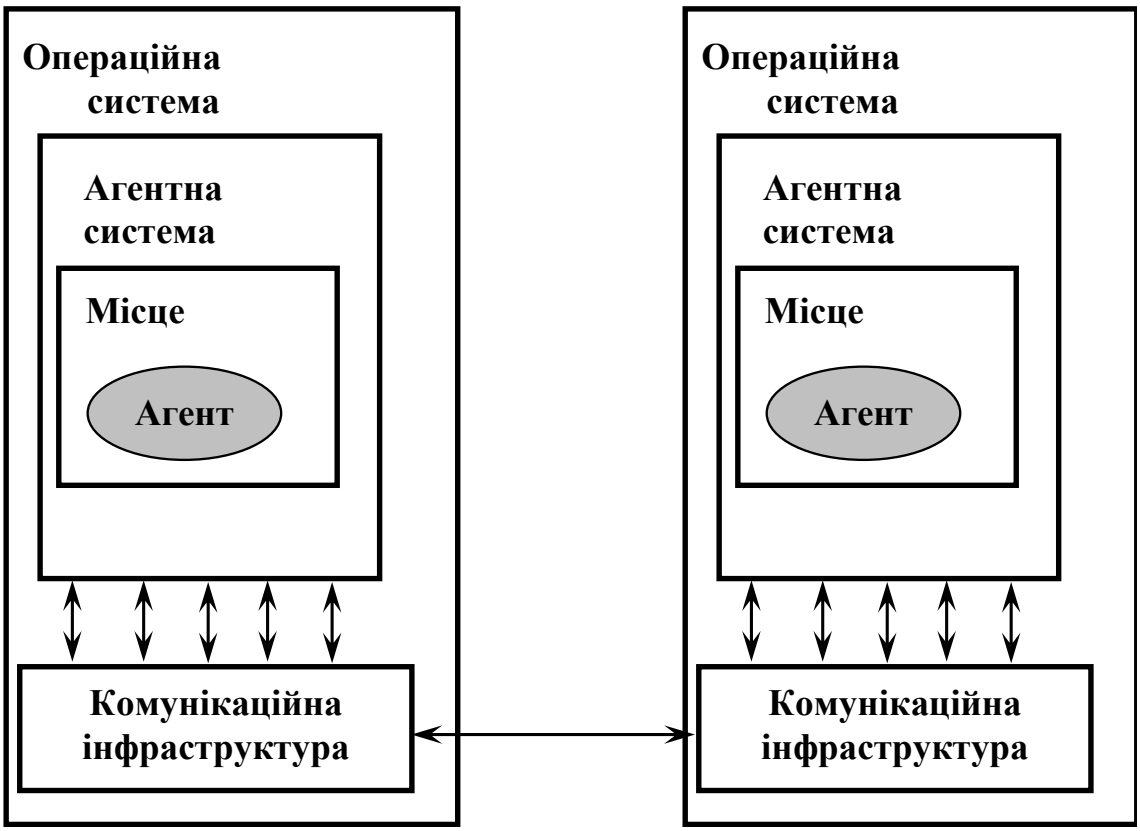

Рис. 7.5. Зв’язки між агентними системами.

Відповідно, необхідно, щоб середовище забезпечувало безпечне для агенту середовище виконання.

З іншого боку, сторонній агент може спробувати атакувати базове середовище та витягти дані чи роздобути інші ресурси. В цьому випадку агентна платформа повинна бути достатньо добре захищена та мати можливість протистояти таким атакам.

Виділяють наступні можливі проблеми безпеки при роботі агентних платформ:

1. Агент атакує Хост: агент може вкрасти або модифікувати дані Хосту.
2. Хост атакує Агента: Хост може вкрасти або модифікувати дані Агента, змінити його стан або код.
3. Зловмисний агент атакує іншого агента.
4. Атака іншими елементами.

Варіант атаки «Агент атакує Хост» є стандартною атакою, в якій код, отриманий з не надійного джерела, намагається отримати повний доступ до системи або ж перешкодити нормальному виконанню задач, підвищив свої повноваження на виконання. В цьому випадку від атаки допомагають традиційні засоби захисту, як ці: контроль рівня доступу; пісочниці; аутентифікація; криптографія.

Також, існують специфічні для агентних платформ методи забезпечення безпеки, наприклад, аналіз історії руху агенту. В цьому випадку платформа може дізнатись про історію пересування агента з логу й зробити висновок про його якість на основі інформації про те, на яких платформах він побував до цього.

Варіант атаки «Хост атакує агент» більш складний в забезпеченні безпеки. В таких випадках традиційні засоби не працюють, так як Хост повинен мати повну інформацію про код агента для його виконання. В цьому випадку можуть бути застосовані наступні засоби:

1) Мобільна криптографія: функції і дані агенту шифруються таким чином, що Хост не може розібрати, яким чином функції працюють й вилучити код. Недоліком даного методу є необхідність пошуку схеми шифрування для довільних функцій, а також необхідність перенесення ключа кадрування.

2) Безпечне переміщення: міграція тільки на певні (довірені) хости.

3) Використання фіктивних даних: в базі даних, системи, аналізуючій роботу агентів, зберігається набор фіктивних даних, які не змінюються під час нормальної роботи агента.

4) Використання довіреного апаратного забезпечення: це можуть бути смарткартки, інтегровані мікросхеми й інше.

Таким чином, основна ідея використання агентної технології в області аналізу даних полягає в тому, щоб інкапсулювати в агенті цикл вилучення знань із даних. В цьому випадку агент постає у ролі аналітика. Для вирішення задач інтелектуального аналізу даних реалізовані й успішно застосовуються різні типи мобільних агентів, використовуючи різноманітні алгоритми аналізу.

Література

Основна

1. Ситник В.Ф. Системи підтримки прийняття рішень: [навч. посіб.] / Ситник В.Ф.– К.: КНЕУ, 2004. – 614 с.
2. Глибовець М.М., Олецький О.В. Штучний інтелект. Підручник для студентів вищих навчальних закладів, які навчаються за спеціальностями "Комп'ютерні науки" та "Прикладна математика". — К.:Вид.дім "КМ Академія", 2002. — 369 с.
3. Литвин. В.В. Інтелектуальні системи [Текст] : підручник / В. В. Литвин, В. В. Пасічник, Ю. В. Яцишин. – Львів : Новий Світ-2000, 2009. - 406 с.
4. Іванченко Г. Ф. Системи штучного інтелекту : навч. посібник / Г.Ф. Іванченко. – К., 2011. – 382 с.
5. Кузьменко Б.В. Системи штучного інтелекту : Навч.посібник / Б. В. Кузьменко, О. А. Чайковська. – К. : Альтерпрес, 2006. – 140 с.
6. Mitchell M. An Introduction to Genetic Algorithms. MIT Press, 1996. - 221 p.
7. Bäck, T. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press, Inc. 1996. - 314 p.
8. Amir Husain. The Sentient Machine: The Coming Age of Artificial Intelligence. Scribner, 2017. - 224 p.
9. Ahmed S., Karsiti M.N. (eds.) Multiagent Systems. InTech, 2009. – 434 p.
10. Alkhateeb F., Al Maghayreh E., Abu Doush I. (eds.) Multi-Agent Systems – Modeling, Interactions, Simulations and Case Studies. InTech, 2011. – 512 p.
11. Mitchell, Tom. Machine learning. McGraw-Hill, 1997. – 432 p.
12. Bishop, C.M. Neural networks for pattern recognition. Oxford, England: Oxford University Press, 1996. - 498 p.
13. The Oxford Handbook of Cognitive Neuroscience. Volume 1: Core Topics. Edited by Kevin N Ochsner and Stephen M Kosslyn. Oxford University Press, 2016. - 640 p.
14. Patricia S. Churchland and Terrence J. Sejnowski. The Computational Brain. Springer, 1992. - 558 pp.
15. LiMin Fu. Neural networks in computer intelligence.New York McGraw-Hill 1994. - 460 p.
16. Robert Hecht-Nielsen. Neurocomputing. Addison-Wesley, 1990. - 433 p.
17. John MacKrell. Supporting Collaborative Product Definition via Scaleable, Web-Based PDM.- Prepared by CIMdata, Inc., 2000. – 233 с.
18. Zbignlew, M. Genetic algorithms + data structures = evolution programs. Berlin: SpringerVerlag. 1992. - 327 p.

Допоміжна

1. Uziel Sandler, Lev Tsitolovsky. Neural Cell Behavior and Fuzzy Logic. - Springer, 2008.– 478 p.

2. Whitley, L. D., & Vose, M. D. (Eds.). Foundations of genetic algorithms 3. Morgan Kaufmann. 1995. — 396 p.

3.Wright, S. Evolution and the genetics of populations. Vol. 4: Variability within and among. Berlin: SpringerVerlag. 1996. — 372 p.

4. Natural Populations. Chicago: University of Chicago Press. 1977. — 437 p.

Укладачі:
І.М. Удовик, Г.М. Коротенко, Л.М. Коротенко, В.О. Трусов, А.Т. Харь

Методи та системи штучного інтелекту

Навчальний посібник

для студентів спеціальності
122 «Комп'ютерні науки»
(галузь знань 12 Інформаційні технології)

Редакційно-видавничий комплекс

Підписано до друку _____. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк..
Обл.-вид. арк.. Тираж прим. Зам. № .

Державний вищий навчальний заклад
"Національний гірничий університет"
49005, м. Дніпро, пр. Д. Яворницького, 19