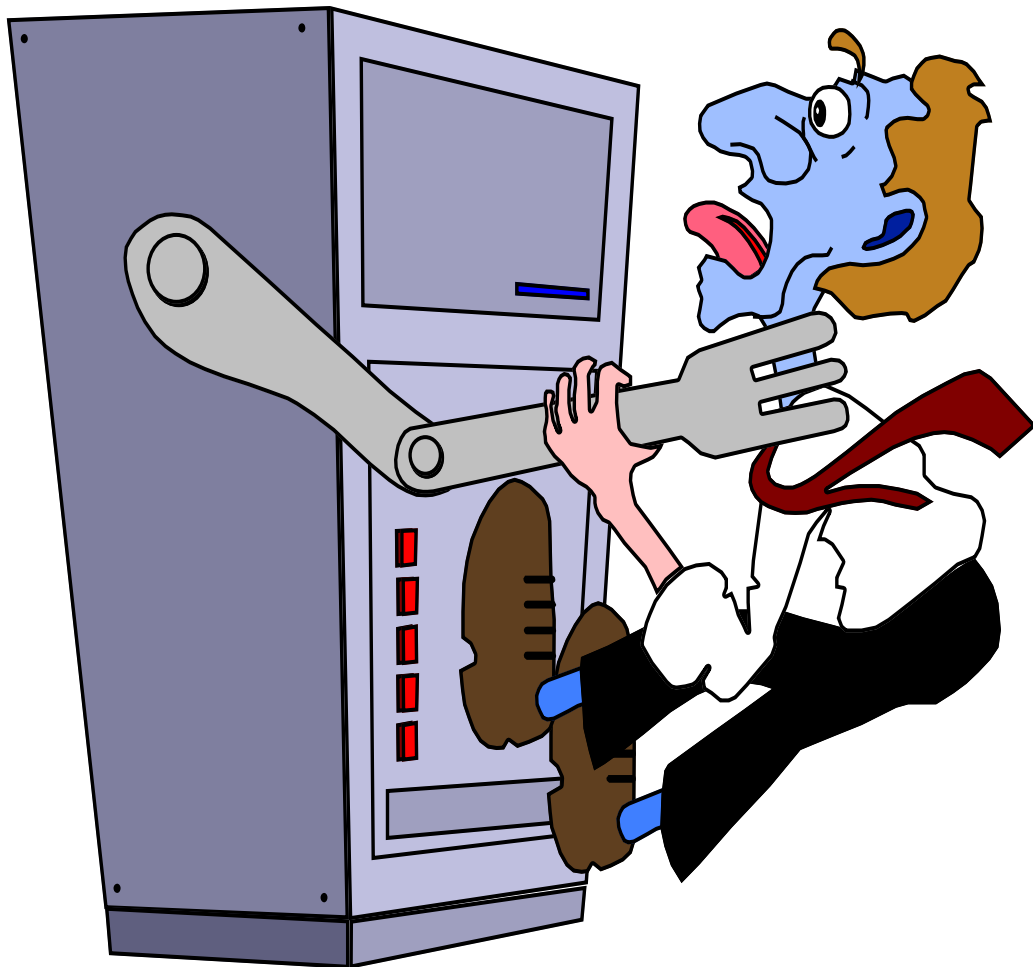


Глибовець М.М., Олецький О.В.
Системи штучного інтелекту

Навчальний посібник



ЗМІСТ

ЧАСТИНА I. НЕФОРМАЛЬНИЙ ВСТУП ДО ПРОБЛЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 1. Базові концепції

- 1.1. Інтуїтивне розуміння поняття “інтелект”
- 1.2. Деякі інтелектуальні задачі
- 1.3. Деякі визначення та їх критика
- 1.4. Співставлення зі зразком та його застосування для організації фатичного діалогу
Запитання і вправи до розділу

Розділ 2. Загальна характеристика інтелектуальних систем

- 2.1. Характеристика алгоритмічного та декларативного підходів
- 2.2. Квазіалгоритми та основні джерела квазіалгоритмічності
- 2.3. Характеристика інтелектуальних систем з загальнокібернетичних позицій
- 2.4. Класифікація основних напрямків досліджень
Запитання і вправи до розділу

ЧАСТИНА II. ОСНОВНІ КЛАСИЧНІ ПІДХОДИ ДО ВИРІШЕННЯ ТИПОВИХ ІНТЕЛЕКТУАЛЬНИХ ЗАДАЧ

Розділ 3. Основні підходи до планування цілеспрямованих дій

- 3.1. Перші спроби. “Загальний вирішувач задач”
- 3.2. Прийняття рішень як оптимізаційна задача
- 3.3. Евристичний пошук
- 3.4. Основні алгоритми пошуку на дереві
- 3.5. “Жадібні” алгоритми.
- 3.6. І-АБО-графи
- 3.7. Простір задач і простір станів
- 3.8. Обмежуючі правила як засіб скорочення перебору
Запитання і вправи до розділу

Розділ 4. Методи програмування ігрових задач

- 4.1. Ігрові задачі
- 4.2. Дерево гри та мінімаксна процедура.
- 4.3. Мінімаксна процедура на усікненому дереві. Статичні оціночні функції.
- 4.4. Альфа-бета-відтинання.
- 4.5. Огляд сучасних шахових програм.
- 4.6. Деякі підходи до самонавчання ігрових програм.
- 4.7. Підхід людини до шахової гри.
Запитання і вправи до розділу.

Розділ 5. Конекціоністські методи

- 5.1. Загальна характеристика конекціоністського підходу та його місце в теорії інтелектуальних систем
- 5.2. Модельні нейрони як порогові елементи
- 5.3. Пандемоніум Селфріджа

5.4. Перцептрон

5.5. Сучасні нейронні мережі.

Запитання і вправи до розділу.

ЧАСТИНА III. ПРЕДСТАВЛЕННЯ ЗНАНЬ ТА МЕТОДИ ЛОГІЧНОГО ВИВЕДЕННЯ.

Розділ 6. Знання як інформаційна основа інтелектуальних систем

6.1. Інтуїтивне поняття знань.

6.2. Перехід від даних до знань

6.3. Властивості та класифікація знань.

6.4. Загальна характеристика моделей задання знань

Запитання і вправи до розділу.

Розділ 7. Мережні та фреймові моделі знань

7.1. Семантичні мережі.

7.2. Фрейми

7.3. Зв'язок між семантичними мережами та фреймами

Запитання і вправи до розділу.

Розділ 8. Логічні моделі та метод резолюцій.

8.1. Логічні моделі.

8.2. Автоматичне доведення теорем та принцип резолюцій

8.3. Поняття про логічне програмування

Запитання і вправи до розділу.

Розділ 9. Продукційні моделі

9.1. Загальна характеристика продукційних моделей

9.2. Випадок нечітких продукційних правил

Запитання і вправи до розділу.

Розділ 10. Формалізація нечітких, неточних та недостовірних знань

10.1. Нечіткі множини

10.2. Основні операції над нечіткими множинами

10.3. Нечітке логічне виведення. Метод центру тяжіння композиції “максимум-мінімум”

10.4. Багатозначні і модальні логіки

10.5. Базові поняття теорії індуктивних висновків

10.6. Правила Мілля формування гіпотез

10.7. Індуктивна перевірка гіпотез і парадокс Хемпеля

Запитання і вправи до розділу.

ЧАСТИНА IV. ЗАГАЛЬНІ ПІДХОДИ ДО ПОБУДОВИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 11. Загальноінтелектуальні метапроцедури

Запитання і вправи до розділу

Розділ 12. Ситуаційне керування

12.1. Базові концепції

12.2. Семіотичні моделі та мови опису ситуацій

Запитання і вправи до розділу

Розділ 13. Моделі і методи навчання і самонавчання

13.1. Базові визначення

- 13.2. Автомати з лінійною тактикою.
- 13.3. Формування та засвоєння понять.

Запитання і вправи до розділу

ЧАСТИНА V. ДЕЯКІ ЗАСТОСУВАННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 14. Інтелектуальні інформаційні системи

- 14.1. Проблема розуміння.
- 14.2. Системи обробки природної мови.
- 14.3. Етапи аналізу речень природною мовою.
- 14.4. Методи реалізації діалогу.
- 14.5. Загальна структура експертних систем.
- 14.6. Синтез зв'язних текстів.
- 14.7. Загальна характеристика систем машинного перекладу

Запитання і вправи до розділу

Розділ 15. Моделі і методи розпізнавання зразків

- 15.1. Загальна характеристика
 - 15.2. Статистичні методи розпізнавання.
 - 15.3. Структурно-синтаксичні методи розпізнавання.
- Задачі і вправи до розділу

Література

ЧАСТИНА I. НЕФОРМАЛЬНИЙ ВСТУП ДО ПРОБЛЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 1. Базові концепції

*Я не можу сказати, що це таке, але завжди впізнаю, коли побачу
(Д.Мічі, Р.Джонстон. “Комп’ютер-творець”)*

1.1. Інтуїтивне розуміння поняття “інтелект”.

Загальновідомо, що інтелектуалізація інформаційних технологій є однією з основних тенденцій розвитку сучасної комп’ютерної техніки. Що таке інтелектуалізація та тісно пов’язане з нею поняття “штучний інтелект”?

Можна стверджувати, що “штучний” інтелект у тому чи іншому розумінні повинен наближатися до інтелекту природного і у ряді випадків використовуватися замість нього; так само, як, наприклад, штучні нирки працюють замість природних. Чим більше буде ситуацій, у яких штучні інтелектуальні системи зможуть замінити людей, тим більш інтелектуальними будуть вважатися ці системи.

Навряд чи є сенс протиставляти поняття штучного інтелекту і інтелекту взагалі. Тому слід спробувати визначити, що слід вважати інтелектом, незалежно від його походження.

Людина вважається інтелектуальною “від природи”, і цей інтелект був вироблений на протязі мільйонів років еволюції. Людина вміє вирішувати багато інтелектуальних задач. Кожна людина вважає, що вона розуміє значення слова “інтелект”. Водночас, якщо попросити пересічну людину або спеціаліста дати визначення цього слова, чіткої відповіді не буде. І дійсно, дати чітке визначення поняття інтелекту, очевидно, неможливо. Далі будуть проаналізовані деякі спроби визначення цього поняття та буде дана їх критика.

Але відсутність чіткого визначення не заважає оцінювати інтелектуальність на інтуїтивному рівні. Можна навести як мінімум два методи такої оцінки:

■ **метод експертних оцінок.** Рішення про ступінь інтелектуальності приймає досить велика група експертів (незалежно або у взаємодії між собою);

■ **метод тестування.** Існує значна кількість так званих інтелектуальних тестів, апробованих практикою, і ці тести широко використовуються для оцінки рівня розумових здібностей людини, а також у психології та психіатрії. З деякими з подібних тестів можна ознайомитися, наприклад, в книзі Г.Айзенка.

Для прикладу наведемо декілька типових тестових завдань.

Приклад 1. Вставте слово, яке означає те саме, що і два слова поза дужками:

дерево (...) підробка

Приклад 2. Вставте число, яке пропущене:

36 30 24 18 _

Приклад 3. Викресліть зайве слово:

лев лисиця жираф шука собака

І метод експертних оцінок, і метод тестування мають свої недоліки. Головний з них полягає у тому, що оцінка дається виходячи лише з власних уявлень експертів, авторів тестів і т.п. про те, як має бути. Тому ці способи не дуже придатні для оцінки будь-якого інтелекту, крім людського.

Серед психіатрів інколи можна почути вислів: “він міркує логічно вірно, але неправильно”. Наприклад, дається тестове завдання: “серед слів **соловей, чапля, перепілка, стілець, шпак** виділити зайве”.

Більшість людей, не задумуючись, дає відповідь **стілець**, тому що всі інші слова - це назви птахів. І раптом хтось дає відповідь **шпак**, пояснюючи це тим, що це єдине слово, в якому відсутня літера “л”.

Обидві класифікації є логічно вірними і формально рівноправними. Але чому ж перевага віддається одній з них? Тому, що так міркує більшість людей. А чому так міркує більшість людей? Очевидно, тому, що **перша класифікація вважається більш важливою для людської практики**. Це положення приймається на аксіоматичному рівні, без доведення (а хіба можна запропонувати щось інше?) Але те, що є більш важливим для людської практики, зовсім не обов’язково буде так само важливим для розумного робота або для інопланетянина.

Необхідно підкреслити, що поняття “штучний інтелект” не можна зводити лише до створення пристроїв, які імітують людину в усій повноті її діяльності. Насправді ж, спеціалісти які працюють в цій області вирішують іншу задачу: **виявити механізми, які лежать в основі діяльності людини, щоб застосувати їх при вирішенні конкретних науково-технічних задач**. І це лише одна з можливих проблем.

1.2. Деякі інтелектуальні задачі

Можна описати загальні риси будь-якої інтелектуальної системи, охарактеризувавши її з загальнокібернетичних позицій. Але спочатку видається доцільно розглянути деякі задачі, які доводиться вирішувати людському розумові, та дати їх аналіз.

Перша з них - **розпізнавання зразків**, або просто розпізнавання. На інтуїтивному рівні можна сформулювати декілька типових задач розпізнавання:

- ідентифікувати об’єкт, що спостерігається людиною, тобто вирізнити його серед інших (наприклад, побачивши іншу людину, впізнати у ній свою дружину);
- здійснити розпізнавання у класичній постановці, тобто віднести об’єкт, що спостерігається людиною, до одного з задалегідь відомих класів об’єктів (наприклад, відрізнити легковий автомобіль від вантажного).

Людина робить класифікацію просто. Чоловік, повернувшись додому, відразу ж пізнає свою жінку, дитя, собаку і т.д. Але він дуже рідко може **пояснити, як** він це робить.

Якби це можна було пояснити, алгоритми розпізнавання можна було б легко програмувати і широко застосовувати. Але раціонального пояснення, як правило, немає. Теорія розпізнавання, яка інтенсивно розвивається, необхідна для того, щоб навчити вирішувати задачі розпізнавання і штучні інтелектуальні системи. Зокрема, сформульовано такий ключовий принцип: **будь-який об’єкт у природі - унікальний; унікальні об’єкти - типізовані**. У відповідності до цього принципу, розпізнавання здійснюється на основі аналізу певних характерних **ознак**. Вважається, що в природі не існує двох об’єктів, для яких співпадають **абсолютно всі** ознаки, і це теоретично дозволяє здійснювати ідентифікацію. Якщо ж для

деяких об'єктів співпадають **деякі** ознаки, ці об'єкти теоретично можна об'єднувати в групи, або класи, за цими співпадаючими ознаками.

Проблема полягає у тому, що різноманітних ознак існує незліченна кількість. Незважаючи на легкість, з якою людина проводить розпізнавання, вона дуже рідко в змозі виділити ознаки, суттєві для цього. До того ж, об'єкти, як правило, змінюються з часом.

Ми далі побачимо, що розпізнавання об'єктів і ситуацій має виняткове значення для орієнтації людини в навколишньому світі і для прийняття правильних рішень. Розпізнавання, як правило, здійснюється людиною на інтуїтивному, підсвідомому рівні, людина навчилася цьому за мільйони років еволюції. На цьому фоні спроби навчити розпізнаванню складних об'єктів штучні інтелектуальні системи, навіть шляхом автоматизованого виділення характерних ознак, мають досить слабкі досягнення.

Інша задача - **логічне мислення**, перехід від початкових положень до їх наслідків за формалізованими законами логіки. І тут пересічна людина рідко в змозі пояснити, за якими алгоритмами вона здійснює логічні побудови. Але методики та алгоритми, за якими **можна** автоматизувати логічне мислення, досить відомі.

Перш за все, це формальна логіка Аристотеля на основі конструкцій, які отримали назву **силогізмів**. Вони практично неподільно панували в логіці аж до початку XIX століття, тобто до появи булевої алгебри.

Немає ніякої необхідності давати якісь формальні визначення силогізмів. Наведемо лише один класичний приклад.

Перше положення. *Усі люди смертні.*

Друге положення. *Сократ - людина.*

Висновок: *Сократ смертний.*

Якщо перше та друге положення у силогізмі істинні та задовольняють певним загальним формальним вимогам, тоді і висновок буде істинним незалежно від змісту тверджень, що входять до силогізму.

Виконання формальних вимог є важливим, інакше легко припуститися логічних помилок, подібних до таких:

Всі студенти вузу А знають англійську мову

Петров знає англійську мову.

Отже, Петров - студент вузу А.

Або:

Іванов не готувався до іспиту і отримав двійку.

Сидоров не готується до іспиту.

Отже, і Сидоров отримає двійку.

Аристотелем було запропоновано декілька формальних конструкцій силогізмів, які він вважав достатньо універсальними. Лише у XIX столітті почала розвиватися сучасна математична логіка, яка розглядає аристотелевські силогізми як один із часткових випадків.

Основою більшості сучасних систем, призначених для автоматизації логічних побудов, є **метод резолюцій** Робінсона. Він буде описаний у відповідному параграфі даного підручника. Але практична автоматизація логічного мислення зіткнулося з двома серйозними проблемами.

Перша з них - феномен, який Річард Беллман називав **прокляттям розмірності**. Зовнішній світ являє собою винятково складне переплетіння різноманітних об'єктів та зв'язків між ними. Для того, щоб тільки ввести всю цю інформацію до пам'яті інтелектуального комп'ютера, може знадобитися не одна тисяча років. Ще більше років буде потрібно, щоб врахувати всі необхідні факти при логічному виведенні.

Інша проблема - **алгоритмічна нерозв'язність**. Добре відомо, що в рамках будь-якої досить складної формальної теорії існують положення, які є істинними, але які не можна ні довести, ні спростувати (теореми Геделя і Чорча про нерозв'язність формальної арифметики та числення предикатів).

Реальні програми, які здійснюють логічне виведення (вони часто називаються **експертними системами**) мають досить обмежене застосування. Вони мають обмежений набір фактів та правил з певної, більш-менш чітко окресленої предметної галузі і можуть використовуватися лише у цій галузі.

Що ж стосується людини, то якість її логічного мислення часто буває далеким від бездоганного. Люди часто роблять логічні помилки, а інколи взагалі керуються принципами, невірними з точки зору нормальної логіки. У цьому зв'язку можна згадати одну з жартівливих відповідей на запитання "чим відрізняється людина від тварини?": "людина здатна до нерозумних вчинків".

Людина досить рідко проводить логічні побудови до кінця. Дуже часто свідоме логічне виведення на певному етапі обривається, і рішення знову-ж таки приймається на підсвідомому, інтуїтивному рівні. Зрозуміло, що таке рішення може бути помилковим. Але, якби це було не так, людина була б практично не здатною ні до якої діяльності - ні до фізичної, ні до продуктивної розумової.

Для задач, які розглядалися в цьому розділі, характерною була спільна риса: їх **погана формалізованість**, відсутність або невизначеність чітких алгоритмів вирішення. Саме такі задачі і являють собою основний предмет розгляду в теорії штучного інтелекту.

До зовсім іншого класу відносяться **обчислювальні задачі**.

В принципі, важко відповісти на запитання, як саме людина здійснює ті чи інші обчислення. Добре відомими є і низька швидкість, і невисока надійність цього виду людської діяльності. Але були запропоновані ефективні принципи комп'ютерних обчислень, які радикально відрізняються від тих, які застосовуються людиною, і ці добре формалізовані, алгоритмічні методики забезпечили рівень вирішення обчислювальних задач, абсолютно недосяжний для людського інтелекту. Водночас цей високий рівень алгоритмізації значною мірою зумовив слабкість традиційних комп'ютерних систем при вирішенні тих інтелектуальних задач, з якими людина справляється непогано.

1.3. Деякі визначення та їх критика

Було зроблено чимало спроб дати формальне визначення інтелекту взагалі і інтелекту штучного зокрема. Найбільш відомим, очевидно, є визначення предмету

теорії штучного інтелекту, що було дане видатним дослідником у галузі штучного інтелекту Марвіном Мінським і яке у більш або менш видозміненому вигляді потрапило до багатьох словників та енциклопедій: *“штучний інтелект є дисципліна, що вивчає можливість створення програм для вирішення задач, які при вирішенні їх людиною потребують певних інтелектуальних зусиль”*. Це визначення зустріло критику, яка полягала в тому, що під нього можна підвести що завгодно, наприклад, виконання простих арифметичних операцій (а з іншого боку, чому б і ні?). Відтак у ряді книг та енциклопедій до цього визначення додається поправка: *“сюди не входять задачі, для яких відома процедура їх вирішення”*. Важко вважати таке визначення задовільним. Розвиваючи цю думку далі, можна було б продовжити: отже, якщо я не знаю, як виконувати деяку задачу, то вона є інтелектуальною, а якщо знаю - то ні. Наступний крок - це відомі слова Л.Теслера: *“Штучний інтелект - це те, чого ще не зроблено”*. Цей крайній та парадоксальний висновок лише підкреслює дискусійність проблеми.

Є деякі більш конструктивні визначення інтелекту. Наприклад, в [3] наводиться одне з них: *“інтелект є здатність правильно реагувати на нову ситуацію”*. Там же наводиться і критика цього визначення: не завжди зрозуміло, що слід вважати новою ситуацією. Уявіть собі, наприклад, звичайний калькулятор. Цілком імовірно, що на ньому ніколи не обчислювали суму двох нулів. Тоді завдання “додати нуль плюс нуль” можна вважати ситуацією, новою для калькулятора. Безумовно, він з нею впорається (“правильно відреагує на нову ситуацію”), але чи можна на цій підставі вважати його інтелектуальною системою?

Відомий англійський учений Алан Тьюринг сформулював тезу, спрямовану на визначення моменту, з якого машину можна вважати інтелектуальною. Нехай експерт за допомогою телефону, телетайпу і т.ін. спілкується з кимось (або чимось), що може бути як людиною, так і машиною. Експерт дає будь-які тести-завдання. Він повинен визначити, з ким він має справу - з людиною чи з ЕОМ. Якщо він приймає комп'ютер за людину, то комп'ютер може вважатися інтелектуальним.

Така перевірка дістала назву **тесту Тьюринга**. Багато спеціалістів вважали, що тест Тьюринга є цілком задовільним для визначення рівня інтелектуальності комп'ютерної системи. Але виявилось, що це не зовсім так.

В основі тесту Тьюринга лежить неявне припущення про те, що необхідною умовою ведення діалогу є розуміння співрозмовника. Але у кінці 60-х рр. американський кібернетик Дж. Вейценбаум створив дві програми - ЕЛІЗА і ДОКТОР. Вони були створені для використання в психіатрії, але виявилось, що вони можуть обманути експерта в умовах тесту Тьюринга. В основі лежить ідея **фатичного діалогу**.

Визначення. *Фатичним діалогом називається діалог без розуміння співрозмовника, на основі формального перефразування почутого.*

Так, наприклад, на висловлювання пацієнта *“Ви не відверті зі мною”* програма може відповісти *“Чому Ви думаєте, що я не відверта з Вами?”* У ряді випадків програма може видати заздалегідь заготовлену фразу. Так, на слова пацієнта *“Мій батько мене не любив”* вона може відповісти *“Розкажіть мені про Вашу сім'ю”*.

Звичайно, це не має ніякого відношення ні до розуміння, ні до інтелекту, але багато експертів, які проводили з цими програмами тест Тьюринга, вирішили, що мають справу з людиною.

В [2] наводиться курйозний приклад фатичного діалогу. Віце-президент великої американської компанії захотів зв'язатись з відомим системним програмістом Д. Бобровим. Відбулась така телетайпна розмова:

Віце-президент : *Чи можу я зараз скористатись комп'ютером?*

Бобров: *А чому ви питаєте?*

Віце-президент: *Може, мені вдасться підписати контракт.*

Бобров: *Чому ви в цьому впевнені?*

Віце-президент: *Мої потенційні покупці ніколи не бачили систему в дії.*

Бобров: *Це має для вас велике значення?*

Віце-президент: *Звичайно.*

Бобров: *Ви маєте в цьому сумнів?*

Віце-президент: *Що за безглузде зауваження?*

Бобров: *Що Ви маєте на увазі, кажучи про безглузде зауваження?*

Віце-президент: *Чи можу я зараз скористатись комп'ютером?*

Бобров: *Розкажіть мені ще про Ваших потенційних покупців.*

Віце-президент: *Передзвоніть мені 491-1850.*

Віце-президент був, м'яко кажучи, здивований, тим більше, що дзвінка не було. Непорозуміння з'ясувалося пізніше. Виявилось, що телетайп з'єднав віце-президента не з самим Бобровим, а з програмою, яка реалізувала фатичний діалог.

Звичайно, подібний діалог важко вважати особливо осмисленим, але хто може дати чітке визначення рівня осмисленості?!

Фатичний діалог може бути і менш примітивним, але системи, які здатні тільки на підтримку такого діалогу, не можна вважати інтелектуальними. **Розуміння співрозмовника** є абсолютно необхідним для нормального діалогу, а для такого розуміння **необхідно мати певну суму знань про світ**.

Поява систем, що реалізують фатичний діалог, завдала сильного удару по визначенню Тьюринга. Безумовно, воно зберігає своє методологічне значення, але з'ясувалося, що воно не може претендувати на універсальність (так само, як і будь-яке інше з відомих визначень).

У наступному параграфі розглядається класичний метод написання програм, що реалізують фатичний діалог - застосування **співставлення зі зразком**.

1.4. Співставлення зі зразком та його застосування для організації фатичного діалогу

У цьому параграфі описується простий метод, що дозволяє написати псевдоінтелектуальну програму, яка може підтримувати фатичний діалог з людиною або з іншою аналогічною програмою. Параграф написано на основі [4].

Розглядається "анатомія" програми **ДОКТОР** у дещо спрощеному вигляді. Діалог з цією програмою може бути, наприклад, таким:

ДОКТОР: *Я Вас слухаю.*

ЛЮДИНА: *Мене бентежить погода.*

ДОКТОР: *Як давно Вас бентежить погода?*

ЛЮДИНА: *Відтоді, як померла моя мати.*

ДОКТОР: *Розкажіть про Вашу сім'ю.*

ЛЮДИНА: *Всі вони захоплюються комп'ютерами.*

ДОКТОР: *Машини Вас лякають.*

ЛЮДИНА: *Ні.*

ДОКТОР: *Будь ласка, не будьте так небагатослівні...*

І так далі до нескінченності.

Що необхідно для реалізації такого діалогу? В основі методу лежить **співставлення** речень, які вводяться людиною, зі **зразками**, що зберігаються програмою. В залежності від результату співставлення реалізується та чи інша заздалегідь запрограмована операція. І самих зразків, і операцій, що їм відповідають, як правило, порівняно небагато.

Елементарне співставлення зі зразком може бути реалізоване такою ліспівською процедурою:

```
(DEFUN (MATCH P D)
(COND ((AND (NULL P) (NULL D)) T)
(( OR (NULL P) (NULL D)) NIL)
(( OR (EQUAL (CAR P) '?)
(EQUAL (CAR P) (CAR D)))
(MATCH (CDR P) (CDR D)))
((EQUAL (CAR P) '*))
(COND ((MATCH (CDR P) D))
((MATCH (CDR P) (CDR D)))
((MATCH P (CDR D)))))).
```

Насправді можна стверджувати, що співставлення зі зразком є однією з найбільш фундаментальних загальноінтелектуальних процедур. Спробуємо формалізувати це поняття.

Визначення. Нехай **M** - набір зразків, що зберігаються в пам'яті інтелектуальної системи, **F**- деяка множина функцій або алгоритмів, і **r(a, b, f)** - деякий критерій якості. Тоді під співставленням елемента **q**, що спостерігається інтелектуальною системою, зі зразком будемо розуміти знаходження елементу **m** $\in M$ і функції **f** $\in F$, таких, що **f(q)=m** або **f(m)=q**.

Природно також накласти одну з двох умов:

- максимізація критерію якості: **r(m, q, f) $\rightarrow$ max**;
- досягнення задовільного рівня якості: **r(m, q, f) > ϵ** , ϵ - заданий поріг.

Ця постановка задачі співставлення є дуже загальною. Її конкретизації залежать від того, які обмеження накладаються на множини **M** і **F** та на критерій якості. Можна вважати, що **більш інтелектуальна система повинна вміти здійснювати більш складне співставлення**. Зокрема, описаний раніше метод реалізації фатичного діалогу передбачає дуже прості процедури співставлення.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Охарактеризуйте відомі Вам методи оцінки інтелектуальних здібностей. Що між ними спільного та в чому полягає різниця?
2. У чому, на Вашу думку, полягають основні проблеми застосування інтелектуальних тестів для оцінки штучного інтелекту?

3. Назвіть декілька відомих Вам інтелектуальних задач; опишіть їх спільні риси та відмінності.
4. Чи зводиться процес отримання нових знань, зокрема, процес виведення нових фактів з уже існуючих, до чисто логічного виведення?
5. Охарактеризуйте роль інтуїції в розумовій діяльності людини.
6. Назвіть декілька відомих Вам визначень поняття інтелекту, дайте їх критику.
7. Опишіть процедуру тесту Тьюринга.
8. Охарактеризуйте поняття фатичного діалогу.
9. Дайте визначення задачі співставлення зі зразком.
10. Переформулюйте постановку задачі і опишіть процедуру співставлення зі зразком для реалізації фатичного діалогу у термінах загального визначення.
11. Напишіть програму на Ліспі, яка в змозі вести фатичний діалог методом співставлення зі зразком. Фрази людини повинні вводитися з клавіатури, а відповіді людини - виводитися на екран.

Розділ 2. Загальна характеристика інтелектуальних систем

*Солдат повинен бути розумним, винахідливим і кмітливим
(З Статуту Збройних Сил СРСР)*

2.1. Характеристика алгоритмічного і декларативного підходів

На найбільш загальному рівні, можна виділити два підходи до керування складними системами та до програмування роботів і комп'ютерів, що могли б розв'язувати ті чи інші задачі. При програмуванні сучасних комп'ютерів в основному реалізований традиційний **алгоритмічний підхід**, який можна ще назвати імперативним. Цей підхід вимагає заздалегідь продумати та детально розписати, як треба вирішувати певну проблему. Написання програми вимагає задання чітких послідовностей інструкцій. Якщо таку послідовність вдається написати - комп'ютер зможе вирішувати дану задачу, якою б складною вона не була. Але, ясна річ, він буде тупо виконувати інструкції, абсолютно не розуміючи їх змісту. В програмі можуть залишатися помилки. Вони можуть призводити до небажаних наслідків, але комп'ютер ніяк не зможе уникнути цих наслідків. Він не зможе самостійно перейти до виконання іншої задачі, навіть якщо вона майже нічим не відрізняється від попередньої. Далі, якщо зміняться умови, за яких виконується програма, комп'ютер може виявитися безпомічним.

Інший підхід - **декларативний**. Інтелектуальному виконавцеві (людині чи комп'ютерові) досить сказати, **що** треба робити, тобто лише сформулювати завдання. **Як** це завдання буде виконуватися - повинен визначити сам виконавець.

Наведемо відомий анекдот про спілкування з інтелектуальною системою. Людина приходить на роботу і питає у робота:

- Де це?
- Що саме? - запитує той.
- Сам знаєш, - відповідає людина.
- А... - каже машина і приносить те, що потрібно.

Запустити космічний корабель так, щоб він приземлився на Місяці, взяв зразки ґрунту та привіз їх назад - задача дуже складна, але вона піддається точній

алгоритмізації. Математичні методи дозволяють точно розрахувати траєкторію, по якій повинен рухатися такий корабель. Навіть якщо він випадково відхилиться від цієї траєкторії, відомі методи автоматичного регулювання дозволять ліквідувати це відхилення.

Послати слухняного робота до магазину за пляшкою молока - задача, на декілька порядків більш складна. Не кажучи вже про сам процес спілкування з продавцем, робот повинен вирішити ряд не зовсім формалізованих підзадач. Заздалегідь розрахувати траєкторію руху неможливо, оскільки робот повинен уникнути зіткнень з людьми та автомобілями. Якщо магазин закритий на переоблік, він повинен знайти інший магазин. **Неможливо передбачити всі ситуації, які можуть виникнути, а відтак - алгоритмізувати розв'язок задачі.** Тому виконання такого завдання під силу лише інтелектуальній системі, яка **вміє орієнтуватися в зовнішньому світі, аналізувати поточні ситуації та коригувати, адаптувати свою поведінку на основі такого аналізу.**

Інструкції, написані природною мовою, можуть бути, з одного боку, досить неоднозначними, а з іншого - **вони завжди спираються на те, що виконавець має певний апіорний досвід.** Уявіть собі дві таблички. На першій, яка знаходиться при вході на будівництво, написано "Обов'язково надягніть на голову каску". На другій, біля входу до лондонського метро - "Обов'язково візьміть на руки собак".

Ці інструкції однакові за формою, але абсолютно різні за змістом. Людина розуміє не лише те, що написано на табличках, але й те, що "залишається за кадром" і явно не вказується. Людина **уявляє собі ситуації**, які можуть виникати на будівництві та в метро, і **поповнює первинні інструкції**, що сприймаються нею безпосередньо. Зокрема, на будівництві зверху можуть падати цеглини, і тому потрібна каска, щоб захистити голову. До такого висновку людина могла б прийти і самостійно, без нагадування. Без каски на будівництво не пустять, і, якщо каски немає, її потрібно придбати. Але ясно, що купувати собаку перед тим, як заходити в метро, зовсім необов'язково.

Можна сказати, що людина має певну суму **знань про світ**, яка дозволяє їй орієнтуватися в життєвих ситуаціях та приймати правильні рішення. Крім того, людина вміє певним чином **використовувати ці знання**. Цих же рис повинна набути система штучного інтелекту

Зовнішній світ настільки багатий, що неможливо попередньо описати життєві ситуації з тією ступеню деталізації і однозначності, які б дозволили закласти в системи, що проєктуються, жорсткі детерміновані алгоритми поведінки. Тому **інтелектуальні системи повинні мати механізми адаптації, які б дозволяли їм вирішувати поставлені перед ними задачі на основі аналізу поточної ситуації.**

Людина не отримує всі свої знання і уміння зразу, тим більше неможливо передати всі необхідні знання і уміння інтелектуальній системі, призначеної для вирішення досить складних задач. Тому винятково важливе значення має здатність інтелектуальної системи до **навчання**, тобто до поповнення своїх знань і до набуття нових умінь. Таке навчання може відбуватися шляхом спілкування інтелектуальної системи з людиною або з іншою інтелектуальною системою. Якщо ж навчання відбувається самостійно, мова йде про **самонавчання**.

Багато спеціалістів вважають, що справді розвинені інтелектуальні системи повинні мати можливість безпосередньо сприймати оточуючий світ за допомогою спеціальних органів почуттів. Без такого світовідчуття слова залишаються лише

словами; людина ж у процесі мислення істотно спирається на чуттєві асоціації, пов'язані з цими словами. На основі цих асоціацій виникають інші асоціації, і т.д. Все це має рефлекторний характер і зумовлює активну участь підсвідомих інтуїтивних процесів у людському мисленні, прийняття рішень і т.п. На цій підставі Дрейфус у 70-ті роки взагалі піддав сумніву можливість існування штучних систем, подібних до людини за своїми можливостями.

Значного поширення набули *інтелектуальні роботи*. В принципі, можна говорити про їх здатність безпосередньо сприймати зовнішній світ за допомогою органів почуттів, але можливості використання цього світовідчуття для активізації підсвідомих рефлекторних процесів у роботів поки що залишаються дуже слабкими.

2.2. Квазиалгоритми та джерела квазиалгоритмічності

Ми бачили, що чисто алгоритмічний підхід виявив свою неспроможність при створенні інтелектуальних систем, призначених для вирішення досить складних неформалізованих задач. Тому для теорії і практики штучного інтелекту важливе значення має узагальнення поняття алгоритму, яке дістало назву *“квазиалгоритм”* або *“квазиалгоритмічна процедура”*.

Нагадаємо, що алгоритмом називається чітка недвозначна послідовність інструкцій, що зрозуміла виконавцеві і обов'язково призводить до гарантованого результату. На відміну від цього, інструкції квазиалгоритму можуть бути не зовсім чіткими, і результат виконання квазиалгоритмічної процедури не обов'язково є гарантованим.

Можна виділити як мінімум чотири основні джерела квазиалгоритмічності. Вони можуть доповнювати одне одне, у деяких випадках вони можуть бути описані схожими математичними моделями, але ці джерела мають принципово різну природу. Розглянемо їх.

1. *Дія випадкових чинників*, що не залежать від виконавця. Строго кажучи, з огляду на цей фактор квазиалгоритмом слід вважати будь-який, навіть найбільш формалізований, алгоритм. Алгоритм розрахований на роботу при певних умовах; якщо ці умови зміняться, алгоритм може не призвести до потрібного результату. Так, навіть програма на Паскалі, що обчислює суму двох двійок, не буде виконана, якщо під час її виконання в комп'ютері випадково зникне напруга. Якщо ж не вдається чітко окреслити межі застосування алгоритму або забезпечити виконання необхідних умов для його роботи, ми маємо справу зі справжнім квазиалгоритмом.

2. *Недостатнє врахування автором алгоритмічної процедури особливостей виконавця*, і це призводить до того, що виконавець неправильно розуміє, що від нього вимагається. Процедура може бути в принципі алгоритмічною, за нормальних умов призводити до гарантованого результату, але цей результат може бути не передбачений автором і тому бути для нього несподіванкою.

3. *Нечіткість формулювань*, що фігурують в описі процедури. Розглянемо будь-який кулінарний рецепт, наприклад: *“розтерти тісто, змішати з дрібно нарізаними яблуками, додати солі і перцю за смаком і смажити до появи рум'яної корочки”*. Це є типовим прикладом нечіткості формулювань, що

призводить до невизначеностей. До якої межі слід розтирати тісто? Що означає “дрібно нарізані”? Що означає “за смаком”? Як формалізувати момент “появи рум’яної корочки”? Неінтелектуальна система, орієнтована на чисто алгоритмічне керування, просто не зрозуміє цього опису і тому не зможе його виконати. Інтелектуальна ж система, наприклад, людина, повинна спробувати поповнити і уточнити цей опис на основі наявних знань і досвіду. Сам по собі факт виконання завдання буде свідчити про те, що квазіалгоритмічний первинний опис процедури був зведений до деякого внутрішнього алгоритму, але навряд чи виконавець зможе чітко пояснити і розписати цей алгоритм. Крім того, ці внутрішні алгоритми для кожного виконавця будуть різними, і це зумовить різні результати - у даному випадку різний смак виробу.

4. **“Свобода волі”**, яку можуть мати високоорганізовані, по-справжньому інтелектуальні системи. Ці системи можуть мати свої цілі, і, якщо дана процедура суперечить цим цілям, вони можуть відмовитися її виконувати або виконати не так, як це від них вимагається. “Свобода волі” полягає у тому, що інтелектуальна система самостійно приймає рішення про свою поведінку і в залежності від цих рішень може виконувати зовнішні розпорядження, не виконувати їх, або ж виконувати так, як вона вважає за потрібне.

2.3. Характеристика інтелектуальних систем з загальнокібернетичної точки зору

Цей параграф переслідує дві цілі. Перша з них - узагальнити і систематизувати той матеріал, що був даний раніше, і друга - пояснити, як інтелектуальна система будь-якої природи може бути описана з загальнокібернетичних позицій.

У 1948 р. видатний американський учений Норберт Вінер виклав основи нової науки - **кібернетики** у своїй видатній роботі **“Кібернетика, або керування і зв’язок у тварині і машині”**. Кібернетика вивчає риси, спільні для складних систем будь-якої природи, зокрема, процеси перетворення інформації у системах та керування ними.

Ключовим поняттям кібернетики є поняття **кібернетичної системи**. Кібернетична система визначена в “Словнику з кібернетики” як **“сукупність пов’язаних один з одним об’єктів (елементів системи), що здатні сприймати, зберігати, перетворювати інформацію, а також обмінюватися інформацією”**.

Так, наприклад, людський організм можна розглядати як систему, що складається з окремих елементів-клітин; органи людського тіла при цьому розглядаються як **підсистеми**. Суспільство можна розглядати як систему, елементами якої виступають окремі люди.

Можна дати ще одне визначення системи, в основі якого лежить визначення, сформульоване Холлом і Фейджином:

Визначення. Системою називається сукупність порівняно простих елементарних структур або процесів, що взаємодіють між собою та об’єднані в єдине ціле для виконання деякої спільної функції, що не зводиться до функцій окремих компонентів. Система має такі ознаки:

- вона взаємодіє з зовнішнім середовищем та іншими системами як єдине ціле;
- вона являє собою ієрархію підсистем більш низького рівня;

- вона є підсистемою для деякої системи більш високого рівня;
- вона зберігає загальну структуру взаємодії елементів при зміні зовнішніх умов та внутрішнього стану.

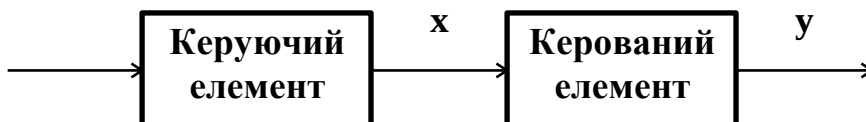
І вся система в цілому, і кожний окремий елемент може розглядатися як **інформаційний перетворювач**, що сприймає від зовнішнього середовища деякі **вхідні величини** (подразники, стимули) і в залежності від цих подразників та свого **внутрішнього стану** генерує **вихідні величини** (реакції). Іншими словами, якщо через $x_1, \dots, x_n$ позначити вхідні подразники, через $y_1, \dots, y_m$ - реакції системи, а через $a_1, \dots, a_q$ - змінні, що визначають поточний стан, то

$$y_j = f(x_1, \dots, x_n, a_1, \dots, a_q), j = 1, \dots, m.$$

Безумовно, внутрішні стани інформаційних перетворювачів також можуть змінюватися. Як моделі інформаційних перетворювачів можна розглядати, наприклад, скінченні автомати або інші формалізми.

Далі, ключовим поняттям кібернетики є поняття **контуру керування**. Виділяють два основні типи контурів керування: **замкнений** та **розімкнений**.

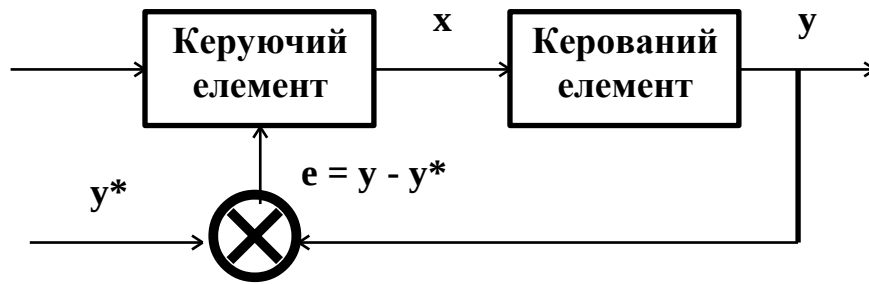
Мал.2.1 ілюструє найпростіший розімкнений контур керування. Він передбачає наявність двох основних елементів: керуючого та керованого. Для простоти викладення вважається, що кожний елемент має одну вхідну і одну вихідну величину (один вхід і один вихід); інші випадки повністю аналогічні. Вихідні сигнали керуючого елемента є вхідними для керованого елемента. Отже, для того, щоб забезпечити потрібні вихідні реакції керованого елемента, необхідно перш за все настроїти керуючий елемент.



Мал.2.1. Розімкнений контур керування

Очевидно, що розімкнений контур керування не є достатньо гнучким. Крім виходу керуючого елемента, на керований елемент впливають випадкові чинники, внаслідок чого його реакція може змінюватися непередбаченим чином. Тоді доводиться перенастроювати систему. Виникає питання, чи не можна автоматизувати таку перенастройку; іншими словами, чи не може контур керування набути необхідних **адаптивних** характеристик? Так, адаптивне керування може бути здійснене в рамках замкненого контуру на основі одного з найбільш фундаментальних принципів теорії автоматичного керування і кібернетики - **принципу зворотного зв'язку**.

Принцип зворотного зв'язку полягає в тому, що вихідні сигнали системи знову подаються на її вхід. Це проілюстровано на мал. 2.2.



Мал.2.2. Замкнений контур керування зі зворотнім зв'язком

До складу керуючого елемента включено новий компонент, який порівнює еталонний вихід y^* з поточним виходом y . Різниця $e = y - y^*$ аналізується керуючим елементом, і на підставі цього аналізу коригується керуючий сигнал x .

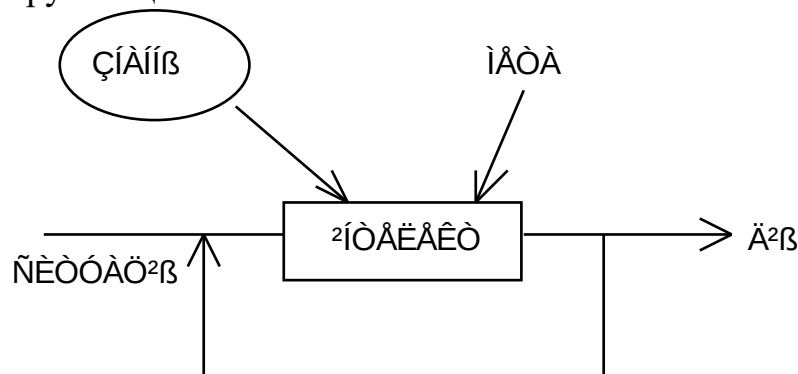
Виділяють два основні типи зворотнього зв'язку:

- **негативний**, спрямований на ліквідацію відхилення реакції системи від деякого усталеного режиму;
- **позитивний**, спрямований на збільшення відхилення.

Інтелектуальна система також може припускати зовнішнє керування, але для неї характерною є **самокеруваність**. Система має певну мету і прагне так планувати свої дії, щоб досягати цієї мети. Як вхідні стимули системи можна розглядати поточну ситуацію, що сприймається і аналізується системою. Результатом реакції системи стає зміна зовнішньої ситуації, і поведінка системи коригується в залежності від того, бажаною чи небажаною є ця зміна.

Необхідною компонентою інтелектуальної системи є **знання про світ**, на основі яких вона поповнює первинний опис зовнішньої ситуації з метою більш глибокого розуміння цієї ситуації, аналізує можливі наслідки своїх дій, і, можливо, формує або коригує свою мету.

Мал. 2.3. ілюструє вищесказане.



Мал.2.3 Схема самокерування інтелектуальної системи.

Усе це відноситься до будь-якої інтелектуальної системи, незалежно від того, чи мова йде про природний розум людини, чи про природний розум інопланетянина, чи про штучний інтелект. У деякому грубому наближенні можна

вважати, що чим **більш інтелектуальною є система, тим більш ефективно вона може досягати поставленої перед нею мети.**

Отже, перелік інтелектуальних задач повинен впливати із загальної концепції розуму, який повинен орієнтуватися в навколишній ситуації, ініціювати якісь дії з метою досягнення своєї мети та накопичувати знання про світ. До цього переліку можна віднести такі задачі:

- **аналіз ситуацій;**
- **розпізнавання образів;**
- **логічне мислення;**
- **розуміння нової інформації;**
- **навчання і самонавчання;**
- **планування цілеспрямованих дій.**

На основі вищесказаного можна сформулювати таке визначення інтелектуальної системи.

Визначення. *Інтелектуальною системою називається самокерована кібернетична система, яка має певну суму знань про світ і здатна на основі безпосереднього сприйняття і подальшого аналізу поточної ситуації планувати дії, спрямовані на досягнення певної мети, а також поповнювати свої знання.*

Безумовно, це визначення, так само, як і всі інші, не може претендувати на повноту і універсальність. Воно є сфокусованим на основних задачах, які повинна вміти вирішувати будь-яка природна, технічна або програмна система для того, щоб вона могла вважатися інтелектуальною.

2.4. Класифікація основних напрямків досліджень

Однією з найбільш традиційних класифікацій напрямків досліджень у галузі штучного інтелекту є такою.

Перший напрямок носить назву **біологічний** і полягає в такому. Велися спроби моделювання діяльності мозку людини, його психофізіологічних властивостей. Потім за допомогою ЕОМ чи інших технічних пристроїв для реалізації відповідної моделі сподівалися отримати штучний інтелект, штучний розум. З часом стало зрозумілим, що в повному обсязі цю задачу вирішити неможливо.

Власне кажучи, і навряд чи потрібно. Моделювання особливостей людського розуму в повному обсязі призведе до копіювання не тільки позитивних якостей, але й негативних. Стало ясно, що таке моделювання, хоч і має певне значення для створення штучного інтелекту, більше підходить для вивчення інтелекту природного. Таке вивчення оформилося у вигляді самостійної науки, яка дістала назву **когнітивної психології**.

Інший підхід носить назву **прагматичного**. В ньому майже не розглядається психофізіологічна діяльність людського мозку. Розвиваються підходи до вирішення інтелектуальних задач, незалежно від того, як співвідносяться ці підходи з тими процесами, що відбуваються в мозку людини.

Існує інша класифікація систем штучного інтелекту в залежності від моделей та методів, що використовуються для вирішення практичних задач. Йдеться про **символьний та конекціоністський підходи**. Ці підходи будуть детально розглянутими в ході вивчення курсу; зараз можна відмітити, що **символьний**

підхід орієнтований на моделювання процесів свідомого логічного мислення, а конекціоністський - підсвідомих рефлексорних процесів. Підходи можуть застосовуватися в чистому вигляді або в комбінації, і найбільших успіхів можна досягти, лише комбінуючи ці підходи. Ясно, що без використання рефлексорних методів жодна серйозна інтелектуальна проблема не може бути вирішена через феномен, який Річард Беллман називав *“прокляттям розмірності”*. З іншого боку, рефлексорні рішення далеко не завжди правильні та оптимальні, і тому людина не може обійтись без логічного мислення, як не можуть без нього обійтись і системи штучного інтелекту.

І, нарешті, дослідження в галузі штучного інтелекту можна класифікувати **за вирішуваними задачами.**

Серед практичних інтелектуальних задач, у розв’язку яких можна відмітити реальні здобутки - **розпізнавання образів, логічне мислення, створення діалогових систем типу людина-машина та систем автоматизованого перекладу текстів.**

Нарешті, можна коротко зупинитися на соціальних наслідках інтелектуалізації інформаційних технологій. Позитивні наслідки зрозумілі та добре відомі. З наслідків, не дуже сприятливих для людини, найочевидний - це поступове витіснення людей машинами як у процесі виробництва так і в управлінні суспільством. Подібні тенденції почали проявлятися ще задовго до початку комп’ютерної ери. Що стосується майбутнього, то тут письменники - фантасти малюють похмурі картини повного захоплення влади комп’ютерами і навіть фізичного винищення людства.

Було запропоновано ряд принципів, які орієнтовані на зменшення ризиків, пов’язаних з комп’ютеризацією та інтелектуалізацією. Найбільш відомими є принципи, сформульовані американським фантастом Айзеком Азімовим. Ось ці принципи:

- комп’ютер або робот не повинен заподіяти шкоди людині;
- він повинен виконувати накази людини, якщо це не суперечить першому принципу;
- він повинен прагнути до самозбереження, якщо це не суперечить першим двом принципам.

Інша проблема - як ці, або подібні їм, принципи втілювати в життя. Слід відмітити і те, що час, коли штучний інтелект розвинеться настільки, що зможе реально загрожувати людському, настане ще дуже не скоро.

Серед інших негативних аспектів інтелектуалізації можна відмітити те, що сучасні комп’ютерні технології полегшують здійснення не тільки корисних справ, але й різних зловживань, злочинів і т.п.

ЗАПИТАННЯ ТА ВПРАВИ ДО РОЗДІЛУ

1. Що таке квазіалгоритм, у чому полягає відмінність квазіалгоритмічної процедури від алгоритмічної?
2. Охарактеризуйте відомі Вам джерела квазіалгоритмічності, наведіть приклади.
3. У чому, на Вашу думку, проявляється квазіалгоритмічність людського мислення?
4. Дайте визначення поняття “система”.

5. Опишіть принцип зворотнього зв'язку, наведіть приклади позитивного і негативного зворотнього зв'язку.
6. Наведіть декілька прикладів інформаційного перетворювача.

ЧАСТИНА II. ОСНОВНІ КЛАСИЧНІ ПІДХОДИ ДО ВИРІШЕННЯ ТИПОВИХ ІНТЕЛЕКТУАЛЬНИХ ЗАДАЧ

Розділ 3. Основні підходи до планування цілеспрямованих дій

*Розвиток має бути
планомірним і пропорційним
(З матеріалів з'їздів КПРС)*

3.1. Перші спроби. “Загальний вирішувач задач”.

У кінці 50-х та на початку 60-х років Ньюелл, Саймон і Шоу опублікували піонерські роботи, які заклали початок класичного евристичного підходу до вирішення інтелектуальних задач; при цьому основний акцент був зроблений на планування цілеспрямованих дій. Першою інтелектуальною програмою стала розроблена ними **програма Логік-Теоретик**, за допомогою якої можна було доводити теореми математичної логіки. Ідеологія нового підходу полягала в породженні різноманітних здогадок та припущень з подальшою перевіркою їх справедливості.

Розвиток цього напрямку був пов'язаний з **програмою “Загальний вирішувач задач” (GPS - General Problem Solver)** (інша назва - “Універсальний вирішувач задач”). Були запропоновані деякі принципи розв'язку задач з довільної предметної області. Підхід ґрунтується на розгляді ситуацій, що виникають при вирішенні задач, та операторів, які можуть змінювати ці ситуації. Методика **GPS** передбачає аналіз цілей та засобів для їх досягнення. Аналіз цілей впливає з аналізу розбіжностей між поточною та бажаною ситуаціями. Схему прийняття рішення можна представити у такому вигляді [3]:

1. Проаналізувати поточну ситуацію.
2. Порівняти ситуацію з бажаною, якщо відмінностей немає - кінець роботи.
3. З'ясувати, який оператор або оператори можна застосувати для зменшення існуючої різниці.
4. Послідовно застосовувати оператори, знайдені на кроці 3, поки один з них не спрацює.
5. Повернутися на крок 1.

Ця проста схема, очевидно, не залежить від конкретної предметної області та в принципі може бути застосована до вирішення будь-якої задачі. На попередньому етапі необхідно зафіксувати *перелік можливих відмінностей між поточною та бажаною ситуаціями та перелік операторів, які можуть ліквідовувати ці відмінності*.

Ключовим поняттям при використанні “Універсального вирішувача задач” є **евристична таблиця, або таблиця відмінностей**. Її рядки відповідають типам відмінностей, а стовпці - різним операторам. Якщо даний оператор може застосовуватися для зняття даного типу розбіжностей, в клітині перетину рядка і стовпця ставиться одиниця, в іншому випадку ставиться нуль.

Як приклад ілюстрації цього підходу Ньюел, Шоу і Саймон наводять такий монолог: *"Я хочу доставити мого сина до дитячого садка. У чому полягає різниця між тим, що є, і бажаною ситуацією? У відстані. Що може змінити відстань? Мій автомобіль. Але він пошкоджений. Що потрібно щоб його полагодити? Новий акумулятор. Де знайти новий акумулятор? У майстерні або в крамниці. Далі, відмінність полягає у тому, що у майстерні не знають, що мені потрібний новий акумулятор. Що може зняти цю відмінність? Зв'язок. Які є засоби для зв'язку? Телефон"*, і т.д. При цьому, часто виникає необхідність повернення назад. Так, наприклад, акумулятори можуть бути в крамниці Сміта або в крамниці Брауна. Нехай вибрано крамницю Сміта. Якщо з'ясується, що телефон у крамниці Сміта не працює, доведеться повертатися на декілька кроків назад, змінювати рішення та телефонувати до крамниці Брауна.

При розвитку і вдосконаленні ідеї GPS стало ясно, що при прийнятті рішень необхідно брати до уваги інші характеристики конкретної задачі, наприклад, причинно-наслідкові зв'язки. В [3] наведено ще один приклад на цю тему. Є дворукий нерухомий робот-маніпулятор, призначений для переміщення предметів. Є два оператори: *"взяти лівою рукою"* та *"взяти правою рукою"*. Нехай спочатку робот спробував взяти предмет правою рукою і зазнав невдачі. Тут важлива причина невдачі. Якщо вона полягає у тому, що до предмета неможливо дістати, слід спробувати інший оператор - *"взяти лівою рукою"*, оскільки ліва рука розташована в іншій точці простору. Якщо ж причиною невдачі була надто висока температура предмету, то невдача буде чекати і при спробі взяти його лівою рукою, і тому цю спробу робити не варто.

Можливість застосування **GPS** була проілюстрована на ряді порівняно простих задач. Проте процедура виявилася не настільки універсальною. В реальних ситуаціях кількість типів відмінностей та можливих операторів може бути надто великою. Але є ще одна причина, яка особливо яскраво виявилася при спробі перевірити *"Загальний вирішувач задач"* на грі в шахи [2].

Позицій у шахах дуже багато, але відмінностей між ними та операторів (можливих ходів) відносно мало. Був визначений набір розбіжностей та заповнена евристична таблиця. Але програма грала дуже погано. Причина полягала у тому, що раціонально визначити відмінності між **довільними та кінцевими (матовими або нічийними) позиціями виявилось неможливим**.

Задача про мавпу і банани є однією з найбільш відомих задач у теорії штучного інтелекту. Проілюструємо на ній методику застосування "Загального вирішувача задач".

3.2. Прийняття рішень як оптимізаційна задача

Широкий клас задач прийняття рішень може бути сформульований у вигляді класичної оптимізаційної задачі: знайти рішення, на якому деяка цільова функція досягає свого максимуму при заданих обмеженнях. Так, наприклад, керівник фірми бажає максимізувати свій дохід, не порушуючи при цьому закони. Або потрібно якнайшвидше посадити космічний корабель на Марсі, але так, щоб сила удару була не надто великою.

Подібні задачі являють собою предмет науки, яка називається **дослідженням операцій**. Більш формально оптимізаційну задачу можна сформулювати в такому вигляді:

Знайти $x = (x_1, \dots, x_n)$, при якому функція $f(x)$ досягає максимума, але задовольняються обмеження $g_i(x) \geq 0$.

Функція $f(x)$ називається **цільовою функцією**, а функції $g_i(x)$ - **обмеженнями** оптимізаційної задачі.

Будь-який елемент x який задовольняє обмеженням $g_i(x) \geq 0$, називається **припустимим рішенням** задачі. Якщо умова максимізації не ставиться, йдеться про **задачу пошуку припустимих рішень**; ця задача має велике значення для теорії та практики штучного інтелекту.

Якщо обмежень немає, йдеться про **безумовну оптимізацію**.

Умова максимізації цільової функції ніяк не звужує загальності постановки. Дійсно, якщо треба вирішити задачу **мінімізації** функції $g(x)$, ми завжди можемо поміняти знак цієї функції і вирішувати задачу **максимізації** функції $h(x) = -g(x)$.

Втім, не будь-яка інтелектуальна задача припускає очевидне зведення до оптимізаційної, оскільки часто не вдається записати в явному вигляді цільову функцію або обмеження.

В залежності від вигляду цільової функції та функцій-обмежень розрізняють різні часткові випадки оптимізаційних задач, для кожного з них існують свої методи вирішення. Так, оптимізаційна задача з лінійною цільовою функцією та лінійними обмеженнями називається **задачею лінійного програмування**. **Нелінійне програмування** є більш складним розділом дослідження операцій. Якщо всі x_i цілі, йдеться про **дискретне програмування**. Задачі цього типу називаються також **комбінаторними задачами**, а пошук оптимальних рішень для комбінаторних задач - **комбінаторною оптимізацією**.

Існує очевидний і досить універсальний метод вирішення оптимізаційних задач, який може бути застосований, якщо множина припустимих рішень M обмежена (компактність є більш сильною вимогою, але для більшості практичних ситуацій вона не є необхідною). Це метод **повного перебору**, який полягає у переборі всіх можливих варіантів. Метод дає гарантоване рішення, якщо множина M скінченна (ситуація, характерна для дискретного програмування), і існує ефективний алгоритм породження будь-якого елемента з M та обчислення на цьому елементі цільової функції. Якщо ж множина припустимих рішень являє собою континуум, слід використовувати сіткову апроксимацію.

Таким чином, ми маємо типову загальноінтелектуальну процедуру. Якщо інтелектуальна система потрапляє в нову ситуацію і намагається планувати подальші дії, вона може спробувати звести задачу планування цілеспрямованих дій до оптимізаційної задачі (для цього, очевидно, достатньо визначити множину припустимих рішень M і цільову функцію $f(x)$). Якщо це вдається, повний перебір варіантів у більшості випадків дозволить отримати оптимальне рішення.

Але повний перебір має очевидний недолік: для більшості практичних ситуацій кількість варіантів, які доводиться перебирати, надто велика, і реалізувати метод за прийнятний час не є можливим. Тому **основною задачею класичної**

теорії дослідження операцій слід вважати пошук більш ефективних методів вирішення оптимізаційних задач.

Що ж стосується теорії і практики інтелектуальних систем, тут слід особливо підкреслити два аспекти:

- по-перше, далеко не завжди слід шукати саме оптимальне рішення; часто достатнім є рішення субоптимальне або навіть просто припустимим;
- зведення інтелектуальної задачі до оптимізаційної також є самостійною задачею, яка заслуговує на особливу увагу;
- розвинені системи штучного інтелекту повинні в процесі самонавчання самостійно виробляти методи вирішення оптимізаційних задач.

3.3. Евристичний пошук

Багато практичних задач може бути вирішено на основі процедури, яка носить назву **евристичного пошуку**. Евристичним пошуком прийнято називати процедуру систематизованого перебору на основі послідовного прийняття рішень. На основі [5] можна навести таку загальну схему евристичного пошуку:

1. *Вибрати деяку дію з області можливих дій.*
2. *Здійснити дію; це призведе до зміни ситуації.*
3. *Оцінити нову ситуацію.*
4. *Якщо досягнуто успіху - кінець; якщо ні - повернутися на крок 1 і почати спочатку.*

Зокрема, “Загальний вирішувач задач” можна розглядати як одну з можливих реалізацій евристичного пошуку.

Евристичний пошук часто розглядається як **пошук шляху на дереві або на графі**. Дійсно, розглянемо типову схему вирішення задачі, згідно з якою на кожному кроці можна вибирати одну з можливих дій. Тоді можна говорити про породження **дерева можливостей**, вузли якого відповідають ситуаціям проблемної області, а дуги - можливим діям. Тоді задача переходу від початкової ситуації до бажаної зводиться до задачі пошуку шляху на дереві.

Аналогічно виникає і графова інтерпретація.

3.4. Основні алгоритми пошуку на дереві

Серед різноманітних методів, за допомогою яких можна організовувати пошук на дереві потрібної вершини, а відтак і шляху до неї, прийнято виділяти дві основні стратегії:

- **пошук в ширину**;
- **пошук в глибину** (інші назви - **перебір з поверненнями**; **бектрекінг**).

Процедура пошуку в ширину передбачає аналіз на кожному кроці синів усіх вершин, що були проаналізовані до цього (в інтерпретації прийняття рішень це означає **паралельну** перевірку усіх можливих альтернатив).

Процедура пошуку в глибину передбачає першочерговий аналіз нащадків тих вершин, що були проаналізовані останніми. Це означає, що всі альтернативи аналізуються **послідовно**, одна за одною; аналіз деякої альтернативи завершується лише тоді, коли вдається остаточно встановити, призводять вона до успіху чи ні.

Якщо ж альтернатива призводить до невдачі, відбувається **повернення** і розгляд інших альтернатив.

І пошук в ширину, і пошук в глибину при досить загальних умовах мають, взагалі кажучи, експоненційну оцінку часової складності. Можна навести ряд прикладів, коли перебір в ширину дозволяє виграти час порівняно з перебором в глибину і навпаки. Як правило, пошук в глибину дозволяє зекономити **пам'ять**, оскільки при її реалізації немає необхідності запам'ятовувати все дерево, достатньо зберігати в пам'яті лише вершини, що мають відношення до поточної альтернативи.

На практиці процедура пошуку в глибину набула значно більшого поширення. Можна стверджувати, що перебір з поверненням став класичною загальноінтелектуальною процедурою, що лягла в основу сучасних методик планування цілеспрямованих дій, програмування ігор, автоматизованого доведення теорем тощо.

3.5. “Жадібні” алгоритми

“Жадібні алгоритми” належать до класичних методів дослідження операцій. Їх застосування, як правило, дозволяє уникнути експоненційного зростання складності задачі.

Визначення. *Алгоритм вирішення деякої задачі називається жадібним, якщо на кожному кроці він намагається якнайближче підійти до мети. Іншими словами, жадібний алгоритм - це алгоритм, який заради негайного наближення до мети жертвує перевагами, можливими в перспективі.*

Якщо множина припустимих рішень являє собою деяку область **n**-вимірному простору, а цільова функція є диференційованою, жадібні алгоритми називаються **градієнтними**.

Нехай ми маємо деяке наближення до оптимального рішення. Суть градієнтних методів полягає в тому, що для найшвидшого досягнення максимуму необхідно рухатися до нього по найбільш крутому схилу, тобто на кожному кроці намагатися наблизитися до мети якнайшвидше. Математично це означає, що необхідно рухатися в напрямку градієнта цільової функції.

Якщо і обмеження, і цільова функція є функціями опуклими (у такому випадку кажуть про задачу **опуклого програмування**), то можна довести, що жадібні алгоритми гарантовано призводять до мети, тобто до оптимального рішення. Аналогічні результати справедливі і для ряду задач комбінаторної оптимізації (задачі на матроїдах). Але, взагалі кажучи, жадібні алгоритми не дозволяють досягти оптимального рішення, оскільки вони досягають не глобального максимуму цільової функції, а лише локального.

3.6. І-АБО - графи

І-АБО- графи зарекомендували себе як зручний спосіб представлення задач та методів їх вирішення. Ідеологія І-АБО-графів базується на декомпозиції задачі на підзадачі.

Будемо називати задачу **елементарною**, якщо для неї існує явний алгоритм вирішення. Тоді метою декомпозиції будь-якої задачі є її зведення до сукупності елементарних задач.

Визначення. *I-АБО-графом називається орієнтований граф, вершини якого відповідають задачам, а дуги - відношенням між задачами. Між дугами вводяться відношення I та АБО.*

Нехай задача **A** зв'язана дугами з задачами **B** та **C** (при плануванні рішень це означає, що задачу **A** можна звести до задач **B** та **C**). Будемо казати, що дуга **AB** зв'язана з дугою **AC** відношенням **I**, якщо для вирішення **A** необхідно вирішити і **B**, і **C**. Якщо ж для вирішення **A** достатньо вирішити лише одну з цих задач, то будемо казати, що відповідні дуги зв'язані відношенням **АБО**.

Будь-який I-АБО-граф може бути зведений до певної нормальної форми, у якій з будь-якої вершини виходять або тільки I-дуги, або тільки АБО-дуги.

Визначення. Вершину, з якої виходять лише I-дуги, будемо називати **I-вершиною**.

Вершину, з якої виходять лише АБО-дуги, будемо називати **АБО-вершиною**.

Одну з вершин графа, що відповідає початковій задачі, будемо називати **початковою вершиною**.

Визначення. Вершина називається **розв'язною**, якщо задача, що відповідає цій вершині, має рішення.

Метою пошуку, що здійснюється на I-АБО-графі, є встановлення того, чи є початкова вершина розв'язною, чи ні. Взагалі кажучи, можна дати таке рекурсивне визначення розв'язної вершини:

- 1) Заключні вершини розв'язні, оскільки вони відповідають елементарним задачам;
- 2) Незаклучна I-вершина розв'язна, якщо розв'язними є всі вершини - нащадки;
- 3) Незаклучна АБО-вершина розв'язна, якщо розв'язною є хоча б одна з вершин - нащадків.

Аналогічне визначення нерозв'язної вершини читач може сформулювати самостійно.

Визначення. **Вирішальним графом** називається підграф з розв'язних вершин, який показує, що початкова вершина є розв'язною.

Вирішальний граф **T** являє собою **I-АБО-дерево**. Це дерево визначається таким чином:

- коренем дерева є початкова вершина **P**;
- якщо **P** є АБО-вершиною, то в **T** міститься будь-який з її розв'язних нащадків з I-АБО-графа разом з власним вирішальним деревом;
- якщо **P** є I-вершиною, то в **T** містяться будь-який з її розв'язних нащадків з I-АБО-графа разом з власним вирішальним деревом.

3.7. Простір задач і простір станів

Задачі планування цілеспрямованих дій прийнято розподіляти на два класи [1]: *планування в просторі станів (SS-проблема) та планування в просторі задач (PR-проблема)*.

При плануванні в просторі станів заданим вважається деякий набір станів (ситуацій). Опис кожної ситуації складається з опису стану як зовнішнього світу, так і самої інтелектуальної системи. Відомі дії, які може здійснювати система, і які визначають перехід з одного стану до іншого. Проблема полягає у пошуку шляху від початкового стану до одного з кінцевих. Легко бачити, що у такій постановці задачу планування цілеспрямованих дій можна уявляти собі як задачу пошуку шляху з однієї вершини до іншої на деякому графі. Відповідно, після того, як зведення задачі до формальної моделі проведено, можна використовувати відомі алгоритми пошуку шляхів на графах (алгоритми Мура, Дейкстри, гілок та границь і т.п.).

Визначення. *Графом станів задачі називається орієнтований граф, вершини якого відповідають можливим станам предметної області, а дуги - методам переходу від стану до стану.*

Дуги можуть мати мітки, які інтерпретуються як вартість або довжина відповідного переходу.

Тоді вирішення задачі являє собою пошук шляху від початкового стану до цільового; при цьому типовою є вимога оптимізації цього рішення, тобто пошуку найкоротшого шляху.

Класичним прикладом планування в просторі станів можна вважати задачу пошуку шляху від одного міста до іншого. Можна навести й інші, менш очевидні приклади.

Планування в просторі задач передбачає декомпозицію вихідної задачі на підзадачі, поки не буде досягнута задача, для якої є готовий алгоритм вирішення. Таку декомпозицію зручно уявляти собі у вигляді **I/АБО-графа**. Описаний раніше "Загальний вирішувач задач" можна розглядати як один з таких методів. Дійсно, в жорсткій послідовності здійснюються такі кроки: *аналізуються відмінності між початковою та бажаною ситуаціями, з'ясовується, що потрібно для усунення цих відмінностей; генеруються нові підзадачі.* Якщо позначити початковий стан через **A**, мету - через **B**, а засіб досягнення мети - через **C**, то декомпозиція здійснюється за схемою **<A,B> -> <A,C>, <C,B>**.

Відомі також алгоритм Ченга і Слейгла, метод ключових операторів і т.п.

Перспективним видається комбінування методів пошуку в просторі станів та в просторі задач.

3.8. Обмежуючі правила та евристики як засіб скорочення перебору

Ми бачили, що повний перебір або перебір з поверненням дозволяє у ряді випадків прийняти деяке рішення. Але часто реалізація цих методів забирає дуже багато часу і тому є неможливою чи недоцільною. Тому в таких ситуаціях вдаються до **обмежуючих правил** та **евристик** як до типових засобів скорочення перебору. Саме так у більшості випадків і діє людина при плануванні своїх дій.

Зараз ми зосереджуємося на евристиках, які використовуються при плануванні цілеспрямованих дій. В інших випадках слово "евристика" може мати дещо інше значення.

Визначення. *Обмежуючим правилом при плануванні цілеспрямованих дій називається правило, якому повинні бути підпорядковані альтернативні дії; що розглядаються.*

Іншими словами, застосування обмежуючих правил дозволяє включати до перебору не всі можливі дії, а лише ті, які не суперечать цим правилам. Це у ряді випадків дозволяє різко скоротити перебір, а інколи навіть звести задачу до поліноміальної. Зокрема, описані вище жадібні алгоритми можна розглядати як застосування обмежуючих правил.

Природа обмежуючих правил може бути різноманітною. Інколи вдається довести, що обмежуюче правило дозволяє відкинути завідомо безперспективні гілки і досягти того ж результату, що й повний перебір без застосування цього правила. Такі правила є **теоретично обґрунтованим** і повинні безумовно застосовуватися.

Але частіше доводиться зустрічатися з ситуацією, коли обмежуючі правила спираються на наявний апріорний досвід, але не обґрунтовані теоретично, і не гарантують або просто не дозволяють отримати оптимальне рішення. Застосування цих правил дозволяє скоротити перебір, але за рахунок втрати гарантованої оптимальності (власне, у багатьох випадках нічого кращого і не залишається). Саме такі правила і називаються евристиками.

Визначення. *Евристикою при плануванні цілеспрямованих дій називається обмежуюче правило, яке спирається на наявний досвід і не гарантує оптимальності рішення.*

Часто буває і так, що існує не одна евристика, а декілька, і вони повністю або частково суперечать одна одній. Тоді окремою проблемою стає проблема застосування евристик.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. (**Задача про мавпу і банани**). Деяка мавпа знаходиться у кімнаті, до стелі якої підвішена зв'язка бананів. У кімнаті знаходиться також скриня. Мавпа може переміщуватися по кімнаті, пересувати скриню та залазити на неї. Мавпа може дістати банани тільки тоді, коли вона залізе на скриню, розміщену точно під бананами.

Напишіть програму, яка вирішує задачу про мавпу і банани за допомогою процедури “Загального вирішувача задач”.

2. Проілюструйте на довільному прикладі зведення проблеми прийняття рішень до оптимізаційної задачі.

3. Перелічіть відомі Вам класи оптимізаційних задач.

4. Опишіть відому Вам універсальну процедуру вирішення задачі прийняття рішень. У чому полягають її позитивні та негативні риси?

5. Опишіть процедуру евристичного пошуку. Проілюструйте застосування цієї процедури до довільної задачі.

6. Охарактеризуйте процедури перебору в ширину та в глибину, порівняйте їх.

7. Напишіть програму на Паскалі або на Сі, яка реалізує пошук потрібної вершини на дереві:

- методом пошуку в ширину;

- методом пошуку в глибину.

8. Охарактеризуйте суть жадібних алгоритмів. Чи обов'язково вони дозволяють отримати оптимальне рішення?

9. Наведіть хоча б один клас оптимізаційних задач, для яких жадібний алгоритм дозволяє отримати оптимальне рішення.
10. Проілюструйте на довільному прикладі представлення довільної задачі у вигляді І-АБО-графа. Побудуйте для цього графа вирішальне дерево.
11. Дайте рекурсивне визначення нерозв'язної вершини І-АБО-графа.
12. На прикладі довільної задачі проілюструйте методику планування в просторі станів. Явно розпишіть усі можливі стани та процедури переходу від одного стану до іншого. Побудуйте граф станів та проілюструйте вирішення задачі методом Дейкстри.
13. Охарактеризуйте поняття обмежуючих правил та евристик. В чому полягає їх значення при плануванні цілеспрямованих дій?
14. Поясніть, чому жадібні алгоритми можна розглядати як застосування обмежуючих правил (розпишіть відповідні обмежуючі правила в явному вигляді).
15. Для довільної предметної області наведіть приклади обмежуючих правил та евристик.

Розділ 4. Методи програмування ігрових задач

*Мне же неумение поможет-
Этот Шифер ни за что не сможет
Угадать, чем буду я ходить.*
(В. Висоцький. "Честь шахової корони")

4.1. Ігрові задачі

Ігрові задачі є типовим класом задач, які традиційно відносяться до інтелектуальних. Оскільки вибір чергового ходу в іграх є не що інше, як прийняття рішення, методи програмування ігрових задач найтіснішим чином пов'язані з методами планування цілеспрямованих дій і прийняття рішень, що були описані в попередньому розділі.

Характерною особливістю ігрових задач є **наявність суперника**, який активно перешкоджає здійсненню цілей, які ставить перед собою кожний гравець.

Теорія ігор є важливою складовою частиною дослідження операцій. Її перше систематизоване викладення було зроблено Нейманом і Моргенштерном у 1944 р., хоча перші результати відносяться до 20-х років.

Визначення. *Теорія ігор - це математична дисципліна, яка вивчає питання поведінки учасників конфліктних ситуацій та має на меті виробити оптимальні, для кожного з учасників, стратегії такої поведінки. Конфліктною при цьому називається ситуація, коли гравці мають різні цілі (різні функції виграшу) та можуть вибирати різні засоби досягнення своїх цілей (стратегії).*

Для побудови систем штучного інтелекту найбільший інтерес становлять методи знаходження планів гри і оптимальних стратегій для таких ігор, як шахи, шашки, "хрестики-нулики" і т.п. З точки зору теорії ці ігри є ідентичними між собою. Вони відносяться до класу **позиційних ігор двох осіб**. Кожний гравець може по черзі зробити будь-який хід з тих, які дозволяються правилами гри. Ці ігри є **детермінованими** у тому розумінні, що перебіг гри та вибір ходу не залежать від випадкових чинників. Крім того, це є **ігри з повною інформацією**, тобто кожному гравцеві доступна вся інформація про будь-яку позицію, яка утворюється в процесі

гри. Нарешті, вказані ігри відносяться до класу *антагоністичних ігор*, або *ігор з нульовою сумою*. Це означає, що сума вигравів обох гравців дорівнює нулю, тобто виграв одного гравця дорівнює програшу іншого. З цього випливає, що замість двох функцій виграву можна розглядати одну.

Можна назвати відомі позиційні ігри, які належать до інших класів. Так, нарди є грою з повною інформацією, але не є детермінованою у тому розумінні, що гравець не може зробити довільний хід; його вибір обмежений випадковими чинниками. Преферанс не є грою з повною інформацією і не є грою детермінованою у тому розумінні, що початкова позиція залежить від випадку. Але після того, як початкова позиція зафіксована, гравець може зробити будь-який хід, який дозволений правилами.

4.2. Дерево гри та мінімаксна процедура

Ключовим для позиційних ігор є поняття *дерева гри*. Корінь цього дерева співпадає з початковою позицією. Кожний вузол цього дерева характеризується номером гравця, що має робити хід. Дуги відповідають ходам, тобто, якщо в позиції 1 можливий хід, який переводить позицію 1 в позицію 2, то з позиції 1 до позиції 2 йде орієнтована дуга, яка відповідає цьому ходу. Право вибору ходу в позиції 2 належить, звичайно, уже іншому гравцеві. Кожний вузол дерева корисно характеризувати також його *рівнем*, тобто відстанню від кореня. Якщо на k -му рівні право вибору ходу належить одному з гравців, то на $k+1$ -му рівні воно переходить до його суперника.

Нехай грають два гравці - **A** і **B**. Для визначеності будемо вважати, що право вибору ходу в початковій позиції належить гравцеві **A**. Функція виграву співпадає з функцією виграву цього гравця, тобто гравець **A** аналізує дерево гри зі своєї точки зору та прагне максимізувати виграв. Вершини, у яких право ходу належить гравцеві **A**, прийнято називати *альфа-вершинами*; вершини ж, у яких право ходу належить його суперникові, називаються *бета-вершинами*.

Теорія стверджує, що антагоністичні детерміновані позиційні ігри з повною інформацією (такі, як шахи), дерево гри яких має скінченну кількість вершин, мають оптимальну стратегію, відступати від якої не вигідно жодному з суперників. Іншими словами, **при правильній грі обох сторін гра завжди має закінчуватися однаковим результатом**.

Пропонується таке неформальне доведення цього факту. Ми припускаємо, що гра завжди завершується за скінченну кількість ходів (саме це і означає скінченність дерева гри). Якщо це не так, потрібно ввести додаткові правила, які дозволяють обривати нескінченні варіанти і фіксувати для них деякий результат.

Так, наприклад, дерево гри в шахи є потенційно нескінченним (хоча кількість можливих позицій є скінченною). Тому до Шахового Кодексу включено ряд правил, за якими фіксується нічия. Це, зокрема, трикратне повторення позиції, або ситуація, коли на протязі останніх 50 ходів не було зроблено жодного взяття і жоден пішак не зробив ходу (про недавні суперечки з цього приводу див. далі).

Для будь-якої гри, що завершується за скінченну кількість ходів, є теоретично можливим побудувати повне дерево гри, що охоплює всі можливі позиції. Почнемо аналіз дерева з завершальних позицій (листів дерева). Розглянемо довільні завершальні позиції, які мають одного батька. Нехай для визначеності це альфа-вершини. Кожна завершальна позиція має оцінку, яка співпадає з функцією

виграшу та являє собою результат гри. Ясно, що гравець **В**, якому належить право вибору ходу у батьківській вершині, вибере хід, який мінімізує цю оцінку (мінімізує його програш). Ця мінімальна оцінка **передається даній вершині знизу**.

Нехай всі наступники довільної бета-вершини уже проаналізовані та кожному передана знизу певна оцінка. Загальне правило можна сформулювати таким чином: **з бета-вершини повинен бути зроблений хід, що веде до позиції-наступника з найменшою оцінкою**.

За допомогою аналогічних міркувань можна встановити правило, згідно з яким повинен вибирати ходи гравець **А** : **з альфа-вершини повинен бути зроблений хід, що веде до позиції-наступника з найбільшою оцінкою**.

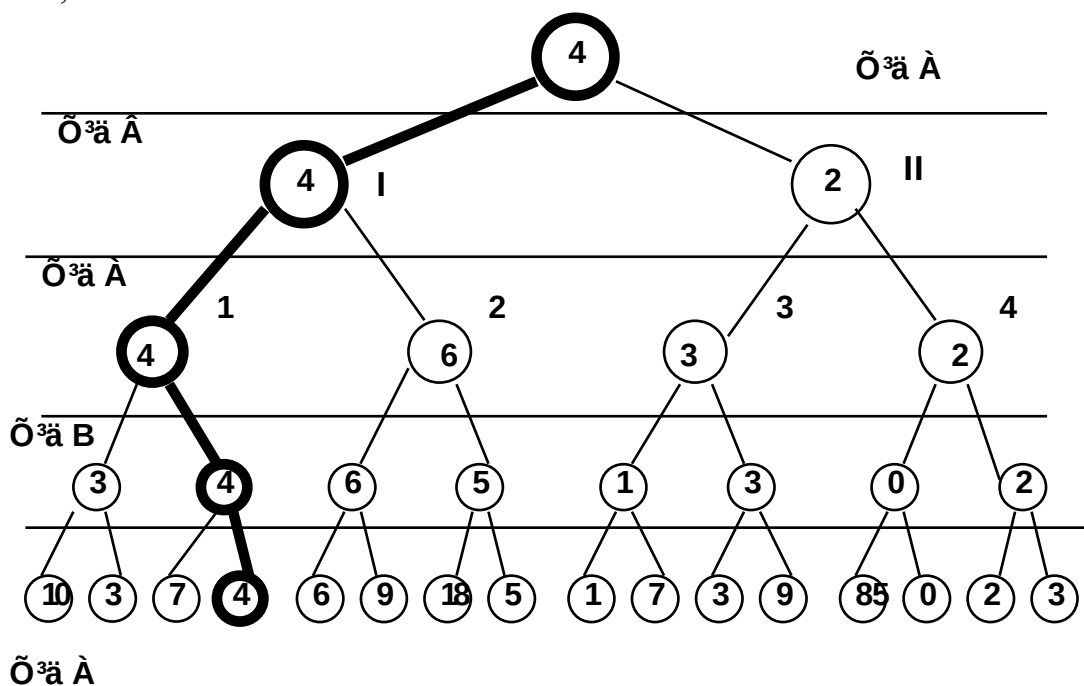
Такий аналіз може бути зроблений для будь-якої позиції, включаючи початкову. Ми **конструктивно** довели, що для будь-якої позиції результат гри є повністю визначеним, тобто сформулювали **правила, які визначають процедуру, що дозволяє досягти цього результату і знайти стратегію гри, оптимальну для кожного гравця**. Ця процедура носить назву **мінімаксної процедури**.

Оцінки позицій, передані знизу, будемо називати **мінімаксними оцінками**. Зрозуміло, що при правильній грі обох сторін вони співпадають з остаточними виграшами гравців.

Мінімаксна процедура є процедурою повного перебору, оскільки при її застосуванні необхідно породити, починаючи з вершини, все дерево гри, а потім отримати оцінки, рухаючись у зворотному порядку.

Для практичної реалізації мінімаксної процедури застосовуються, як правило, бектрекінгові алгоритми, хоча теоретично ніщо не заважає застосувати і перебір в ширину.

Мал. 4.1 ілюструє застосування мінімаксної процедури. Жирним кольором виділено варіант, оптимальний для обох гравців. Оцінки всіх вершин, крім завершальної, є мінімаксними.



Мал. 4.1 Ілюстрація мінімаксної процедури вибору ходу.

Для таких ігор, як "хрестики-нулики", побудова дерева гри та реалізація мінімаксної процедури не викликає особливих проблем. Але для шахів рішення

може бути знайдено тільки теоретично, навіть найпотужнішому комп'ютерові на повну реалізацію цієї процедури не вистачило б часу, що пройшов з моменту Великого Вибуху, тобто створення Всесвіту.

Дійсно, на кожному ході можна зробити вибір в середньому з 30 варіантів. Отже, аналіз дерева гри лише на один хід, з урахуванням відповіді суперника, потребує перегляду приблизно $1000 = 10^3$ позицій. Оскільки шахова партія триває приблизно 40 ходів, ми можемо приблизно оцінити загальну кількість вузлів дерева як 10^{120} . Читач може самостійно підрахувати, скільки часу повинен зайняти аналіз такого дерева.

Звичайно, в шаховій партії одній позиції можуть відповідати різні вузли дерева, але це не змінює загальної ситуації. Таким чином, ключовою слід вважати **проблему ефективного скорочення перебору**.

Звичайним підходом є обмеження *глибини перебору*, тобто пошук реалізації мінімаксної процедури для дерева гри з обмеженою кількістю ходів. При невеликій глибині (5-10 напівходів) дерево перебору скорочується до розмірів, що є прийнятними для перебору на сучасних комп'ютерах.

4.3. Мінімаксна процедура на усіченому дереві. Статичні оціночні функції

Основна проблема, пов'язана з обмеженням глибини перебору, полягає у тому, що для позицій, які знаходяться на максимальній глибині та на яких завершується аналіз варіантів з поточної вершини, оцінка позиції ще не визначена. У таких випадках замість остаточної оцінки використовують *статичні оціночні функції*.

Визначення. *Статичною оціночною функцією називається числова функція, яка призначена для оцінки виграшу гравця у даній позиції та залежить виключно від цієї позиції.*

Це визначення є простим і інтуїтивно зрозумілим. Але воно не зовсім чітке, оскільки і мінімаксна оцінка позиції в решті-решт однозначно визначається цією позицією. Справа в тому, що статичні оцінки, на відміну від мінімаксних, не враховують *динаміки* позиції, тобто можливостей її зміни; саме тому вони й називаються статичними.

Більш чітке визначення можна дати, якщо ввести до розгляду деяку **формальну модель гри**.

Визначення. Формальною моделлю гри будемо називати п'ятірку $M = \langle S, S_0, P, Q, F \rangle$, де:

S - множина всіх можливих позицій;

S_0 - початкова позиція;

P - множина правил, які зумовлюють можливість переходу від однієї позиції до іншої; іншими словами - множина припустимих ходів;

Q - множина завершальних позицій;

F - функція, яка визначена на завершальних позиціях та визначає виграші гравців (для антагоністичної гри достатньо задати функцію виграшів для одного з гравців).

Тепер можна дати визначення мінімаксної та статичної оціночних функцій.

Визначення. *Мінімаксною оціночною функцією будемо називати функцію $f: M \rightarrow R$, яка визначена для будь-якої позиції та обчислюється на основі описаної раніше мінімаксної процедури; значення цієї функції є мінімаксними оцінками.*

Визначення. *Статичною оціночною функцією будемо називати функцію $f: S \rightarrow R$, яка визначена для будь-якої позиції та обчислюється на основі власних факторів цієї позиції; значення цієї функції будемо називати статичними оцінками.*

Ясно, що при реалізації мінімаксної процедури **оцінка позиції, передана знизу, зовсім не зобов'язана співпадати з її власною статичною оцінкою** (і, як правило, не співпадає). Зрозуміло, що якби передані оцінки повністю співпадали зі статичними, така статична оціночна функція однозначно визначала б виграші гравців, і ніякі мінімаксні процедури були б непотрібними.

З іншого боку, очевидно, що статичні оцінки позицій, які знаходяться на горизонті, повинні бути певним чином пов'язані з мінімаксними оцінками цих позицій, отриманими для повного дерева (невідповідності суттєво позначаються на якості гри).

Часто використовуються **лінійні статичні оціночні функції**, які обчислюються за формулою

$$S(P) = \sum_{i=1}^n a_i x_i$$

де $x_i, i = 1, \dots, n$ - чисельні значення факторів, що впливають на оцінку позиції, $a_i, i = 1, \dots, n$ - вагові коефіцієнти, що визначають міру важливості того чи іншого фактора.

Так, при побудові шахових програм загальноприйнятим є аналіз матеріального співвідношення; пішак оцінюється одним очком, легка фігура - 3, тура має коефіцієнт 5 або 5,5; ферзь - 9 або 10 (ферзь слабший за дві тури). Втрата короля рівнозначна програшу, і тому він оцінюється великим числом (наприклад, 200). Певним чином оцінюються також позиційні фактори (активність фігур, пішаків структура тощо).

Важливо розуміти, що скорочене **дерево гри залежить від поточної позиції**, для позиції-наступника воно зміниться (подумайте самостійно, як саме).

Мінімаксна процедура, застосована до скороченого дерева, дозволяє знайти оптимальну стратегію лише для цього скороченого дерева, а не для всієї гри. Програма аналізує лише варіанти, які не виходять за "горизонт" (тобто які не перевищують максимальної глибини). Наприклад, у шахах цілком вірогідна ситуація, при якій з "оптимального" вузла програма зразу одержує мат. Якби глибина перебору була більшою, вона б обрала зовсім інший варіант. Тому **глибина перебору є одним з вирішальних чинників, що визначає силу ігрових програм та якість їх гри.**

Слід відмітити, що варіант може обірватися і до завершення максимальної глибини через остаточний виграш (мат) або теоретичну нічию, це може бути легко враховано.

4.4. Альфа-бета відтинання

Існує, і є загальноживимим варіант мінімаксної процедури, який еквівалентний описаному вище з точки зору оптимальності результату, але дозволяє досягти цього результату більш швидко. Йдеться про **альфа-бета-відтинання**. Воно дозволяє скоротити перебір шляхом відтинання тих гілок, про які заздалегідь відомо, що вони не призведуть до результату, кращого, ніж уже гарантований.

Проілюструємо суть альфа-бета відтинання на тому ж прикладі. Нехай гравець А робить аналіз дерева гри на мал.4.1, починаючи з кореня. Він має зробити вибір між ходами, позначеними I та II. З позиції, що утворюється після ходу 1, ведуть гілки, позначені цифрами 1 та 2, а з II - гілки 3 та 4.

Нехай уже проаналізована гілка 1, яка дала оцінку 4. З цього випливає, що гарантований виграш складає 4, тому варіанти, які не дадуть більшого виграшу, розглядати не варто. Так, якщо далі проаналізована гілка 4, вона дає максимальну оцінку 2 для позиції II. Звідси випливає, що хід до позиції II не дасть виграшу, більшого, ніж 2, і всі інші гілки, що виходять з цієї позиції (у нашому випадку гілку 3) аналізувати не варто.

Іншими словами, якщо деякий хід досягає максимально можливого виграшу, інші його альтернативи не розглядаються. Якщо для деякого ходу знайдене спростування, інші відповіді суперника не розглядаються.

Дамо більш формальний опис альфа-бета відтинання. З кожним вузлом пов'язується **попередня оцінка**, яка змінюється в ході аналізу його наступників. В альфа - вершині вона дорівнює $-\infty$ перед початком аналізу. Якщо остаточна оцінка вузла-наступника перевищує поточну попередню оцінку, цій попередній оцінці присвоюється нове значення. **Відтинання у альфа-вершині (припинення аналізу цієї вершини та її наступників) відбувається, коли попередня оцінка цієї вершини стає більшою, ніж попередня оцінка попередника (батька) цієї вершини.**

Аналогічне правило відтинання у бета-вершині спробуйте сформулювати самостійно.

Більш детальний опис та можливі алгоритмізації мінімаксної процедури і альфа-бета відтинання можна знайти у багатьох джерелах, наприклад, [4-6].

Ефективність скорочення перебору при альфа-бета відтинанні суттєво залежить від порядку перегляду варіантів. Згідно [4], кількість позицій, що доводиться аналізувати при максимальних альфа-бета відтинаннях, приблизно дорівнює подвоєному квадратному кореню від кількості позицій, які переглядаються при повному переборі. Там же наводиться типово дерево гри, яке аналізується при максимальних альфа-бета-відтинаннях.

На перший погляд, подвоєний квадратний корінь - це не таке велике скорочення. Але якщо врахувати, що в шахах уже при аналізі на 3-5 ходів доводиться переглядати 10^{10} - 10^{14} позицій, виявляється, що альфа-бета відтинання може скоротити час на вибір ходу в сотні тисяч або мільйони разів.

Існує ряд практичних рекомендацій з приводу упорядкування перебору позицій-наступників при використанні альфа-бета відтинання. Наприклад, **в першу чергу аналізувати наступників з найкращою статичною оцінкою.**

У літературі можна знайти багато досліджень про альфа-бета відтинання. Нехай ми при аналізі позиційної гри не враховуємо конкретної специфіки даної гри, а обмежуємося лише інформацією про те, що гра є скінченна, детермінована та з повною інформацією. Тоді **інших правил відтинання, еквівалентних повному переборі, не існує.**

Існує, звичайно, ряд рекомендацій, як підвищити якість гри шахових програм. Всі відомі програми мають у своєму складі "**службу кращих ходів**", яка спирається на рекомендації "*у таких-то позиціях грай так-то*" та використовує розвинені **дебютні і ендшпільні бібліотеки**. Добре запропонував себе **метод форсованих варіантів**. Його мета - не обривати варіант серед серії розмінів або в інших гострих позиціях. Метод полягає у тому, що після досягнення максимальної глибини аналіз продовжується, але розглядаються не всі ходи, а лише форсовані, наприклад, тільки шахи та взяття фігур. Метод підвищує якість комбінаційної гри та дозволяє різко скоротити ймовірність грубих помилок.

4.5. Огляд сучасних шахових програм

Шахові програми до останнього часу грали, взагалі кажучи, слабше за гросмейстерів, хоча значно сильніше, ніж пересічні шахісти - аматори. Сила шахових програм значно зростає, коли на дошці залишається мало фігур.

Останнім часом шахові програми інтенсивно використовуються для аналізу шахових закінчень [Д.3]. При цьому використовується **ретроспективний аналіз**, який іде в зворотному порядку від матових позицій або від позицій з відомою оцінкою.

За допомогою шахових програм було повністю проаналізовано ряд ендшпільів, наприклад, "*ферзь з пішаком проти ферзя*", "*тура і кінь проти тури*" та ін. Часто такий комп'ютерний аналіз змінював попередню теоретичну оцінку. Наприклад, закінчення "*тура і кінь проти тури*" в теорії вважалось нічийним, але комп'ютер виявив великий процент виграшних позицій. У Шаховому Кодексі є правило, згідно з яким партія закінчується внічию, якщо протягом 50 ходів не було жодного взяття і жоден пішак не зробив ходу. Для деяких типів позицій робиться виключення. На основі детального комп'ютерного аналізу було встановлено, що для зміни матеріального співвідношення у багатьох закінченнях 50 ходів не вистачає, і тому ці закінчення теж потрібно ввести до Кодексу як виключення. Шахові програми використовуються також для перевірки шахових задач, у яких вимагається поставити мат за визначену наперед кількість ходів. Завдяки цьому у багатьох таких задачах були виявлені грубі помилки.

Регулярно проводяться чемпіонати світу серед шахових програм. Перший з них відбувся у 1974 році, і його переможцем стала програма "**Kaïca**" (тодішній CPCR). Сила сучасних шахових програм продовжує зростати. Такі програми, як **Hiarch 6.0, Rebel 8.0, MChess Pro 6.0** при виконанні на процесорах **Pentium** показують досить високу якість гри. Можна згадати спеціалізований шаховий комп'ютер **Deep Blue** з серії **RS/6000**, який влітку 1997 року в шаховому матчі з 6 партій виграв у Каспарова з рахунком 3.5 на 2.5.

4.6. Деякі підходи до самонавчання ігрових програм.

Зрозуміло, що ігрові програми створюються не тільки для шахів. Так, дуже цікавою є програма для гри в шашки, створена Семюелем. Головною метою роботи

Семюеля була не стільки гра, як вивчення можливостей "самовдосконалення" і "самонавчання" ігрових програм [3].

Ідея одного з підходів Семюеля полягала у такому. В найпростішому випадку статична оціночна функція являє собою лінійний вираз

$$s = k_1 a_1 + k_2 a_2 + \dots + k_n a_n,$$

де a_i - числові оцінки факторів, що впливають на оцінку позиції, k_i - деякі вагові коефіцієнти. Якість гри визначається цими коефіцієнтами, і тому їх необхідно оптимізувати. Їх можна підігнати шляхом пробних ігор, а можна підібрати шляхом безпосереднього аналізу дерева гри.

Якби статична оціночна функція співпадала з оцінкою, переданою знизу, така оцінка була б ідеальною, а її отримання було б еквівалентне повному аналізу дерева. Нехай s - статична оцінка даної позиції; q - оцінка, передана знизу; $e = s - q$ - різниця між цими оцінками. Ідея Семюеля полягала в адаптивній мінімізації цієї різниці. Він визначав **коефіцієнти кореляції** між a_i та e . Якщо кореляція додатня, відповідний коефіцієнт k_i потрібно зменшити, якщо від'ємна - то збільшити. Подібні підходи часто називають *декореляцією помилок*.

4.7. Підхід людини до шахової гри

Загальновизнано, що реалізовані у сучасних шахових та інших ігрових програмах варіанти повного перебору є **прикладом "нелюдського" підходу до гри**: людина ніколи не робить такого перебору, до якого залучається величезна кількість безглузвих ходів. Шахіст швидко вловлює основні особливості позиції, він швидко помічає ходи, на які слід звернути увагу у першу чергу. У мозку шахіста виникає *дерево планів* [5]. Ці плани виникають на основі того, що в ході аналізу позиції шахіст має змогу поставити перед собою **певні цілі**. План має визначену кінцеву ціль, тому в першу чергу аналізуються ходи, які можуть сприяти або завадити реалізації цього плану. Такий підхід реалізований у програмі **ROBIN** [5].

Слід відзначити також дослідження гросмейстера, екс-чемпіона світу з шахів та доктора наук М.М. Ботвинника та його програму **"Піонер"** [Д.4]. У найбільш загальних рисах, **підхід ґрунтується на розгляді шахової гри на основі вирішення задачі багаторівневого керування з неточною метою**. Шахова гра розглядається як багаторівнева система. Кожна шахова фігура має певну мету, і інші фігури можуть допомагати або заважати їй у досягненні певних цілей. Ці локальні цілі повинні узгоджуватися з більш глобальними цілями (виграш, досягнення матеріальної переваги), і за цим слідує система керування вищого рівня.

Людська гра спирається на ряд **евристик**, застосування цих правил може різко скоротити перебір і часто дозволяє зробити розумний хід майже без аналізу варіантів. До таких шахових евристик відносяться, наприклад, вимоги боротьби за центр, швидкого розвитку фігур, своєчасної рокіровки і т.п. Зрозуміло, що стратегія, яка спирається на такі правила, далеко не завжди є оптимальною. Вона може бути субоптимальною, або просто раціональною.

Незважаючи на певні досягнення у цій галузі, спроби ухилитися від повного перебору в шахових програмах до цього часу носили експериментальний характер, і ці програми не досягали великих спортивних успіхів.

Крім розглянутої гри з суперником, який завжди намагається робити ходи, що є для нас найгіршими, можна розглядати такі позиційні ігри: *гра з природою, яка робить випадкові ходи; гра з партнером, який робить найкращі для нас ходи* (така гра часто називається *кооперативною*).

Можна також говорити про *рефлексію*, тобто про врахування намірів і можливостей суперника. Так, гравець, роблячи ризикований хід, може сподіватись на те, що суперник не знайде найкращу відповідь. Гра у такому випадку набуває рис імовірнісного, азартного характеру.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Поясніть зв'язок між ігровими задачами і плануванням цілеспрямованих дій.
2. Назвіть характерну особливість ігрових задач, які вирізняють їх серед інших задач прийняття рішень.
3. Дайте поняття дерева гри; побудуйте дерево довільної конкретної гри.
4. Напишіть програму, яка грає в “хрестики-нулики” за допомогою мінімаксної процедури. Програма має запитувати у користувача, хто має робити перший хід. Перебіг гри повинен відображатися на екрані в графічному режимі.
5. Напишіть програму для гри в калах на основі усікненої мінімаксної процедури з альфа-бета-відтинанням.
6. Проілюструйте на прикладі довільного дерева гри мінімаксну процедуру на повному дереві, мінімаксну процедуру на усікненому дереві та альфа-бета-відтинання. Покажіть на цьому ж прикладі вплив глибини перебору на якість гри.
7. Наведіть відому Вам теоретичну оцінку максимального скорочення перебору при альфа-бета-відтинанні. Покажіть на прикладі, як порядок перебору ходів-альтернатив впливає на ефективність альфа-бета-відтинання.
8. Покажіть на дереві гри позиції, які аналізуються при максимальних альфа-бета-відтинаннях.
9. Покажіть, як можна побудувати І-АБО-граф для довільної ігрової задачі.
10. Дайте загальну характеристику сучасних шахових програм.
11. Чим відрізняється підхід, реалізований у сучасних шахових програмах, від підходу людини-шахіста до гри?

Розділ 5. Конекціоністські методи

*Гуртом і батька легше бити
(Народне прислів'я)*

5.1. Загальна характеристика конекціоністського підходу та його місце в теорії інтелектуальних систем

Конекціоністський підхід до побудови систем штучного інтелекту розвинувся на противагу символічному, що є характерним для сучасних моделей знань. В основі конекціоністського підходу лежить спроба безпосереднього моделювання розумової діяльності людського мозку. Відомо, що мозок людини складається з величезної кількості нервових клітин (нейронів), що взаємодіють між собою. Ці

“обчислювальні елементи” мозку функціонують набагато повільніше, ніж обчислювальні елементи комп’ютерних систем. Але ефективність людського інтелекту досягається як за рахунок паралельної роботи нейронів, так і за рахунок того, що механізми їх взаємодії були вироблені шляхом тривалої еволюції.

Для багатьох цілей нейрон можна розглядати як елемент з певним критичним значенням. Це означає, що він або ж дає на виході деяку постійну величину, якщо сума його входів досягає певного значення, або ж залишається пасивним.

Мак-Каллок і Пітс довели, що будь-яку обчислювану функцію можна реалізувати за допомогою спеціально організованої мережі ідеальних нейронів, логічні властивості яких з високою достовірністю можна приписати реальному нейрону. Але ця мережа буде мати наступні вади. По-перше, проблема полягає в тому, чи можна знайти якийсь розумний принцип реорганізації мережі, який дозволяв би випадково об’єднаній, спочатку, групі ідеальних нейронів самоорганізовуватись в “обчислювальний пристрій”, здатний вирішувати довільну задачу розпізнання. По-друге, потрібно використовувати велику кількість нейронів. Так, модель мурашки потребує використання близько 20000 нейронів, людини - 100 млрд. нейронів, що на практиці неможливо.

Нейрологічна теорія стала також основою системи розпізнавання, яка дістала назву *перцептрон*. В цьому підході основна увага приділялась встановленню характеристик, приписаних фіксованій множині детекторів ознак. Альтернативний підхід розпізнавання зводиться до пошуку “добрих” ознак, на основі яких розпізнавання здійснюється найбільш чітко. Наприклад, перцептрон Розенблатта передавав повідомлення від “ока”, яке реалізовувалось системою фотоелементів, в блоки електромеханічних комірок пам’яті, які оцінювали відносні величини електричних сигналів. Ці комірки з’єднувались між собою випадковим чином, створюючи мережу з прямими зв’язками. Зазначимо, що в ній були відсутні зворотні зв’язки між нейроноподібними елементами. Перцептрон міг навчатись шляхом спроб і помилок, а також корекцією електричних імпульсів.

Мінським і Пейпертом в кінці шестидесятих років було математично доведено, що перцептрони не в змозі виконувати багато приписуваних їм функцій, наприклад, розпізнавання частково затулених предметів. Після цього результату розвиток перцептронної теорії призупинився.

З часом, Мінський виражав жаль з приводу опублікування результату. Він вважав, що для реального прориву в практиці створення систем штучного інтелекту необхідний пристрій, подібний до перцептрона.

Так, один з сучасних напрямків створення розумних машин - це *нейрокомп’ютер*. Нейрокомп’ютер - це програмно-технічна система (спеціалізована ЕОМ), яка реалізує деяку формальну модель природної мережі нейронів.

В основу машин п’ятого покоління покладено ідею паралельної обробки інформації в нейроноподібних системах. Не зважаючи на те, що електронний процесор працює в тисячі разів швидше, ніж його нейронний еквівалент у мозку, мережі нейронів вирішують багато задач (особливо нечислових) в тисячі раз швидше, ніж електричний процесор.

Причини цього такі:

1) Характер взаємозв’язків між нейронами дозволяє розв’язувати багато задач на основі паралельної обробки;

2) У нейронній мережі пам'ять не локалізована в одному місці (як в послідовних машинах), а розподілена по всій структурі. В біологічних системах пам'ять реалізується підсиленням або послабленням зв'язків між нейронами, а не зберіганням двійкових символів;

3) Біологічні мережі реагують не на всі, а тільки на визначені зовнішні подразнення. Кожний нейрон виступає як елемент прийняття рішення і як елемент зберігання інформації. Перевага такої структури - "життєздатність" (вихід з ладу декількох нейронів не приводить до значної зміни даних, що зберігаються, або ж до руйнування всієї системи).

4) Можливість адресації за вмістом (асоціативної пам'яті) є ще однією важливою характеристикою систем з розподіленою пам'яттю (кожний елемент відшукується за його вмістом, а не зберігається в комірці пам'яті з визначеним номером).

В основу зв'язків в нейрокомп'ютерах покладено *принцип асоціацій*. Асоціативні зв'язки пронизують все мислення людини. Існує думка, що **процеси мислення є не що інше, як розповсюдження певного збудження, як деяка ланцюгова реакція**. Навіть найбільш примітивні процеси навчання принципово залежать від послідовності подій в часі. Це й закладено в природу нейронних систем. Тому їм притаманне реагування тільки на жорстко визначені зовнішні подразнення. Наприклад, домашні тварини "навчаються" ігнорувати повторні несуттєві зовнішні подразнення ("цокання" годинника), але посилюють сприйняття подразнень, які можуть мати серйозні наслідки (звук автомобільних гальм).

Тому майбутнє належить комп'ютерам, які базуються на аналізі зв'язків, а не обробці символів. Мінський говорив, що якщо комп'ютер повинен діяти подібно мозку, тоді й його конструкція повинна бути також подібна до мозку.

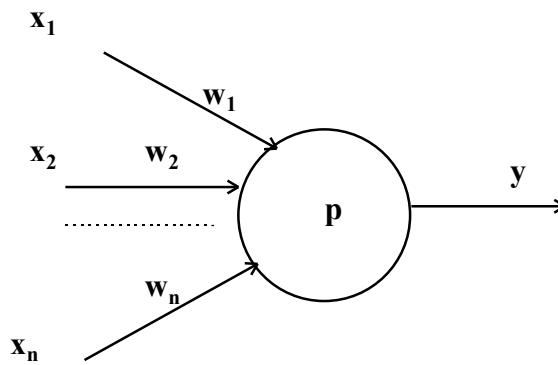
Моделі нейронних мереж і схеми з адресацією за вмістом мають і недоліки. Внаслідок нефіксованої організації вони можуть плутати різні об'єкти. Але це аналогічно звиканню, посиленню чуттєвості до асоціацій, які лежать в основі психічних особливостей людини. Іншими словами, будь-який комп'ютер, який претендує на "розумність" повинен мати такі особливості.

6.2. Модельні нейрони як порогові елементи

У своїй піонерській роботі Мак-Каллок і Піттс розглянули модельний нейрон як деякий пороговий елемент з n входами та одним виходом. Кожному входові приписується вага w_i , яка може бути як додатною, так і від'ємною.

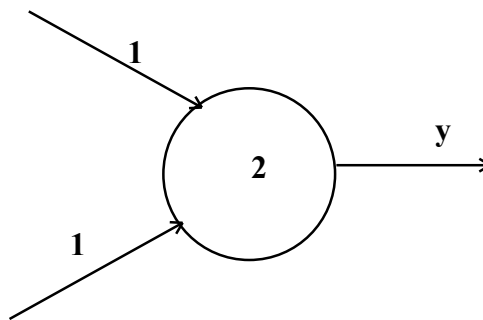
Нехай на вхід модельного нейрону подаються сигнали x_i , $i=1,...,n$. Вихід нейрона у найпростішому випадку дорівнює **1**, якщо $S \geq p$ та **0**, якщо $S < p$. Тут p - деякий фіксований поріг,

$$S = \sum_{i=1}^n w_i x_i.$$

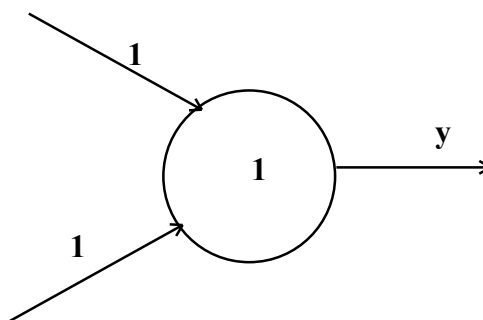


Мал. 5.1. Простий модельний нейрон

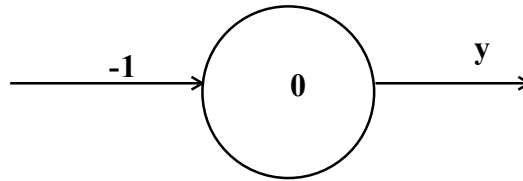
Мак-Каллок і Піттс довели, що обчислення будь-якої обчислюваної за Тьюрингом функції можна реалізувати за допомогою спеціально організованої мережі модельних нейронів. Цей результат стає більш зрозумілим, якщо розглянути порогові елементи, які реалізують три базові логічні функції: кон'юнкцію, диз'юнкцію та заперечення.



Мал. 5.2. Пороговий елемент, що реалізує кон'юнкцію



Мал. 5.3. Пороговий елемент, що реалізує диз'юнкцію



Мал. 5.4. Пороговий елемент, що реалізує заперечення

Згодом стали розглядати нейрони, які реалізують нелінійні перетворення.

Коефіцієнти w_i майже ніколи не бувають відомими заздалегідь. Важливим є те, що ці коефіцієнти спочатку можуть бути розставлені довільно, і **процес навчання нейронної мережі полягає в зміні цих коефіцієнтів за тими чи іншими правилами**. Основною проблемою при цьому є пошук ефективних алгоритмів такого навчання.

5.3. Пандемоніум Селфріджа

Однією з перших реалізацій нейронних мереж став *пандемоніум Селфріджа*. Була запропонована методика його застосування для розпізнавання зразків.

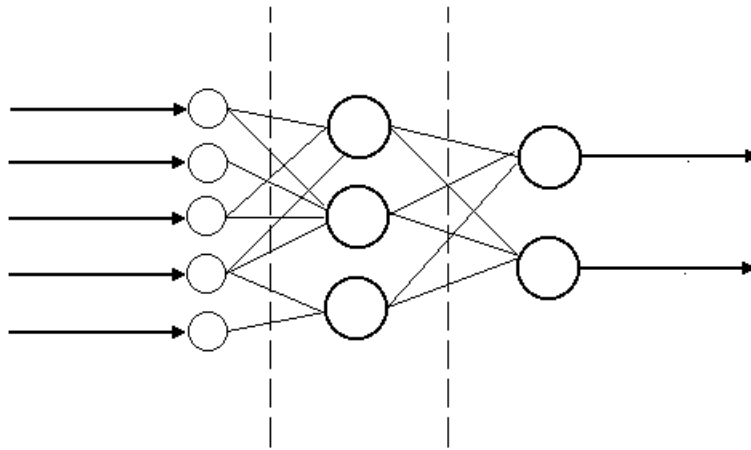
На найнижчому рівні пандемоніуму Селфріджа знаходяться **демони даних**, які безпосередньо сприймають зовнішній світ. Демони даних зв'язані з **демонами розуміння**; кожний демон розуміння відповідає певному класові або поняттю.

Кожний демон розуміння повинен визначити, наскільки схожим з еталоном є зразок, що розпізнається. Чим більшим є ступінь схожості, тим сильніший сигнал надсилається до **демону рішення**. Останній, в свою чергу, вибирає найсильніший сигнал.

Часто цю схему модифікують так, що між демонами даних і розуміння розміщуються **обчислювальні демони**, які виділяють характерні ознаки об'єктів, що розпізнаються.

5.4. Перцептрон

Типова схема перцептрону подана на мал. 5.5. Історично першим застосуванням перцептрону було розпізнавання зразків, і тому ми обмежимося тільки цією практичною задачею.



Мал. 5.5. Схема перцептрону.

Найпростіший перцептрон складається з трьох шарів. На першому шарі розташовані чутливі рецепторні елементи (**S -елементи**), на які подаються вхідні зображення. **S -елементи** пов'язані з елементами другого шару (**A -елементами**). **A -елемент** збуджується, тобто генерує вихідний сигнал, тоді і тільки тоді, коли збуджується достатня кількість зв'язаних з ним **S -елементів**.

A -елементи з'єднуються з **R -елементами** (вихідними елементами) дугами зі змінними ваговими коефіцієнтами. Показаний на мал. 5.4 перцептрон відповідає випадкові двох класів. Якщо вихід елемента **R₁** перевищує вихід елемента **R₂**, то система відносить зразок до першого класу, а якщо навпаки - тоді до другого.

Навчання перцептрону відбувається шляхом зміни вагових коефіцієнтів. Якщо класифікація правильна, посилюються зв'язки, що йдуть від тих **A -елементів**, які спрацювали. Навпаки, при неправильній класифікації коефіцієнти зв'язків, що йдуть від елементів, які спрацювали, зменшуються.

Виявилося, що персептронні моделі можна застосовувати для розпізнавання простих зразків. Але початкова ейфорія, пов'язана з персептронами, швидко закінчилася, коли з'ясувалося, що можливості цих пристроїв дуже обмежені. У [Д.6] наводяться результати експериментів з перцептроном, який відрізняв трикутники від кіл з рівнем помилок 33% і прямокутники від еліпсів з рівнем помилок 50%, що еквівалентно випадковому вгадуванню. Ще більш сильні результати, а саме теоретичне доведення того факту, що персептрони при досить слабких обмеженнях в принципі не можуть розрізнити деякі класи об'єктів, наводиться в [Д.1].

Можливості перцептрону були детально проаналізовані математично. Сформульована **теорема збіжності**. Згідно з цією теоремою, якщо в принципі існують такі набори вагових коефіцієнтів, при яких система відрізняє представників одного класу від представників іншого, то алгоритм навчання перцептрону дозволяє отримати один з цих наборів.

Інше питання -, які саме зразки можна розрізнити між собою. Мінський і Пейперт довели, що прості персептрони мають дуже обмежені можливості. Зокрема, вони не можуть розпізнавати частково затулені об'єкти.

Після цього розвиток персептронної теорії дещо призупинився, і Мінський навіть висловлював жаль з приводу опублікування цього результату.

5.5. Сучасні нейронні мережі

Згодом розвиток нейронних мереж знову пожвавився. Були запропоновані нові схеми організації цих мереж, а також нові алгоритми їх навчання.

Класичні перцептрони добре працюють у випадку **лінійної роздільності** класів. Але Хеч-Нільсен на основі одного з результатів Колмогорова довів, що, якщо додати до перцептрону ще один шар, то вимога лінійної роздільності перестає бути необхідною.

Тим не менше, розвиток теорії багатошарових нейронних мереж стримувався через відсутність ефективних алгоритмів їх навчання. Такі алгоритми були запропоновані в 60-х - 70-х роках, і в їх основу ліг **принцип зворотного розповсюдження помилок**.

Згідно з [Змитр], загальна схема навчання зі зворотним розповсюдженням помилок має такий вигляд:

1. **Обрати черговий навчальний вектор і подати його на вхід мережі.**
2. **Обчислити вихід мережі.**
3. **Обчислити різницю між фактичним і бажаним виходами.**
4. **Скоректувати вагові коефіцієнти так, щоб мінімізувати помилку.**
5. **Повторювати процедуру, поки помилка не досягне прийнятної величини.**

Ця схема постійно удосконалювалася. Були запропоновані інші типи мереж: мережі зустрічного розповсюдження, мережі зі зворотними зв'язками тощо.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Охарактеризуйте суть конекціоністського підходу до побудови інтелектуальних систем.
2. Дайте визначення порогового елемента.
3. Поясніть, як можна реалізувати основні логічні елементи традиційного комп'ютера (інвертор, схему І, схему АБО, тригер) на основі порогової логіки.
4. Опишіть схему пандемоніума Селфріджа.
5. Опишіть схему класичного перцептрона Розенблатта.
6. Охарактеризуйте основні можливості та основні обмеження класичного перцептрону.
7. Напишіть програму, яка реалізує розпізнавання довільних двох класів на основі перцептрону Розенблатта.
8. Опишіть у загальних рисах процедуру зворотного розповсюдження помилок.

ЧАСТИНА III. ПРЕДСТАВЛЕННЯ ЗНАНЬ ТА МЕТОДИ ЛОГІЧНОГО ВИВЕДЕННЯ

Розділ 6. Знання як інформаційна основа інтелектуальних систем

*Інтелектуальне казіно "Що, де, коли?" є єдиним місцем, де кожен може заробити гроші завдяки власному розумові
(В.Ворошилов)*

6.1. Інтуїтивне поняття знань

На інтуїтивному рівні знання можна визначити як інформацію, на основі якої можна отримувати нову інформацію, тобто нові знання.

Можна дати таке неформальне визначення знань.

Визначення. Знаннями інтелектуальної системи називається трійка $\langle F, R, P \rangle$, де F - сукупність **фактів**, що зберігаються в пам'яті системи в явному вигляді, R - сукупність **правил**, які дозволяють на основі існуючих фактів отримувати нові, P - сукупність **процедур**, які визначають, яким чином слід застосовувати правила.

Знання, доступні системі, складають **базу знань** цієї системи.

Приклад. Розглянемо таку базу знань.

Ф1. Заєць їсть траву.

Ф2. Вовк їсть м'ясо.

Ф3. Заєць є м'ясом.

П1. Якщо A є м'ясом, а B їсть м'ясо, то B їсть A .

П2. Якщо A їсть м'ясо, то A є хижаком.

П3. Якщо A їсть B , то B є жертвою.

П4. Якщо хижак вмирає, жертва експоненційно розмножується.

П5. Якщо жертва вмирає, хижак вмирає.

Даний приклад фактично є фрагментом неформалізованого опису моделі Лотки-Вольтерра, добре відомої в екології. Відмітимо, що **подібна база знань може лягти в основу простої експертної системи.**

Літерами **Ф** позначені **факти**, які відображають елементарні знання; і їх безпосередній аналіз дозволяє експертній системі відповідати на найпростіші запитання, такі, як "*Чи їсть вовк м'ясо?*". Для **позитивної відповіді на це та подібні йому запитання досить просто знайти відповідний факт у базі знань.**

Принципово іншим є запитання типу "*Чи є вовк хижаком?*". Відповідь на нього простим пошуком знайти неможливо: такого факту в базі знань немає. Тому для **відповіді на це запитання необхідно застосовувати правила, які дозволяють на основі існуючих фактів отримувати нові знання, тобто нові факти.** Такі правила позначені у нашому прикладі літерою **П**. Так, для відповіді на поставлене запитання досить знання факту **Ф2** і правила **П2**.

Безумовно, до бази знань можна було б зразу включити твердження "*Вовк є хижаком*". Але легко побачити, що, якщо, крім вовків розглядати інших хижаків (ведмедів, лисиць та ін.), то безпосередньо включати до бази знань факти, що вони є хижаками, явно недоцільно. Взагалі, все, що може бути виведене логічним шляхом, не варто оголошувати як факти.

Нарешті, **процедури** визначають, яким саме чином слід застосовувати правила. Часто ці процедури є складовою частиною мови, якою програмується експертна система (це відноситься в першу чергу до спеціалізованих мов штучного інтелекту, таких як **Лісп, Пролог, Пленер**). Якщо ж таких процедур недостатньо, програміст має сам подбати про їх написання.

З формалізацією знань пов'язано багато проблем, деякі з них можна проілюструвати уже на цьому простому прикладі.

Проблема виключень. Правило П1, взагалі кажучи, невірне: з того, що хтось їсть м'ясо, зовсім не випливає, що він їсть саме зайців. Ще більш показовий такий приклад: на перший погляд, ніхто нібито не піддає сумніву твердження "**якщо А - птах, то А літає**". Саме у такому вигляді його, як правило, формулюють люди-експерти, і саме в такому вигляді воно може потрапити до бази знань. Але, якщо відомо, що А - пінгвін, висновок стає невірним.

Проблема неповних знань. Іноді може виникнути така ситуація. З'ясовується питання про те, чи призведе завіз мангустів в деяку країну до зникнення кролів, в ситуації коли твердження про те, *чи їдять мангусти кролів*, у базі знань немає. У таких випадках бажано, щоб добра експертна система не просто відповіла "*Ні*", а спитала про наявність даних, яких не вистачає.

Проблема недостовірних знань. Експерт може бути не зовсім впевненим у тому чи іншому факті.

Проблема невизначеності. Нехай відомо, що певний набір симптомів свідчить про наявність деякої хвороби з імовірністю 50%, а лікування цієї хвороби даним препаратом призводить до одужання у 90% випадків. У таких випадках при логічних міркуваннях слід враховувати коефіцієнти визначеності; експертна система, що враховує такі коефіцієнти, може прийти до висновку, що лікування даним препаратом буде ефективним з імовірністю 45%.

Проблема зміни ситуацій. Некоректне застосування правил П4 і П5 у нашому прикладі може призвести до логічної помилки. Якщо хижак вимре внаслідок того, що вимерла жертва (це відображено у правилі П5), надалі говорити про те, що "*якщо хижак вмирає, жертва експоненційно розмножується*" (правило П4), не має ніякого сенсу.

6.2. Перехід від даних до знань

Сучасний етап розвитку інформатики характеризується розвитком моделей даних у напрямку переходу від традиційних моделей (реляційна, ієрархічна, мережна кодасілівська) до моделей знань, і, відповідно, від моделей даних до моделей знань. Стали розрізняти **екстенсіональні** та **інтенсіональні** представлення. Під екстенсіональним представленням мається на увазі просто набір фактів, або кортежів, що входять до бази даних. Іntenсіональне представлення - це задання правила, якому підпорядковуються записи у базі даних. Іntenсіональне відношення можна формально визначити як відношення **$R(A_1, \dots, A_n)$** . Вважається, що до бази даних, що описується відношенням **R** , належать ті і тільки ті факти, які задовольняють йому. Якщо в чисто екстенсіональній базі даних всі факти необхідно зберігати в явному вигляді, то при наявності інтенсіональних представлень це не обов'язково. Нові факти можуть бути породжені відношенням

Р, за його ж допомогою може бути перевірена істинність того чи іншого твердження. Розвинені інтенціональні представлення дають змогу говорити не про дані, а про знання, і, відповідно, не про бази даних, а про бази знань.

Були запропоновані **засоби структуризації знань**, такі, як **узагальнення та агрегація** [64]. Узагальнення задає ієрархію типів (екземпляр - тип - надтип). Так, наприклад, можна говорити, що об'єкт **Вася Петров** відноситься до типу **Студент**, тип **Студент**, в свою чергу, є підтипом типу **Людина** і т.п. Надтип об'єднує атрибути, або ознаки, спільні для його підтипів; підтипи можуть наслідувати ті чи інші властивості підтипів.

За допомогою агрегації об'єкт визначається як конструкція, що складається з деяких складових частин. Так, наприклад, за допомогою агрегації в базі знань можна вказати, що аудиторія має вікна, двері і т.п.

6.3. Властивості та класифікація знань

В [1] сформульовані такі особливості знань:

- **внутрішня інтерпретованість:** кожна інформаційна одиниця повинна мати унікальне ім'я, за яким інформаційна система її знаходить, а також відповідає на запити, в яких це ім'я згадується;
- **структурованість:** знання повинні мати гнучку структуру; одні інформаційні одиниці можуть включатися до складу інших (відношення типу "частина-ціле", "елемент-клас");
- **зв'язність:** в інформаційній системі повинна бути передбачена можливість встановлення різних типів зв'язків між різними інформаційними одиницями (причинно-наслідкові, просторові та ін.);
- **семантична метрика:** на множині інформаційних одиниць корисно задавати відношення, які характеризують ситуаційну близькість цих одиниць;
- **активність:** виконання програм в інтелектуальній системі повинно ініціюватися поточним станом бази знань.

Згідно з класифікацією Попова Е.З. [1], знання класифікуються за областями і за типами знань. Можна виділити такі області знань: **проблемна область, область мови, область системи, область користувача, область діалогу**. Чим більше в нас областей знань, тим більша можливість інтерпретації вхідних висловлень.

Спосіб задання знань включає два аспекти: спосіб організації знань і модель задання. Вони різняться рівнями задання і рівнем детальності. За рівнями задання виділяють знання нулевого рівня (конкретні і абстрактні знання) і знання більш високих рівнів (метазнання). Перший рівень складають знання про задання знань нулевого рівня. Число рівнів може бути продовжено. Розділення знань по рівням задання забезпечує можливість точного налаштування і адаптації інтелектуальної системи.

Організація знань за рівнями детальності дозволяє розглядати знання з різним ступенем деталізації. Кількість рівнів деталізації залежить від специфіки задач вирішення, обсягу знань і способу їх задання. Традиційно виділяють три рівні: відображення загальної організації знань, логічна і фізична організація окремих структур знань. Введення рівнів деталізації забезпечує гнучкість системи за рахунок ізоляції змін одного рівня від іншого.

6.4. Загальна характеристика моделей задання знань

Людське і машинне сприйняття значно відрізняються. Тому необхідно використовувати моделі знань, які дають можливість представляти знання в системах штучного інтелекту. При проектуванні моделі задання знань необхідно враховувати такі фактори, як *однорідність задання і простоту розуміння*. Однорідність задання приводить до спрощення механізму керування логічним виведенням і спрощення керування знаннями. Задання знань повинно бути зрозумілим експертам і користувачам системи. Для складних задач необхідне структурування і модульне задання.

Моделі задання знань діляться на логічні і евристичні в залежності від рівня декларативності (процедурності) мовних засобів, які використовуються для опису знань. Поєднання обох типів задання знань привело до створення змішаного задання, наприклад, декларативне задання знань з приєднаними процедурами виведення (фреймові моделі, семантичні мережі).

Типовими моделями задання знань є: *логічна модель, продукційна модель, модель, що базується на використанні фреймів, модель семантичної мережі*

ЗАПИТАННЯ ТА ВПРАВИ ДО РОЗДІЛУ

1. Наведіть інтуїтивне визначення поняття знань.
2. Охарактеризуйте поняття фактів, правил та процедур.
3. Наведіть приклад бази знань з довільної предметної області, що включає до свого складу факти та правила.
4. Опишіть відомі Вам проблеми, пов'язані з базами знань.
5. У чому полягає різниця між екстенсіональними та інтенсіональними представленнями даних та знань?
6. Охарактеризуйте поняття узагальнення та агрегації.
7. Перелічіть основні властивості знань.
8. Перелічіть відомі Вам моделі задання знань.

Розділ 7. Мережні та фреймові моделі знань.

*У него повсюду связи -
В магазине и на базе...
(З пісні)*

7.1. Семантичні мережі

В основі мережних моделей лежить конструкція, що називається **семантичною мережею**. Формально ці моделі можна задати у вигляді $\langle I, C_1, C_2, \dots, C_n, \Gamma \rangle$. Тут I - множина інформаційних одиниць, $C_1, C_2, \dots, C_n$ - типи зв'язків між інформаційними одиницями, Γ - відображення, що задає зв'язки між інформаційними одиницями.

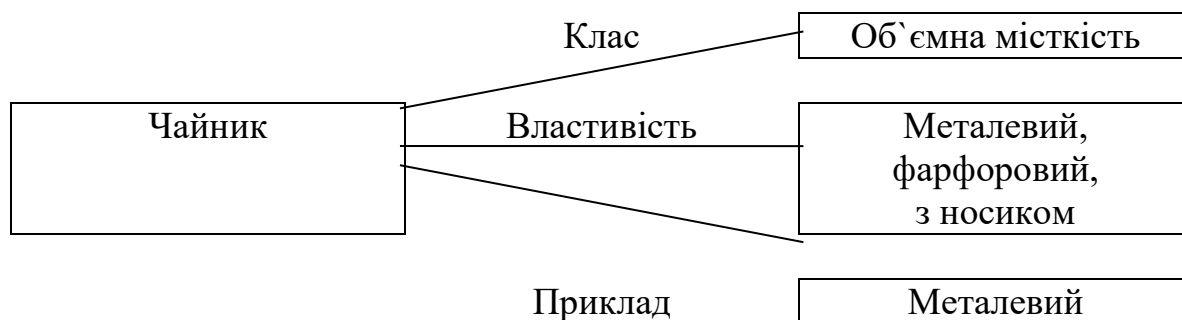
Мережні моделі тісно пов'язані з моделями "**сутність-зв'язок**". Під сутністю розуміється об'єкт довільної природи. Об'єкт, що існує в реальному світі, називається **П-сутністю**. Його представлення в базі знань називається **М-сутністю**. У базі знань відображаються також зв'язки між сутностями.

Виділяють **класифікуючі** мережі, **функціональні** мережі та **сценарії**. Класифікуючі мережі дозволяють задавати відношення ієрархії між інформаційними одиницями. Функціональні мережі характеризуються наявністю функціональних відношень, які дозволяють описувати процедури обчислень одних інформаційних одиниць через інші. У сценаріях використовуються відношення типу "причина-наслідок", "дія", "засіб дії" та ін.

Під семантичною мережею розуміється мережа, в якій поєднані зв'язки різних типів. **Семантичну мережу можна неформально уявляти собі у вигляді графа, вершини якої, як правило, позначають об'єкти предметної області, а дуги відповідають зв'язкам між ними.**

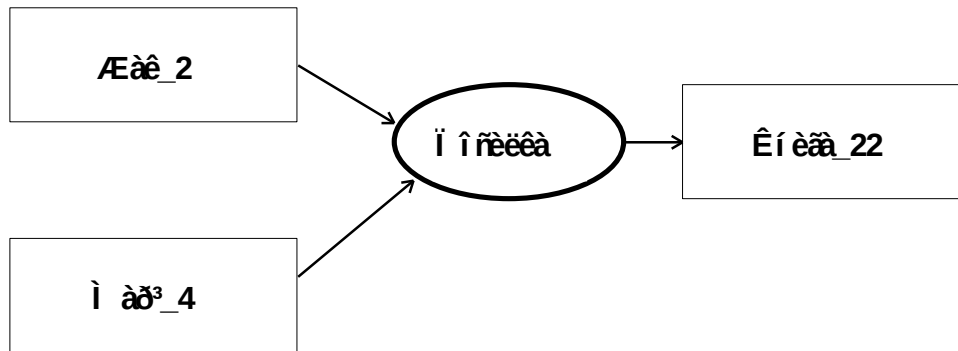
Запропоновано ряд формалізацій для представлення знань у вигляді семантичних мереж. Ми будемо в основному спиратися на дослідження Куїлліана та Сови, спираючись у викладенні на роботи [9, 10]. При цьому досить часто буде вживатися більш сучасна термінологія.

Базовим є поняття **концептуального об'єкту**, який графічно зображається **концептуальним графом**. У моделі Куїлліана П-сутність описується концептуальним об'єктом, причому до опису включаються клас, до якого належить П-сутність, властивості сутності та його прототип (або приклад). Як приклад можна навести концептуальний граф для чайника.



Мал. 7.1. Приклад концептуального графа

Більш загально, у вигляді концептуального графа можна представляти будь-яку логічну формулу або словесне висловлювання. Такий концептуальний граф можна вважати простою семантичною мережею, що відповідає одному твердженню. Так, наприклад, висловлювання **"Жак надсилає книгу Марі"** може бути представлене у вигляді предикату **ПОСИЛКА(ЖАК_2, МАРІ_4, КНИГА_22)**, а концептуальний граф матиме вигляд:



Мал. 7.2. Приклад простої семантичної мережі, що відповідає одному фактові

Цифри потрібні для того, щоб дати кожній сутності своє індивідуальне ім'я, яке не співпадає ні з якими іншими.

Прямокутники концептуального графа призначені для представлення аргументів предикатів, а кола - для представлення самих предикатів. Круг і прямокутник з'єднуються дугою, якщо вони є відповідно ім'ям та аргументом того самого предиката.

Для спрощення вигляду концептуального графа можна не виділяти імена предикатів у вигляді окремих кругів, а писати ці імена прямо на дугах.

Часто буває корисним представити предикат, що відповідає деякій складній фразі природної мови, у вигляді кон'юнкції бінарних предикатів. Бінарним називається предикат, що має два аргументи. У нашому прикладі предикат ПОСИЛКА (ЖАК_2, МАРІ_4, КНИГА_22) можна переписати у вигляді

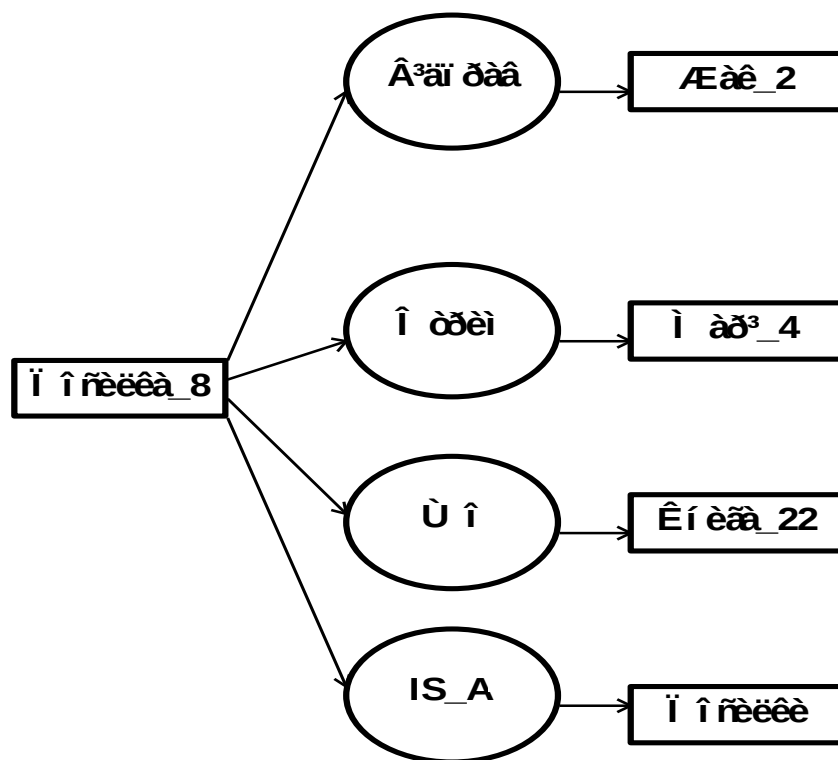
Відправник (Посилка_8, Жак_2)

Отримувач (Посилка_8, Марі_4)

Що (Посилка_8, Книга_22)

IS_A (Посилка_8, Посилки).

Відповідний концептуальний граф матиме вигляд:



Мал. 7.3. Концептуальний граф у бінарній формі

Зверніть увагу на предикат **IS_A**. Цей предикат відіграє винятково важливу роль та **означає належність деякої сутності до певного типу**.

Бінарне представлення часто називають представленням "**об'єкт-атрибут-значення**". "**Об'єкт**" є ім'ям бінарного предикату, або, в іншій інтерпретації, назвою сутності, що описується. "**Атрибут**" є назвою деякої ознаки, або властивості цього об'єкта. "**Значення**" відповідає конкретному значенню цієї властивості.

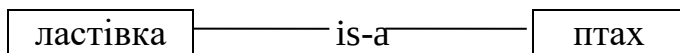
Представлення "**об'єкт-атрибут-значення**" є дуже важливим для теорії штучного інтелекту. Воно використовується і в інших моделях представлення знань та є базовим для ряду реально існуючих інтелектуальних систем.

Семантична мережа - це композиція концептуальних графів, об'єднаних за тими чи іншими правилами. Деякі з правил такого об'єднання (кон'юнкція, спрощення та ін.) описані в [1, 9, 10]. Там же наводяться приклади того, як враховувати квантори існування та узагальнення.

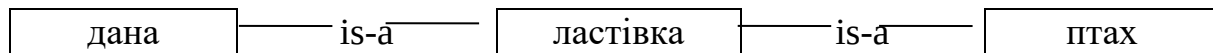
Багато методів логічного виведення на семантичних мережах ґрунтуються на зв'язку **IS_A** та понятті **наслідування**, що тісно з ним пов'язане. **Сутність наслідує атрибути типу, до якого вона належить. Якщо для типу визначені конкретні значення атрибутів, сутність, що належить до цього типу, наслідує їх за замовченням**

Зв'язок PART-OF показує відношення "**ціле - частина**". Цей зв'язок показує відношення між екземплярами класу.

Як приклад розглянемо речення, яке задає факт "**всі ластівки - птахи**". Цьому реченню може відповідати така семантична мережа:



Якщо далі присвоїти ластівці ім'я "Дана", тоді мережу можна розширити наступним чином:



Мал. 7.4. Ілюстрація наслідування на семантичній мережі

Тоді з семантичної мережі можна вивести речення: Дана - птах.

З наслідуванням пов'язано ряд проблем, основна з них - **механізм наслідування не завжди гарантує правильність висновку**. Це можна проілюструвати на такому прикладі.



Мал.7.5. Ілюстрація проблем, пов'язаних з наслідуванням

Так в мережі, зображеній на мал. 5.5, можна вивести неправильне заключення : "Дана вивчається натуралістами". Крім того, можна вивести висновок "Дана має крила", який є неправильним, якщо Дана є, наприклад, пінгвіном.

7.2. Фрейми

Фреймові моделі базуються на теорії фреймів, розроблених у 1975 р. М.Мінським. Опис цих моделей буде вестись в основному на основі [10].

Слово "фрейм" походить від англійського слова frame, яке перекладається як "рамка", "каркас". Мінський запропонував гіпотезу, згідно з якою знання групуються в модулі, і назвав фреймами ці модулі. Фрейм можна інтуїтивно уявляти собі як каркас, на якому тримаються знання, або як рамку, на основі якої описуються різноманітні об'єкти та ситуації. Мінський писав, що коли людина потрапляє в нову ситуацію, вона **співставляє** цю ситуацію з тими фреймами, які зберігаються у неї в пам'яті.

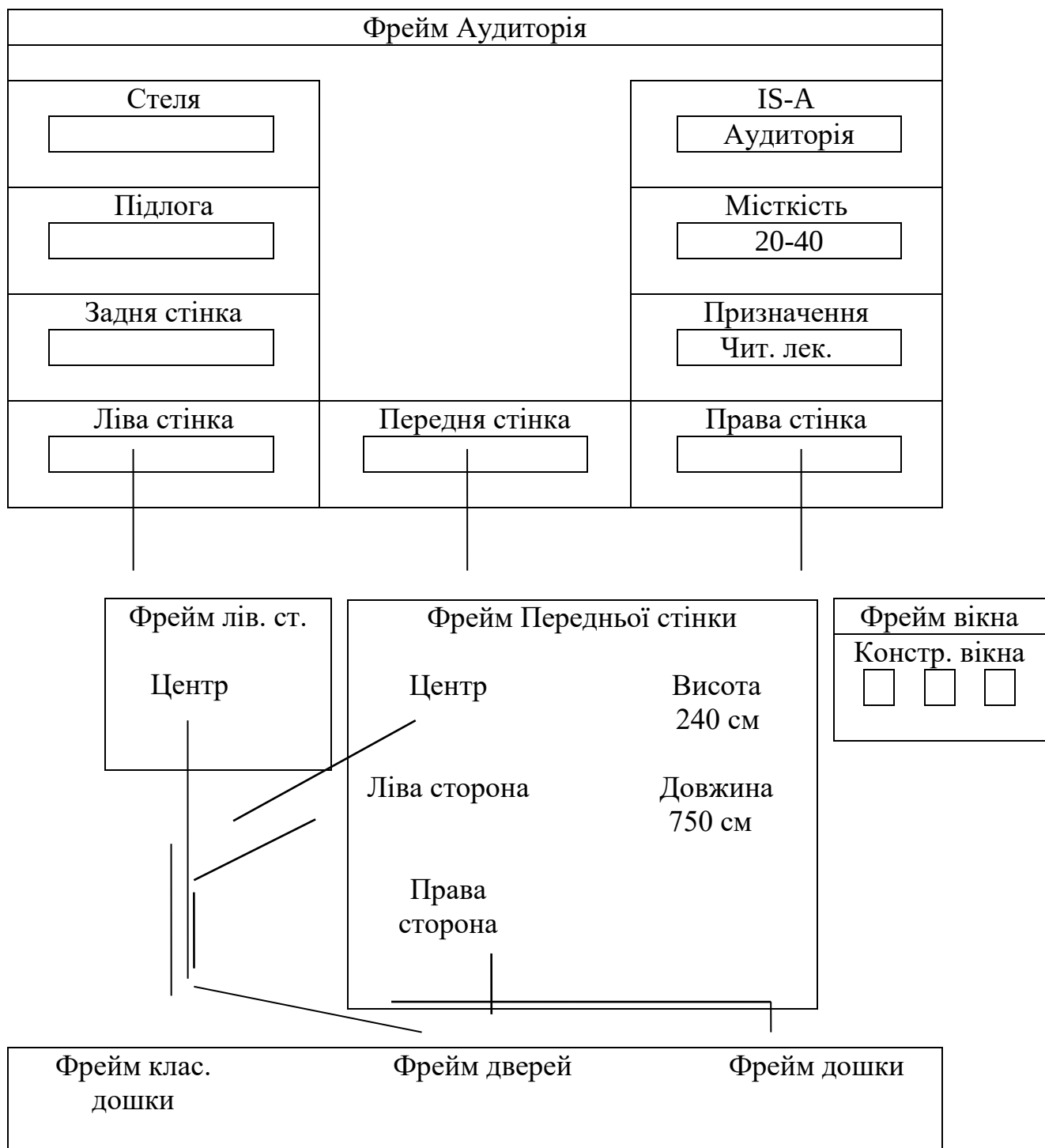
Фрейм можна визначити як структуру даних, що призначена для опису типових ситуацій або типових понять. М. Мінський визначав *фрейм як мінімальний опис деякої сутності, такий, що подальше скорочення цього опису призводить до втрати цієї сутності*.

Розглянемо, наприклад, поняття "Студент", яке описується відповідним фреймом. Кожний студент може бути охарактеризований такими характеристиками, як прізвище, ім'я та по-батькові, факультет, курс. Ці характеристики у фреймовій моделі представляються **слотами**.

На основі фрейму "Студент" може бути описаний будь-який конкретний студент. Цей опис відбувається шляхом заповнення слотів конкретними значеннями. Можна сказати, що **екземпляри понять утворюються в результаті конкретизації фреймів понять.**

Для фреймових моделей характерна *ієрархія понять*. У нашому прикладі фрейм "Студент" наслідує слоти від більш загального фрейму "Людина". Тому, якщо фрейм "Людина" описаний, відповідні слоти в описі фрейму "Студент" можна не задавати. **Якщо ж якісь слоти уже заповнені конкретними значеннями, то ці значення можуть успадковуватися фреймами-нащадками.**

За допомогою фреймів можуть описуватися і більш складні об'єкти. Розглянемо, наприклад, опис аудиторії [10] на основі фрейму "Аудиторія":

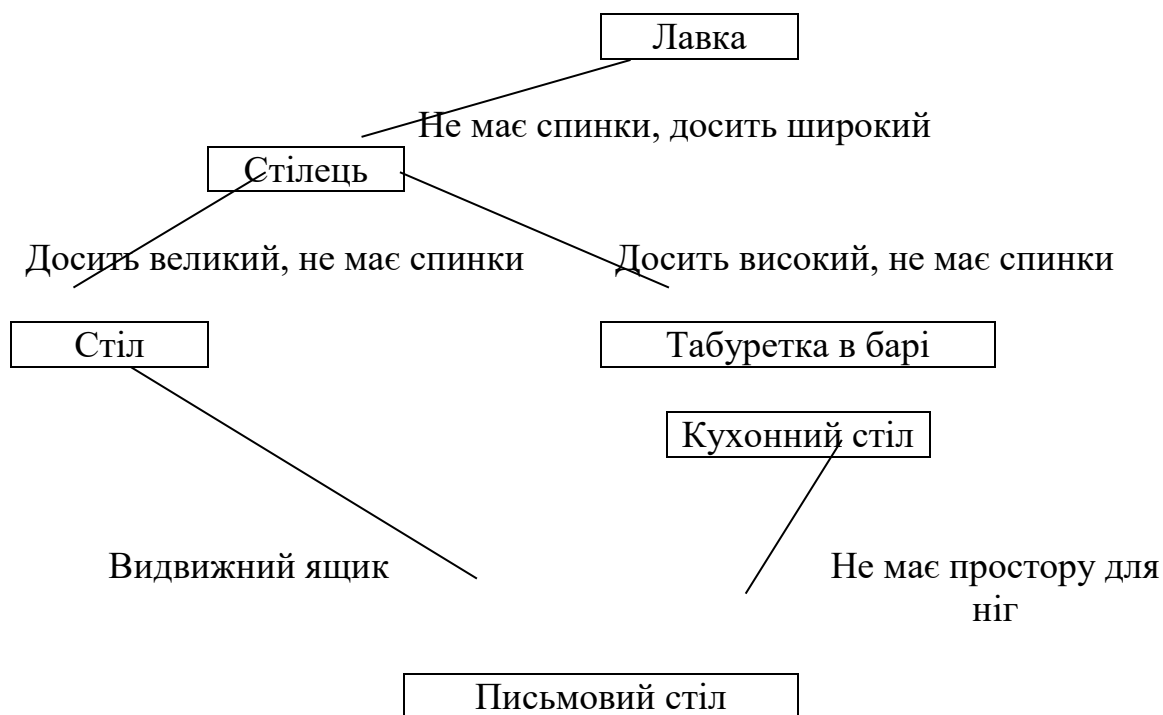


Мал.7.6. Фрейм аудиторії.

На малюнку фактично зображена мережа частково конкретизованих фреймів, пов'язаних між собою зв'язком **HAS_PART**. Взагалі, фрейми можуть об'єднуватися в мережі, і тому їх часто розглядають в одному контексті з семантичними мережами [1].

У [10] показано, як **один і той самий предмет може описуватися різними фреймами, які відповідають різним умовам спостерігання цього предмета**. Риси, спільні для цих різних фреймів, описуються **базовим фреймом**. Можна вважати, що фрейми, які відповідають різним умовам спостереження, успадковують основні характеристики базового фрейму. Зрозуміло, що зберігати ці похідні фрейми в пам'яті не обов'язково.

При пошуку фреймів, які найбільш підходять, велике значення мають мережі подібностей, відмінностей та ін. Як приклад можна навести відому **мережу подібностей за Уїнстоном [10]**:



Мал. 7.7. Мережа подібності за Уїнстоном

Існують різні моделі представлення знань за допомогою фреймів. Фрейм може мати, наприклад, таку структуру:

-

Ім'я Фрейму	Вказівник наслідуванн я	Вказівник ат-рибутів слоту	Значення слоту	Демони
Слот 1				
Слот 2				
Слот n				

Мал.7.8. Структура даних фрейму.

Далі будуть пояснені основні елементи цієї структури.

1) Ім'я фрейму. Цей ідентифікатор є єдиним в даній фреймовій системі.

Кожний фрейм складається з довільного числа слотів (декілька з них системні, решта - користувача). Традиційно перелік основних системних слотів такий:

- a) слот IS - A, що вказує на фрейм-предок даного фрейму;
- b) слот - вказівник фреймів нащадків, який є списком вказівників цих фреймів;
- c) слот для введення імені користувача, дати визначення, дати зміни, тексту коментаря;
- d) інші слоти.

Кожний слот в свою чергу задається фіксованою структурою даних.

2) Ім'я слоту - ідентифікатор, унікальний в фреймі. Деякі імена слотів мають смислове навантаження:

- a) IS - A(визначає тип відношення);
- b) DDESENDANTS (вказівник прямого дочірнього фрейму);
- c)DEFINED BY (користувач, який визначає фрейм);
- d)DEFINEDON (дата визначення фрейму);
- e)MODIFIEDON (дата модифікації фрейму);
- f)COMMENT - коментар;
- g)HAS_PART, RELATIONS - системні і використовуються при редагуванні бази знань і керуванні виведенням.

3) Вказівники наслідування використовуються тільки в фреймових системах ієрархічного типу, що ґрунтуються на відношеннях "IS - A". Вони вказують, яку інформацію про атрибути слотів фрейму вищого рівня наслідують слоти з таким же ім'ям в фреймі нижчого рівня.

Типові вказівники наслідування:

- a) UNIQUE(U: унікальний) - показує, що кожний фрейм може мати слоти з різними значеннями;
- b) SAME(S: такий же) - всі слоти повинні мати однакові значення;
- c) RANGE(R: встановлення меж) - значення слотів нижчого рівня повинні бути в межах, вказаних значеннями слотів фрейма верхнього рівня;

d) **OVERRIDE(O: ігнорувати)** - при відсутності вказівки значення слотів фрейма верхнього рівня становиться значенням слота фрейма нижчого рівня, але в випадку визначення нового значення, значення слотів нижчого рівня вказуються в якості значень слотів. О виконує одночасно функції вказівників U і S.

4) Визначення типу даних - вказується, що слот має числове значення, або виступає вказівником іншого фрейму.

5) Значення слоту. Поле для введення значення слоту. Значення слоту повинно співпадати з вказаним типом даних цього слоту, і, крім того, повинна виконуватись умова наслідування.

6) Демон. Тут дається визначення **демонів** типу **if-needed, if-added, if-removed**. Демоном називається процедура, яка автоматично запускається при виконанні деякої умови. Демони запускаються при зверненні до відповідного слоту. Наприклад, демон **if-needed** запускається, якщо в момент звернення до слоту його значення не було встановлено, **if-added** - при підстановці значення в слот, **if-removed** - при стиранні значення слоту.

Демон можна вважати різновидом **приєднаної процедури**, які також можна включати до опису фреймів. Приєднана процедура запускається за повідомленням, переданим з іншого фрейму.

Відомі спеціалізовані мови для представлення знань фреймами, такі, як KRL та ін.

Часто фреймові моделі ще називаються **об'єктними**.

Загальна форма об'єктного представлення визначена в [9] як

Об'єкт, (атрибут_j, значення_j)(j = 1, ... m).

Об'єктна модель лягла в основу **об'єктно-орієнтованих мов програмування**, таких, як **Smalltalk, C++, Delphi**.

7.3. Зв'язок між семантичними мережами та фреймами

На сучасному етапі фреймові моделі та семантичні мережі розглядають, як правило, у спільному контексті. Для цього є глибокі причини.

З одного боку, ніщо не заважає розглядати вузли семантичної мережі як фрейми з власною внутрішньою структурою.

З іншого боку, можна вводити різноманітні зв'язки між слотами фреймів. Тоді фрейм набуває рис семантичної мережі.

Глибокий аналіз об'єднаної мережно-фреймової моделі знань може, очевидно, принести чимало користі при аналізі глибинних структур знань та взаємозв'язків між різними інформаційними одиницями.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Дайте визначення семантичної мережі.
2. Намалюйте семантичну мережу, що описує деякий набір фактів з довільної предметної області.
3. Опишіть суть проблем, пов'язаних з наслідуванням.
4. Охарактеризуйте поняття фрейму.
5. Опишіть роль фреймів у розумінні.

6. Охарактеризуйте поняття мережі подібностей за Уїнстоном, наmalюйте довільний конкретний приклад.
7. У чому полягає зв'язок між семантичними мережами і фреймами?

Розділ 8. Логічні моделі та метод резолюцій

- *Підемо шляхом логіки...*

- *Підемо разом...*

(З кінофільму “Іронія долі, або З легким паром!”

8.1. Логічні моделі

Логічна модель задання знань базується на формалізмах логіки предикатів першого порядку. Позитивною характеристикою логічних моделей є однозначність теоретичного обґрунтування і можливість реалізації системи формально точних визначень і заключень; недолік - формальний процедурний стиль мислення. Людська логіка - це інтелектуальна модель з нечіткою структурою. В цьому полягає її відмінність від строгої логіки.

Визначення. Логічною моделлю називається формальна система, що задається четвіркою $\langle T, P, A, B \rangle$. Тут T - це множина базових елементів, P - множина синтаксичних правил; A - множина аксіом, B - множина правил виведення.

8.2. Автоматичне доведення теорем та принцип резолюцій

Теоретичною основою логічного програмування є **методи автоматичного доведення теорем**. Перші спроби знайти загальний алгоритм для вирішення цієї задачі простежуються в роботах Лейбніца, Пеано, Гільберта. Центральним серед сучасних методів автоматичного доведення теорем вважається **правило резолюції**. Це правило було сформульовано Робінсоном у 1965 р. на основі більш ранніх робіт Ербрана і Правиця. Застосування цього правила виявилось більш ефективним, ніж класичне виведення у математичній логіці на основі *modus ponens* та ін.

Історичний нарис та непоганий опис методу резолюцій можна знайти в [8].

Будемо описувати знання, представлені в базі знань, за допомогою **основних атомарних формул і фразової форми логіки предикатів**. Атомарна формула - це предикат, що не містить змінних (містить лише константи).

Обмежимося константами a, b, c і предикатами $Q(x,y), P(x), R(x,y), S(x)$. Тоді під атомарними формулами можна розглядати такі вирази, як $Q(a,b), P(c)$ тощо. Такі атомарні формули відповідають фактам бази знань. Правила за своєю суттю є імплікаціями, але у логічному програмуванні використовується *фразова форма*. Фразова форма - це спосіб запису формул, при якому використовується лише кон'юнкція, диз'юнкція та заперечення. Нагадаємо, що вирази $A \rightarrow B$ і $\sim A \vee B$ еквівалентні між собою. Тут символи $\rightarrow, \sim, \vee$ означають відповідно імплікацію, заперечення та диз'юнкцію.

Далі, правила у базах знань повинні бути записані у вигляді **фраз Хорна**. Фраза Хорна - це диз'юнкція атомарних формул, з яких лише одна не стоїть під знаком заперечення. Наприклад, запис $\sim A \vee \sim B \vee \sim C \vee D$ є фразою Хорна. Цей запис

означає, що твердження **D** виконується, якщо одночасно виконуються твердження **A, B, C** (згадайте правила де Моргана!).

Тепер можна сформулювати правило резолюції. Дві фрази можуть бути резольвовані одна з іншою, якщо одна з них містить атомарну формулу без заперечень, а інша - заперечення тієї ж атомарної формули з одним і тим самим предикатом та з однаковою кількістю аргументів, і ці аргументи можуть бути уніфіковані (співставлені один з іншим). Дві фрази

$$\begin{aligned} P(a) \vee \sim Q(b,c) \\ Q(b,c) \vee \sim R(b,c) \end{aligned}$$

є резольвовані, і резольвента записується як **$P(a) \vee \sim R(b,c)$** .

Це правило стане більш зрозумілим і очевидним, якщо його переписати у термінах імплікацій. Нагадаємо, що резольвента стає новою фразою теорії (в математичній логіці під теорією розуміється певний набір тверджень разом з правилами виведення).

Різні константи не уніфікуються одна з одною, так що фрази

$$\begin{aligned} P(a) \vee \sim Q(b,c) \\ Q(c,c) \vee \sim R(b,c) \end{aligned}$$

не є резольвованими.

Змінна уніфікується з будь-якою константою. Так, фрази

$$\begin{aligned} P(a) \vee \sim Q(a,b) \\ Q(x,y) \vee \sim R(x,y) \end{aligned}$$

резольвується, при цьому змінна **x** співставляється з константою **a**, змінна **y** - з константою **b**. У резольвенті **$P(a) \vee \sim R(a,b)$** змінні замінені на константи, з якою вони співставилися.

Якщо в процесі виведення утворилися фрази **$P(a)$** та **$\sim P(a)$** , їх резолюцією є **пуста фраза**. Саме виведення пустої фрази і означає суперечливість теорії.

Тепер можна сформулювати алгоритм перевірки справедливості будь-якого твердження, виходячи з існуючої бази знань, на основі принципу резолюції.

При перевірці породжується деяка теорія, що утворюється **шляхом додавання заперечення нового твердження до уже існуючих** (сама база знань при цьому вважається несуперечливою). Послідовно застосовується правило резолюції, і утворені резольвенти додаються до пробної теорії. Якщо при цьому врешті утворюється пуста фраза, то твердження, яке перевіряється, істинне.

Приклади застосування правила резолюції можна знайти в [8]. Розглянемо один з них.

Нехай є множина фраз

$$\begin{aligned} P(a) \vee \sim Q(a,b) \\ Q(x,y) \vee \sim R(x,y) \\ S(b) \\ R(a,b) \end{aligned}$$

Потрібно з'ясувати, чи є фраза **$P(a)$** наслідком цієї множини фраз. На першому кроці додамо її заперечення **$\sim P(a)$** до решти фраз.

Далі, алгоритм повинен бути сфокусовано на наслідках такого додавання. Це досягається за допомогою таких двох правил: у першій резолюції слід використовувати щойно додане заперечення фрази; кожна наступна резолюція повинна використовувати результат попередньої резолюції.

Спробуйте виконати усі необхідні резолюції самостійно.

Розглянемо ще один приклад застосування принципу резолюції і уніфікації.

Нехай маємо відношення **АРТИСТ(x)**, яке позначає твердження x є артистом і аналогічне відношення **СПІВАК(x)**. Це дає змогу вираз “співак є артистом” формалізувати наступною імплікацією:

СПІВАК(x) $\rightarrow$ АРТИСТ(x).

Тепер, після задання виразу “Яремчук - співак”, спробуємо довести за допомогою методу резолюції вираз “Яремчук - артист”.

Спочатку, додамо до попередніх фраз теорії формулу заперечення того твердження, яке ми хочемо довести. Матимемо наступні фрази теорії:

СПІВАК(x) $\rightarrow$ АРТИСТ(x) (1)

СПІВАК(Яремчук) (2)

$\neg$ АРТИСТ(Яремчук) (3).

Формулу (1) перепишемо у вигляді $\neg$ СПІВАК(x) $\vee$ АРТИСТ(x). Потім зробимо уніфікаційну підстановку x : **Яремчук** відносно формул (1) і (2). В результаті можемо видалити тотожно істинну формулу $\neg$ СПІВАК(Яремчук) $\vee$ СПІВАК(Яремчук). Залишок з’єднаємо зв’язкою $\vee$ і знову ж видалимо. В результаті маємо пусту фразу, що означає суперечливість нашої теорії. Звідси методом обґрунтування логічного висловлення виводиться формула **АРТИСТ(Яремчук).**

Метод резолюцій Робінсона має чіткі аналогії з *методом зворотного виведення*, запропонованим С.Масловим у 1964 році.

8.3. Поняття про логічне програмування

В принципі, експертні системи на основі логічних моделей можна програмувати за допомогою будь-якої мови програмування. Існує безліч описів того, як писати експертні системи на Бейсику, Фортрані, Паскалі, Сі і т.п. Але, очевидно, найкраще пристосовані для цього спеціалізовані мови штучного інтелекту, і особливо **Пролог**.

Пролог є мовою **логічного програмування**. Взагалі кажучи, можна виділяти чотири основні парадигми програмування: процедурна (традиційні методи програмування; типові представники мов програмування - Паскаль і Сі в їх авторських варіантах); функціональна (найбільш відомий представник - Лісп); об’єктно-орієнтована (Smalltalk, C++, Delphi); логічна (Пролог).

Останні три парадигми виділяються в [1] як основні для програмування систем штучного інтелекту.

Парадигма логічного програмування означає [1], що для виконання програм використовується механізм доведення теорем, який дозволяє з’ясувати, чи містяться суперечності у деякому наборі логічних формул. Сама програма розглядається як набір логічних формул разом з теоремою, яка повинна бути доведена.

Перша версія інтерпретатора мови Пролог була написана Колмьєром і Русселем у 1973 р. Цікаво, що вона була написана на Фортрані. Ця версія працювала дуже повільно. У 1977 р. Уоррен та Перейра з Едінбургського університету створили більш досконалу версію.

Подальший розвиток Прологу був пов'язаний з процедурною семантикою імплікації, підміченою Ковальськi. Він опублікував роботу, в якій стверджувалося, що імплікація $p, q \rightarrow r$ допускає не тільки чисто логічне трактування, але й процедурну (якщо хочете довести r , спробуйте довести p і q). Таким чином, послідовність хорнівських фраз може інтерпретуватися як деяка недетермінована програма. Сама послідовність обчислень може бути задана такою метапроцедурою, як *перебір з поверненням*.

Пролог був обраний основним засобом для реалізації **інтелектуальних комп'ютерів п'ятого покоління**. Цей проект був анонсований в Японії на початку 80-х років.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Дайте визначення логічної моделі; наведіть приклад.
2. Представте базу знань з довільної предметної області у вигляді набору атомарних формул і фраз Хорна та показати застосування методу резолюцій для виведення нових знань.
3. Опишіть, як можна застосовувати бектрекінгові алгоритми до задачі логічного виведення.
4. Написати програму на Паскалі, яка ілюструє логічне виведення та може робити висновки про істинність або хибність запитів до програми.
5. Поясніть суть логічного програмування.
6. Наведіть приклади застосування логічного програмування для вирішення конкретних практичних задач.

Розділ 9. Продукційні моделі

- *Що буде, якщо щось буде?*

- *Буде черга, а потім знову нічого не буде.*

(Анекдот часів розвиненого соціалізму)

9.1. Загальна характеристика продукційних моделей

У *продукційних моделях* знання задаються за допомогою правил "якщо-то" (явище-реакція). Вони є дуже широко використовуваними, оскільки для них характерні простота модифікації баз знань.

З одного боку, їх правила виведення близькі до логічних моделей, а з іншого - вони відображають знання у порівнянні з класичними логічними моделями, більш наочно [1].

Визначення. *Продукційною моделлю називається представлення знань у вигляді сукупності продукцій. Продукція визначається в [1] як вираз такого вигляду:*

$$(i) Q; P; A \Rightarrow B; N$$

Тут **(і)** - ім'я продукції, за допомогою якого вона виділяється серед усієї множини продукцій;

Q - сфера застосування продукції; предметна область, до якої вона відноситься;

P - умова застосування продукції;

A $\Rightarrow$ B - ядро продукції; інтерпретація залежить від конкретної ситуації; часто ядро інтерпретується як "якщо маєш **A**, зроби **B**";

N - постумова продукції, постулює процедури, які необхідно виконати після реалізації ядра.

Як приклад можна навести таку продукцію:

МАГАЗИН; У ПОКУПЦЯ ГРОШІ; ЯКЩО ПОКУПЕЦЬ ХОЧЕ КУПИТИ РІЧ, ВІН ПЛАТИТЬ В КАСУ; ЗМІНИТИ БАЗУ ДАНИХ ПРО ТОВАРИ.

Інформаційні системи, побудовані на основі продукційних моделей, називаються **продукційними системами**. У продукційних системах можна виділити **базу фактичних знань, базу (набір) продукцій та систему керування продукціями**. Остання визначає, в якій послідовності слід застосовувати продукції, щоб отримати потрібний результат. Особливо це стосується випадків, коли у деякій ситуації можна застосувати не одну, а декілька продукцій (вирішення конфліктів) [1, 10]. Для керування застосовуються два загальні принципи: **обмеження конфліктного набору та застосування алгоритмів вирішення конфлікту**.

Серед відомих стратегій керування вирішення конфліктів в [1] виділені: принцип "стосу книг"; принцип найбільш довгої умови; принцип метапродукцій; принцип "класної дошки"; принцип пріоритетного вибору; принцип керування за іменами.

Продукційна система складається з трьох основних компонент. Перша компонента - це набір правил, які використовуються як база знань, тому її ще називають базою правил. Наступна компонента є робоча пам'ять, в якій зберігаються умови, які відносяться до конкретних задач предметної області, і результати виведень, отриманих на їх основі. Третя компонента є механізмом логічного виведення. Він використовує правила в відповідності до робочої пам'яті.

Існують дві основні стратегії логічного виведення у продукційних системах. Стратегія, при якій для отримання заключення проводиться робота по вибору попередньо записаного вмісту робочої пам'яті, застосуванню правил і доповнень даних в пам'яті, називається **прямим виведенням**. Спосіб, в якому на основі фактів, які потребують підтвердження, щоб виступати в ролі заключення, досліджується можливість застосування правила, яке дає можливість підтвердження, називається **зворотнім виведенням**.

У випадку застосування зворотнього виведення умова зупинки системи наступна: або досягається першочергова ціль, або закінчились правила, які застосовуються для досягнення цілі в ході виведення.

Очевидно, що вибір чергового правила впливає на ефективність виведення. Якщо на кожному етапі логічного виведення існує декілька правил застосування, тоді ця множина правил носить назву **конфліктного набору**, а вибір одного з них називають вирішенням конфлікту.

Для багатьох практичних застосувань продукційних систем недостатньо запису в робочу пам'ять тільки одного зразка і виникає потреба керувати даними, які уточнюють зміст. В таких випадках використовується спосіб задання конкретних даних з допомогою триплету: **об'єкт - атрибут - значення**. Тому кожна окрема субстанція із предметної області розглядається як один об'єкт. Можна вважати, що дані, які зберігаються в робочій пам'яті, показують значення, що приймають атрибути цього об'єкту. Наприклад, дані (*студент1 ім'я Іван*) описують той факт, що існує деякий студент і ім'я цього студенту Іван.

Однією із переваг використання структури об'єкт - атрибут - значення є уточнення контексту, в якому застосовується правило. Так правило, єдине для всіх об'єктів "студент", повинно бути застосованим незалежно від того, чи мова йде про студента Івана, чи що студента Петра. Але коли існує декілька об'єктів, для яких можна використати деяке правило, тоді воно застосовується для кожного об'єкту не більше одного разу. Ця проблема називається *проблемою контексту застосування правила*, а об'єкт застосування правила називають контекстом застосування правила.

На практиці умовна і заключна частина правил ускладнюється. Але основу складає класичне правило, яке описує відношення виведення, встановлене між вмістом робочої пам'яті, звернення до якого надходить із умовної частини, і змістом, описаним в заключній частині.

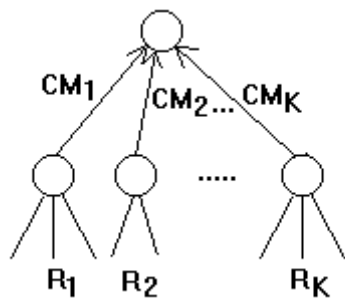
9.2. Випадок нечітких продукційних правил

В реальних системах продукцій іноді буває необхідним працювати з *нечіткою інформацією*. В таких випадках виділяють дві категорії нечіткості:

- а) нечіткість безпосередніх даних;
- б) нечіткість виведення.

Коли заключення виводиться за допомогою декількох правил, які включають в себе і нечіткі, виникає проблема визначення степені нечіткості всього заключення.

Розглянемо випадок, коли одне заключення C виводиться за допомогою множини правил $R_1 - R_n$ (мал.5.11).



Мал.9.1. Синтез достовірності при роботі з нечіткими даними.

Припустимо, що нечіткість заключення C характеризується коефіцієнтом впевненості, заданим числовим значенням. Кожний коефіцієнт впевненості заключення C для кожного правила позначимо відповідно $CM_1 - CM_n$. Розглянемо в якості методу визначення коефіцієнта впевненості CM_t , функцію синтезу :

$$CM_t = \text{КОМБ}(CM_1, \dots, CM_n) .$$

Спосіб пошуку залежить від вигляду КОМБ. В загальному випадку вважається, що якщо хоча б одна з CM_t не містить нечіткості, тоді і CM_t не містить нечіткості,

тому якщо таке правило існує, тоді необхідно спробувати використати його раніше інших. Фактично, якщо коефіцієнт впевненості не містить нечіткості, пошук можна закінчувати, в іншому випадку аналізується КОМБ.

Як уже ми зазначали, існує пряме і зворотнє виведення.

В прямому виведенні повторюється цикл пошуку даних в робочій пам'яті, генерація конфліктного набору, вирішення конфлікту і застосування правил. Використовується 2 основних методи:

- 1) генерація ряду обмежень на породження конфліктного набору;
- 2) визначення більш строгого алгоритму вирішення конфлікту.

Реалізація першого методу передбачає два підходи :

- 1) правила розбиваються на окремі категорії, і в визначених ситуаціях аналізується виконання правил деякої категорії;
- 2) пошук умовної частини правил визначеної категорії не проводиться.

В останньому підході реалізується використання метаправил - правил про правила, правил, в умовній частині яких знаходяться умови, що відносяться до змісту правил і змісту пам'яті, або ж правил, в умовній частині яких вказується на атрибути, які визначають непотрібність пошуку.

Згідно першого підходу правила попереднього групуються по атрибутах, для кожної групи вказується умова (форма якої аналогічна умовній частині правила) і, аналізуючи стан робочої пам'яті проводиться пошук застосування правил тільки з цієї групи.

Якщо при нашому виведенні всі заключення, які можна визначити, знаходяться, тоді конфлікт вирішується просто:

- а) правила застосовуються в порядку їх визначення;
- б) у випадку, коли задана умова зупинки, її можна прискорити - застосувавши виведення по пріоритету глибини, що відповідає спробі пріоритетного застосування правила, що пов'язане з останньою інформацією, якою була доповнена робоча область. При даному способі шлях логічного виведення мінімальний. Можна використати й інші пріоритети, наприклад, правила з більшим числом підумов в умовній частині

Все сказане вище відноситься і для зворотнього виведення. Крім того, додається ще одна задача, пов'язана з послідовністю оцінки умов в умовній частині правил.

Прикладом застосування алгоритмів виведення може послужити алгоритм **RETE**. Згідно алгоритму погодження **RETE**, кожного разу при поповненні робочої пам'яті новим зразком перевіряється правило, в якому він використовується. Якщо зразок задовільняє частині умов правила, тоді він запам'ятовується тільки в цій якості, коли ж задовільняє всім умовам, дане правило включається в конфліктний набір.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

Дайте визначення продукційної моделі та продукційної системи.

Напишіть набір продукцій з довільної предметної області. Покажіть, як за допомогою цього набору продукцій можна вирішувати конкретні практичні задачі.

Поясніть різницю між прямим та зворотним виведенням у продукційних системах.

Розділ 10. Формалізація нечітких, неточних та недостовірних знань

- Лікарю, а що Ви мені написали в діагнозі: ЧЙЗ ?

- Чорт Його Знає...

(З анекдоту)

10.1. Нечіткі множини

Важливе місце в теорії штучного інтелекту займає **теорія нечітких множин та методи нечіткого логічного виведення** [1, 10, 11, 12].

Доцільність введення нечітких множин зумовлена нечіткістю багатьох формулювань, притаманних природній мові. Як можна, наприклад, формалізувати поняття “маленьке число” або “цікава книга”? Чи можна, наприклад, визначити “множину маленьких чисел” з урахуванням того, що деяке число, наприклад, 5, може вважатися “маленьким” у більшій або меншій мірі? Для формалізації таких понять було запропоновано застосовувати апарат **нечітких множин**, точніше, нечітких підмножин. Інша назва нечітких множин - **розмиті множини**.

Відомо, що зі звичайними множинами тісно пов'язане поняття **характеристичної функції**. Характеристичною функцією $\chi_A(x)$ називається функція, значення якої вказують, чи належить елемент x множині A : $\chi_A(x) = 1$, якщо $x \in A$ і $\chi_A(x) = 0$, якщо $x \notin A$.

Професор Л.Заде у 1965 р. запропонував розширити поняття характеристичної функції. Він ввів поняття функції нечіткої належності $\mu_A(x)$ елемента до множини, дозволивши цій функції приймати будь-яке значення від 0 до 1. Чим більшим є значення функції належності, тим у більшій мірі елемент належить до множини. Якщо $\mu_A(x) = 1$, елемент x чітко належить множині A , якщо ж $\mu_A(x) = 0$, то елемент x чітко не належить цій множині. При $0 < \mu_A(x) < 1$ елемент належить множині нечітко. Чим більшим є значення функції належності, тим у більшій мірі елемент належить до множини.

Дамо формальне визначення.

Визначення. Нехай E - довільна непуста множина. **Нечіткою підмножиною** A множини E називається множина пар

$$A = \{(x, \mu_A(x))\}, \text{ де } x \in E, \mu_A(x) \in [0,1].$$

Функція $\mu_A : E \rightarrow [0,1]$ називається **функцією належності** нечіткої підмножини A , а E - **базовою множиною**, або **базовою шкалою**. Значення $\mu_A(x)$, що визначене для кожного $x \in E$, називається **ступенем належності** елемента x нечіткій множині E . Як правило, до A не включаються елементи з нульовим ступенем належності, хоч це і не є обов'язковим. **Носієм** нечіткої підмножини A називається підмножина A^* множини E , що містить ті елементи з E , для яких $\mu_A(x) > 0$.

Носієм нечіткої підмножини завжди є чітка, звичайна підмножина базової множини. Тому правильніше говорити “нечітка підмножина”, а не “нечітка множина”. Втім, ми будемо часто вживати термін “нечітка множина”, якщо це не викликатиме непорозуміння.

Приклад. Формалізуємо поняття “маленьке число” на основі нечіткої множини маленьких чисел.

Нехай базовою множиною E є множина цілих додатніх чисел. Тоді її нечітка підмножина A маленьких чисел може бути, наприклад, такою:

$$A = \{(0, 1), (1, 1), (2, 1), (3, 0.95), (4, 0.9), (5, 0.8), (6, 0.5), (7, 0.4), (8, 0.25), (9, 0.1), (10, 0.05)\}.$$

Носієм цієї нечіткої множини буде множина $A^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Необхідно звернути увагу на такі важливі моменти:

- **функція належності не може бути визначена однозначно, оскільки вона є дуже суб'єктивною;** кожна людина може запропонувати свої значення цієї функції. В іншій ситуації значення ступенів належності могли б бути зовсім іншими;
- **самі ступені належності є розмитими.** Уявимо собі, що дехто оцінив число 4 як маленьке зі ступенем належності 0.9. Якщо у нього спитати, чому 0.9, а не 0.85 чи 0.92, він не зможе чітко відповісти на це запитання. Тому замість конкретних числових значень ступенів належності можна говорити про **інтервали оцінок ступенів належності**. Якщо ступінь належності кожного елемента оцінюється інтервалом, говорять про **нечіткі множини другого роду** $[]$ з функцією належності $\mu_A : E \rightarrow [0, 1]$ $[0, 1]$. Такі міркування можна продовжити і далі. Втім, на практиці найчастіше користуються описаними раніше нечіткими множинами першого роду (з конкретними числовими значеннями ступенів належності).
- ступені належності можуть відображати також **міру невпевненості**.

Більш загально, нечіткі якісні знання можуть бути описані на основі **лінгвістичних змінних**, значення яких описуються словесними виразами. Такою змінною може бути, наприклад, "характеристика віку" (змінна може набувати значень "молодий", "старий", "дуже старий" та ін.). Лінгвістична змінна визначається в [1] як п'ятірка

$$\langle A, T(A), U, G, M \rangle,$$

де A - ім'я лінгвістичної змінної;

$T(A)$ - терм - множина (множина можливих значень);

U - область визначення лінгвістичної змінної, носій лінгвістичних значень;

G - набір операцій по породженню похідних лінгвістичних значень;

M - набір семантичних правил, за якими відбувається відображення значень лінгвістичних змінних в нечіткі множини та ін. (або, іншими словами, M визначає функції належності).

10.2. Основні операції над нечіткими множинами

Над нечіткими множинами може бути визначено досить юагато різноманітних операцій. Розглянемо деякі з них.

Рівність. Дві нечіткі підмножини базової множини E називаються рівними між собою, якщо для будь-якого $x \in E$ $\mu_A(x) = \mu_B(x)$. Будемо позначати це як $A = B$.

Включення. Будемо казати, що **A** включається до **B** (позначається $A \subseteq B$), якщо для будь-якого $x \in E$ $\mu_A(x) \leq \mu_B(x)$.

Перетин. Існує досить багато операцій перетину. Найбільш вживаною з них є така. Перетином двох нечітких множин **A** і **B** називається нечітка множина $C = A \cap B$ з функцією належності

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)).$$

Об'єднання. Найбільш вживана операція об'єднання визначається так. Об'єднанням двох нечітких множин **A** і **B** називається нечітка множина $C = A \cup B$ з функцією належності

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)).$$

Доповнення. Нечітку множину **B** будемо називати доповненням нечіткої множини **A**, якщо

$$\mu_B(x) = 1 - \mu_A(x).$$

Підмножина α -рівня. Нехай $\alpha \in [0,1]$. Підмножиною α -рівня нечіткої підмножини **A** називається звичайна підмножина

$$A_\alpha = \{x: \mu_A(x) \geq \alpha\}.$$

10.3. Нечітке логічне виведення. Метод центру тяжіння композиції максимум-мінімум

Методи нечіткого логічного виведення займають дуже важливе місце в теорії та практиці штучного інтелекту, оскільки вони дозволяють формалізувати міркування, в яких присутні недостатньо визначені поняття.

Розглянемо приклад [12]. Нехай є знання експерта про те, що, якщо рівень води високий, то необхідно відкрити клапан. Це правило може бути записано у вигляді такої продукції продукційної системи:

Якщо РІВЕНЬ ВИСОКИЙ, то ВІДКРИТИ КЛАПАН.

Що повинна порекомендувати експертна система, якщо рівень води **ДОВОЛІ ВИСОКИЙ**? Людина зразу робить висновок, що клапан потрібно **ЗЛЕГКА ВІДКРИТИ**. Але як до такого висновку може прийти інтелектуальна система?

Для формалізації вирішення цієї задачі необхідно зробити такі основні кроки [12]:

- визначити експертним нечіткі множини, які відповідають поняттям **РІВЕНЬ ВИСОКИЙ** та **ВІДКРИТИ КЛАПАН**;
- визначити експертним шляхом нечітку множину, яка відповідає поняттю **РІВЕНЬ ДОВОЛІ ВИСОКИЙ**;
- визначити порогове значення **a**, яке дорівнює максимальному значенню перетину множин **ВИСОКИЙ** і **ДОВОЛІ ВИСОКИЙ**;
- визначити нечітку множину, яка відповідає поняттю **ЗЛЕГКА ВІДКРИТИ КЛАПАН**; функція належності цієї множини визначається як зріз функції, що відповідає поняттю **ВІДКРИТИ КЛАПАН** на рівні **a**;
- провести *дефаздифікацію*, тобто за функцією належності, тобто за аналітичним виглядом функції **ЗЛЕГКА ВІДКРИТИ КЛАПАН** обчислити

конкретне значення кута, на який слід відкрити клапан; це значення визначається як центр тяжіння, або середнє значення функції.

10.4. Багатозначні і модальні логіки

За допомогою **модальних логік** [9] можна виразити можливість або необхідність деякого факту, ставлення когось до того чи іншого факту та ін. Це, наприклад, такі фрази, як: **"На вулиці може йти сніг"**, або **"Вася знає, що квадрат гіпотенузи дорівнює сумі квадратів катетів"**, або **"Маша думає, що Волга впадає у Чорне море"**, або **"Джордж вважає, що Фред не буде заперечувати, що він убив Майкла"** тощо.

Однією з найбільш відомих модальних логік є **тризначна логіка Лукасевича**. У цій логіці будь-яке висловлювання може приймати одне з трьох можливих значень істинності:

0 - неможливе висловлювання (клас I);

1 - можливе, але не необхідне висловлювання (клас U);

2 - необхідне висловлювання (клас N).

Має місце закон **виключеного четвертого**: **будь яке висловлювання належить або класу N, або класу U, або класу I.**

Таблиці істинності для заперечення, кон'юнкції, диз'юнкції та імплікації в логіці Лукасевича визначаються таким чином:

$\sim F$

F	$\sim F$
0	2
1	1
2	0

$F \wedge G$

F\G	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

$F \vee G$

F\G	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

$F \Rightarrow G$

F\G	0	1	2
0	2	2	2
1	1	2	2
2	0	1	2

Можна згадати також **семантику можливих світів**, для якої характерно чотири значення істинності висловлювань: необхідно істинні, нейтрально істинні, нейтрально хибні та необхідно хибні.

Основне призначення **немонотонних логік** [9, 10] - коректна інтерпретація правил з виключеннями.

10.5. Базові поняття теорії індуктивних висновків

Слово "*індукція*" означає здійснення висновку "від часткового до загального". Індуктивний висновок полягає у тому, що інтелектуальна система спостерігає певний набір предметів, для яких справджується певне правило, і на основі цього спостереження приходить до думки, що це правило справджується для всіх предметів. Слід розрізняти *індуктивну перевірку гіпотез* та *індуктивне формування гіпотез* на основі прикладів. У першому випадку перевіряється деяка уже сформована гіпотеза. Якщо при цьому не вдається здійснити повний перебір, ми ніколи не зможемо бути повністю впевнені у справедливості гіпотези; ми можемо лише збільшити або зменшити ступінь нашої впевненості.

У другому ж випадку на початку дослідження гіпотези ще не існує. Вона формується на основі аналізу прикладів, і після цього її часто можна перевірити іншим шляхом.

10.6. Правила Мілля формування гіпотез

Мілля запропонував чотири правила для індуктивних висновків. Ще раз підкреслимо, що індуктивні висновки можуть бути помилковими. Зверніть увагу на те, що у різних джерелах ці правила можуть формулюватися дещо по-різному.

Правило 1 (метод виявлення подібностей). Нехай ми маємо такі спостереження: (a, b, c, d) ; (a, b, d, e) ; (a, g, f, d) . На підставі таких спостережень можна висунути гіпотезу: "***a та d завжди зустрічаються разом***".

Правило 2 (метод виявлення відмінностей). Нехай ми маємо такі спостереження: (a, b, c, d) ; (b, c) ; (c, f) , (a, g, f, d) . На підставі таких спостережень можна висунути ту саму гіпотезу: "***a та d завжди зустрічаються разом***", це правило є посиленням правила 1.

Правило 3 (метод виявлення залишку). Нехай ми маємо такі спостереження: (a, b, c, d, e, f) ; (b, c) ; (e, f) . Індуктивний висновок той самий.

Правило 4 (метод зв'язних змін). Нехай $a_i \in A$, $d_i \in D$, і ми маємо групу спостережень: $(a_i, \dots, d_i)$; крапками позначені несуттєві елементи. Можна зробити висновок про те, що поява елемента з множини A завжди тягне за собою появу елемента з множини D .

На основі правил Мілля був запропонований відомий **ДСМ-метод автоматичного формування гіпотез** [13], названий так на честь Джона Стюарта Мілля. Широко розповсюджений також **GUHA - метод** [1, 17].

Як один з класичних прикладів застосування теорії індуктивних висновків можна навести таку задачу: заданий числовий ряд, наприклад, **1, 4, 9, 16, 25**; знайти наступний член цього ряду. Розвивається формалізована **математична теорія індуктивних висновків** [14]. Як самостійний напрямок досліджень, оформилася теорія синтезу програм на прикладах.

Розвивається також **теорія аналогій [14]**. Під аналогією слід розуміти схожість певних явищ або понять. Під виведенням по аналогії розуміється виведення подібних висновків, якщо є подібні передумови. Як і індуктивне виведення, виведення за аналогією не завжди є правильним. Можна навести такий приклад виведення за аналогією:

*Автобус - громадський транспорт.
Трамвай - громадський транспорт.
Автобус перевозить пасажирів.
Отже, і трамвай перевозить пасажирів.*

У даному конкретному випадку висновок за аналогією виявився правильним, хоча можна навести багато прикладів хибних висновків за аналогією.

Виведення за аналогією може використовуватися інтелектуальною системою, якщо вона зустрічається з незнайомою задачею.

10.7. Індуктивна перевірка гіпотез і парадокс Хемпеля

З індуктивною перевіркою гіпотез пов'язаний відомий *парадокс Хемпеля*, суть якого проілюструємо на класичному прикладі про чорну ворону.

Нехай учений бажає перевірити гіпотезу **"Всі ворони чорні"**. Природний шлях для індуктивної перевірки цієї гіпотези - дослідити якомога більшу кількість ворон і перевірити, чорні вони чи ні. Кожна виявлена не чорна ворона спростовує гіпотезу, але кожна виявлена чорна ворона збільшує впевненість у справедливості цієї гіпотези, тобто є *підкріплюючим прикладом* для неї.

Але ж висловлювання **"Всі ворони чорні"** логічно еквівалентне висловлюванню **"Всі нечорні предмети не є воронами"**. Спостереження, наприклад, зеленої книги збільшує міру впевненості у справедливості останньої гіпотези, але твердження про те, що воно ж є підкріпленням для гіпотези **"Всі ворони чорні"**, викликає сумнів. Якщо це так, то зелена книга з тим же успіхом підкріплює і гіпотезу **"Всі ворони білі"**.

До цього можна додати ще й таке міркування. Нехай деякий інопланетянин перевіряє гіпотезу **"Всі люди на Землі нижчі за 5 метрів"**. Ясно, що виявлення людини зі зростом 4 метри 99 сантиметрів, хоч і підтверджує гіпотезу, не підкріплює її, а, навпаки, **послаблює**.

Більш детальне обговорення парадоксу Хемпеля та пов'язаних з ним проблем можна знайти в [Д.8]. Існує багато формалізацій індуктивного виведення. Очевидно, історично першою була класична схема Мілля, її опис можна знайти в [2, 13].

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. У чому полягає обмеженість класичної логіки при побудові реальних інтелектуальних систем?
2. У чому полягає значимість теорії нечітких множин?
3. Дайте визначення поняття нечіткої множини.
4. Перелічіть відомі Вам операції з нечіткими множинами.

5. Опишіть суть нечіткого логічного виведення за методом “центру тяжіння композиції максимум-мінімум”; проілюструйте цей метод на довільному прикладі.
6. Опишіть основне призначення модальних логік.
7. Опишіть тризначну логіку Лукасевича; наведіть таблиці істинності для основних операцій цієї логіки.
8. У чому полягає суть парадоксу Хемпеля?

ЧАСТИНА IV. ЗАГАЛЬНІ ПІДХОДИ ДО ПОБУДОВИ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 11. Загальноінтелектуальні метапроцедури

Видатним досягненням сучасної науки стала програма, яка на будь-яке запитання дає будь-яку відповідь

(З першоквітневого випуску “Наркому”)

Ми бачили, що для вирішення різних інтелектуальних задач можна застосовувати різні алгоритми та процедури. Але, очевидно, говорити про те, що система стає більш інтелектуальною тільки від того, що для вирішення нової задачі запрограмований новий алгоритм, не доводиться. Інтелектуальною слід вважати систему, яка вміє орієнтуватися і планувати свою поведінку в новій ситуації. Для цього вона повинна мати в своєму розпорядженні певні загальноінтелектуальні метапроцедури.

Спочатку розвивалася **стимульно-реактивна теорія**, яка розглядає навчання як поступове вироблення правильної реакції на зовнішні подразники. Ця теорія згодом була поповнена **лабіринтною теорією**. Згідно з цією теорією, **задача планування цілеспрямованих дій вирішується на основі метапроцедури пошуку в лабіринті можливостей**.

Але лабіринт можливостей може бути не єдиним. Була висунена **модельна гіпотеза**, згідно з якою **центральною метапроцедурою інтелектуальної діяльності є знаходження нового лабіринту, в якому можна організувати ефективний пошук рішення**.

Третьою метапроцедурою є **структуризація опису ситуації, тобто виділення базових елементів та характерних особливостей, на основі яких можна приймати рішення**.

Виділяють також такі метапроцедури, як **зведення задач до підзадач, формування закономірностей на основі спостережень та ін.**

Характерною особливістю інтелектуального прийняття рішень є **практична неможливість проаналізувати всі елементи ситуації та всі варіанти рішень за прийнятний час**. Тому **необхідною рисою інтелектуальної системи повинні стати метапроцедури евристичного скорочення пошуку та припинення такого пошуку, якщо його продовження видається недоцільним**. Зрозуміло, що рішення при цьому не обов’язково буде оптимальним.

Не претендуючи ні на новизну, ні на універсальність, можна сформулювати такі критерії припинення аналізу ситуації та прийняття деякого рішення.

Принцип толерантності. Нехай r - оцінка ситуації, в яку потрапить система за умови, якщо рішення приймається на основі повного аналізу поточної ситуації, s - оцінка ситуації, в яку потрапить система за умови, якщо рішення приймається негайно. Рішення можна приймати, якщо r і s зв'язані відношенням толерантності, тобто якщо ситуації r і s оцінюються як досить схожі.

Принцип мінімізації затрат. Пошук припиняється і рішення приймається негайно, якщо оцінка затрат, пов'язаних з продовженням пошуку, перевищує оцінку втрат від неоптимальності рішення.

Подальше викладення відноситься до добре відомої **теорії ситуаційного керування**, яка інтенсивно розвивалася у 70-ті - 80-ті роки. Одне з найбільш відомих викладень основ цієї теорії дано в [13].

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Опишіть відомі Вам загальноінтелектуальні метапроцедури.
2. Сформулюйте принцип толерантності.
3. Сформулюйте принцип мінімізації затрат.

Розділ 12. Ситуаційне керування

*- Це ще не кінець, - каже внутрішній голос. -
Вріж йому по пиці...*

Врізав...

*- А ось тепер кінець..., - каже внутрішній
голос.*

(З анекдоту)

12.1. Базові концепції

Ситуаційне керування застосовується для вирішення задачі керування дуже складними об'єктами (живі істоти, суспільні групи, фірми та ін). Для цих об'єктів характерні такі риси: **практично неможливо отримати точну математичну модель об'єкту і формалізувати мету його функціонування**, а відтак - отримати математичні критерії оптимального керування; **об'єкти керування часто є унікальними**, і алгоритми керування, придатні для одного об'єкту, не підходять для іншого; **об'єкти керування постійно змінюються у часі та просторі**; **наявна інформація завжди, з одного боку, неповна, а з іншого - надлишкова**; **принципово квазіалгоритмічний характер керування**; керуючі впливи не обов'язково призводять до запланованих результатів; пов'язано це як з неповнотою опису, так і з тим, що об'єкти керування можуть мати свої цілі, які не обов'язково співпадають з цілями системи керування (так звана "свобода волі").

Усі ці фактори призводять до практичної неможливості отримати оптимальні результати. Керування має стати субоптимальним, або просто раціональним.

В [13] описана одна з можливих схем ситуаційного керування. У найбільш загальних рисах, система керування постійно виконує такі операції: **оцінка поточної ситуації та віднесення її до одного з класів; поповнення опису ситуації на основі знань, що містяться в пам'яті системи; вироблення рішення про те, які дії слід виконати для зміни ситуації в бажаному напрямку; реалізація прийнятого рішення.**

Однією з характерних особливостей ситуаційного керування є те, що **кількість можливих ситуацій набагато перевищує кількість можливих рішень.**

Ключовими для ситуаційного керування є **семіотичні моделі та мови опису ситуацій.**

12.2. Семіотичні моделі та мови опису ситуацій

В [13] семіотичні (знакові) моделі визначаються на основі логічних моделей. Якщо формальна логічна модель задається четвіркою $M = \langle T, P, A, B \rangle$, де **T** - множина базових елементів; **P** - синтаксичні правила; **A**-система аксіом формальної моделі; **B**-множина правил виведення, то семіотична модель задається набором $S = \langle M, h_T, h_P, h_A, h_B \rangle$, де **M** - формальна логічна модель; **h_T, h_P, h_A, h_B** - правила зміни відповідно базових елементів, синтаксичних правил, аксіом та правил виведення. За допомогою семіотичних моделей можна виразити як змінність самих ситуацій, так і неоднозначність їх інтерпретації.

В [13] сформульовані три гіпотези, які покладені в основу мов для опису ситуацій.

Гіпотеза 1. *Вся інформація про об'єкт керування та засоби керування може бути виражена засобами природньої мови.*

Гіпотеза 2. *Будь-який текст природньою мовою можна перекласти на формальну мову семіотичної моделі.*

Гіпотеза 3. *При відображенні зовнішнього світу людина виділяє у ньому скінченний набір відношень.*

У природній мові виділені такі основні групи слів: *поняття*; відповідні лексичні одиниці описують однорідні групи фактів, подій, явищ та ін.; *імена* використовуються для конкретизації елементів, які відносяться до поняття (класу); *відношення*, які фіксують зв'язки між поняттями (класифікації, ознак, що є композиціями двох базових "мати ознаку" та "мати певне значення ознаки", кількісні, які є композиціями двох базових "мати одиницю вимірювання" та "мати певне значення одиниці вимірювання", порівняння, належності, часові, просторові, каузальні відношення (причина - наслідок), інструментальні (наприклад, A є засобом для здійснення B), порядку, дії, *квантифікатори* (часто, багато та ін.), *модальності* (необхідно, бажано та ін.), *оцінки* (добре, шкідливе та ін.).

Зрозуміло, що набори понять та дій повністю визначаються семантикою предметної області, але набори понять, квантифікаторів, оцінок і т.п. досить універсальні. В [13] їх виділено біля сотні. На їх основі формуються фрази **мови ситуаційного керування**, за допомогою яких описуються реальні ситуації.

Основною одиницею мови ситуаційного керування є **проста ядерна конструкція**, яка задається трійкою **(xyz)**. Тут **x,z**- поняття, а **y** - відношення або дія. Замість **x** або **y** можуть бути підставлені інші прості ядерні конструкції. В утворенні **складних ядерних конструкцій** беруть участь також квантифікатори та модальні оператори. Вони мають вигляд: **K ((a₁x) (a₂x) (a₃x))**, де **K** - квантифікатор, **a₁** - квантифікатор або модифікатор (до модифікаторів відносять спеціально виділені слова, такі, як "швидко", "повільно", "обережно"); **a₂** може бути також модальністю.

Типовий перелік відношень та приклади їх застосування можна знайти в [13, 2].

Мови опису ситуацій можуть бути легко зведені до семантичних мереж та фреймів.

Поповнення опису ситуації в теорії ситуаційного керування відбувається на основі так званих **псевдофізичних логік** (часова, просторова, каузальна та ін) та **сценаріїв**. [13]. Псевдофізичні логіки відображають базові властивості часу, простору, відношень "причина-наслідок", спільні для всіх предметних областей.

Сценарієм називається послідовність подій, що є типовим для тієї чи іншої ситуації. Легко побачити їх тісний зв'язок з фреймами.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Охарактеризуйте поняття ситуаційного керування.
2. Опишіть основні етапи прийняття рішень в рамках теорії ситуаційного керування.
3. Дайте визначення семіотичних моделей знань.
4. Охарактеризуйте поняття "мови опису ситуацій".

13. Моделі і методи навчання і самонавчання

- *Ви вважаєте, що за двадцять років можна навчити віслюка читати?*
- *За двадцять років хтось обов'язково помре: або я, або замовник, або віслюк. У будь-якому випадку я не несу ніякої відповідальності.*
(Анекдот про Ходжу Насреддіна)

13.1. Базові визначення

Поняття "навчання" та "самонавчання" є одними з ключових у теорії штучного інтелекту. Вони явно або неявно зустрічаються в усіх розділах курсу.

У психології під навчанням розуміється набуття умінь та навичок, які раніше були невідомими. Для інтелектуальних систем поняття навчання і самонавчання аналогічні. Кажуть, що система навчилася чомусь, якщо вона стала здатною до вирішення якихось задач, які до цього вона вирішувати не могла. Звичайно, якщо нова процедура просто кимось програмується, навряд чи є сенс говорити про справжнє навчання (згадайте поняття декларативного і процедурного підходів, а також дискусію про визначення інтелекту взагалі). Тому часто спеціально підкреслюється, що **при навчанні інтелектуальної системи відбувається самостійне виведення нової інформації з уже існуючої або щойно отриманої** [1].

Уїнстон [4] виділяє такі **рівні навчання**: навчання шляхом програмування; навчання шляхом пояснення; навчання на прикладах; навчання шляхом відкриття.

Можна сказати, що **при переході від більш низьких рівнів до більш високих "учень" виконує все більшу частину тієї роботи, яку раніше виконував "учитель"**.

Навчання без "учителя" називається **самонавчанням**.

Одне з можливих математичних формулювань задачі навчання як оптимізаційної задачі можна знайти в [1].

13.2. Автомати з лінійною тактикою

Історично першими були моделі навчання на основі **стимульно-реактивної теорії**. По суті, це було моделювання процесу утворення умовних рефлексів. Система, що сприймає деякий подразник, відповідає на нього певною реакцією. Якщо ця реакція була правильною, система отримує "заохочення", яке підкріплює її впевненість у правильності цієї дії. Якщо ж дія була неправильною, система отримує "покарання".

Для моделювання такого типу навчання виявилось зручним застосовувати **автомати з лінійною тактикою** [2]. У найпростішому випадку кількість станів автомату з лінійною тактикою дорівнює km , де k - кількість можливих дій, m - кількість станів, що відповідає кожній дії. Ці стани вважаються пронумерованими від 1 до m .

Нехай у поточний момент автомат перебуває у j -му стані з тієї групи станів, які відповідають деякій дії, і, відповідно, реалізує цю дію. Якщо він при цьому отримує заохочення, автомат переходить до $j+1$ - го стану цієї ж дії, якщо це можливо. Якщо ж автомат отримує покарання, він переходить до $j-1$ - го стану цієї дії. Якщо при цьому j отримує нульове значення, автомат переходить до першого стану якоїсь іншої дії.

Параметр m називається **глибиною пам'яті** автомата. Грубо кажучи, цей параметр характеризує кількість покарань, що їх повинен отримати автомат, щоб змінити свою поведінку.

Відомо ряд варіантів та модифікацій цієї простої схеми. Можна розглядати випадки, коли глибина пам'яті для кожної дії своя, або автомати, переходя яких залежать від інтенсивності заохочення чи покарання тощо. Велись дослідження з метою визначення оптимальної глибини пам'яті. Досліджувалася також **поведінка колективів автоматів з лінійною тактикою**.

Надалі, крім стимульно-реактивної теорії, стали застосовуватися інші моделі. Вони будуть більш детально розглядатися у наступних розділах.

13.3. Формування та засвоєння понять

Важливе місце в процесі навчання займає **формування та засвоєння понять**. Уїнстон у [4] описує метод засвоєння понять на прикладах шляхом зміни семантичної мережі. При демонстрації першого прикладу система шляхом його безпосереднього аналізу формує в своїй пам'яті модельну семантичну мережу. Далі описується методика корекції цієї семантичної мережі при демонстрації підкріплюючих прикладів, контрприкладів і "*майже того*". "*Майже те*" - це приклад, який не є представником даного класу через деякі невідповідності.

Ключовим для будь-якого наукового дослідження є здатність до **узагальнення**. Під узагальненням можна розуміти формування нових понять на основі виявлення схожості або відмінностей між існуючими, або ж пояснення існуючих фактів. В [1] дається таке визначення узагальнення: **на підставі фактів, які спостерігаються, та наявних апріорних знань, сформулювати правило, яке**

дозволяє пояснювати існуючі факти та прогнозувати нові. Спочатку це правило формулюється як гіпотеза, яка має бути підтверджена або відхилена.

Можна розрізнити методи узагальнення за ознаками та структурно-логічні методи узагальнення. Методи узагальнення за ознаками є типовими для задач розпізнання зразків та кластерного аналізу. По суті, як узагальнення для задачі розпізнавання можна розглядати вироблення правила вирішення. Однією з класичних є схема Бонгарда, яка на основі аналізу певного набору об'єктів дозволяє шляхом перебору ознак виділити ті комбінації ознак, які відрізняють один клас від іншого [Д.6].

Серед інших методів узагальнення за ознаками слід звернути увагу на метод зростаючих пірамідальних мереж [1, 13].

Серед структурно-логічних методів узагальнення виділяють методи узагальнення на семантичних мережах [1, Д.7] та методи, пов'язані з індуктивним виведенням.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Охарактеризуйте поняття навчання і самонавчання.
2. Опишіть методи навчання автоматів з лінійною тактикою.
3. Напишіть програму, яка моделює колективну поведінку автоматів з лінійною тактикою на прикладі довільної задачі.
4. Нехай довільний перелік геометричних фігур заданий своїми описами. Напишіть на Ліспі програму, що вирішує таку задачу: знайти фігуру **X** з цього переліку, таку, що задана фігура **C** відноситься до **X** так само, як задана фігура **A** до заданої фігури **B**.
5. Проілюструйте процес формування модельної семантичної мережі при засвоєнні понять на прикладах.

ЧАСТИНА V. ДЕЯКІ ЗАСТОСУВАННЯ СУЧАСНИХ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ

Розділ 14. Інтелектуальні інформаційні системи.

*И тогда я поняла, что те надо, что те надо,
Но не дам я тебе, что ты хош
(З сучасної пісні)*

14.1. Проблема розуміння

Проблема розуміння полягає в адекватному сприйнятті інтелектуальною системою зовнішньої ситуації або деякої інформації, яка сприймається системою. Ця інформація може бути довільної природи; обмежимося розглядом проблеми розуміння текстів природною мовою. Подальше викладення ведеться на основі [1].

Нехай на вході системи маємо текст **B** = <**T**, **E**, **P**>, де **T** - власне текст; **E** - розширений текст, що включає умови його породження; **P** - розширений текст, який враховує інформацію про суб'єкта, який породжує текст.

Крім цього, введемо такі позначення: **TR** - формальні правила поповнення тексту, що базуються на його внутрішній структурі та на псевдофізичних логіках;

ER - правила поповнення тексту, що базуються на знаннях системи про світ; **PR** - правила поповнення тексту, що базуються на комунікативних та психологічних знаннях, а також на знаннях про особистість того, хто говорить: **A** - відповідь, що генерується системою.

Можна вважати, що система розуміє текст, якщо вона дає правильні відповіді на будь-які запитання, що мають відношення до тексту. В [1] наводиться така класифікація рівнів розуміння.

Перший рівень характеризується схемою **T -> A**, відповіді генеруються лише на основі прямого та безпосереднього сприйняття інформації. Нехай, наприклад, система сприймає текст

"О восьмій ранку Федя пішов на роботу. Він повернувся о восьмій вечора, після чого повечеряв і пішов спати".

При першому рівні розуміння система в змозі відповідати на запитання *"Коли Федя пішов на роботу?"* або *"Коли він повернувся?"*.

Другий рівень характеризується схемою **<T, TR> -> A**. Система в змозі відповідати на запитання типу *"Що було раніше: повернення Феді чи його вечора?"* або *"Чи пішов Федя спати після вечери?"*

Третій рівень характеризується схемою **<T, TR, ER> -> A**. Система може відповісти на запитання типу *"Де був Федя о четвертій дня?"*. Легко зрозуміти, що відповідь може істотно залежати від наявних знань.

Четвертий рівень характеризується схемою **<E, TR, ER> -> A**. Крім власне тексту, беруться до уваги інші джерела інформації. Наявність таких джерел дає можливість правильно сприймати висловлювання типу *"Подивись, яка погода! Чи варто їхати на пікнік?"*.

П'ятий рівень розуміння, який характеризується схемою **<P, TR, PR> -> A**, бере до уваги інформацію про конкретну особистість, що є джерелом тексту, а також загальну інформацію про психологію спілкування, про мету учасників спілкування тощо. Так, наприклад, якщо система сприймає слова Васі про те, що *"Таня обіцяла незабаром повернутися"*, слід брати до уваги:

- чи слід взагалі довіряти Васі?
- який зміст вкладає Вася у слово "незабаром?"
- що ми знаємо про Таню та про те, куди вона пішла?

Крім цього, розглядаються **метарівні розуміння**, які визначають можливі зміни у базах знань та сприйняття і породження метафор та асоціацій.

Легко зрозуміти, що ця класифікація не є єдиною можливою.

14.2. Системи обробки природної мови.

Системи обробки природної мови (СОПМ) розробляються для полегшення діалогу непрофесіонального користувача з ЕОМ і забезпечують реалізацію цього діалогу природною мовою.

Визначимо основні поняття [1], які будемо використовувати далі.

Спілкуванням будемо називати комунікативну взаємодію.

Під *діалогом* будемо розуміти процес досягнення його учасниками конкретних погоджених цілей шляхом обміну зв'язаними висловленнями деякою мовою в обраній предметній області.

Мета діалогу визначає *структуру* діалогу. Структура діалогу може розглядатись на трьох рівнях: *глобальному, тематичному і локальному*.

Глобальний рівень визначає загальні властивості вирішуваних задач. Тематичний рівень визначається алгоритмом вирішення поставленої задачі і розподілом ролей між учасниками діалогу. На локальному рівні аналізуються конкретні пари “*запит - відповідь*”.

Серед обов'язкових функцій СОПМ можна виділити наступні: *ведення діалогу* - визначення його структури і ролі, яку виконує пара користувач - система на даному кроку діалогу; *розуміння* - перетворення отриманих від користувача висловлень на природній мові (ПМ) на висловлення мови внутрішнього задання; *обробка висловлень* - формування і визначення завдань для вирішення задач і підзадач даного кроку діалогу; *генерація* - формування вихідних висловлень природною мовою.

До основних задач *компонент діалогу* (КД) відносяться ведення діалогу і формування та обробка перехоплень ініціатив. На основі аналізу структури і поточному стану діалогу КД формує (якщо ініціатива належить системі) або визначає (коли ініціатива належить користувачу) завдання, яке виконує система на поточному кроці: чи це є генерація питання, чи це є розуміння відповіді і її формування і т.д.

Діалог може вестись або користувачем, або ж системою. Якщо ролі учасників спілкування визначаються під час спілкування, тоді структуру діалогу називають *гнучкою*.

Друга задача КД полягає в обробці ситуації, коли реакція одного з учасників може не співпадати з очікуваннями іншого. Формування перехоплення відбувається, якщо система визнає, що поточна ситуація не відповідає ситуації, передбаченій структурою діалогу. Якщо ж перехоплення ініціативи виконує користувач, тоді система повинна обробити його, розпізнати наявність ініціативи, визначити нову тему спілкування і перейти до структури діалогу, яка відповідає новій темі.

Компонента розпізнання висловлень (КРВ) використовується для розпізнання суті вхідного висловлення і вираження його змісту на внутрішній мові системи. Зміст висловлення традиційно визначає: сутність проблемної області, властивості і відношення, притаманні цим сутностям, комунікативні наміри ініціатора, виражені в цьому висловленні.

Традиційна задача розуміння висловлень ділиться на два етапи: *аналіз і інтерпретація*.

На етапі аналізу виділяються опис сутностей, їх властивостей і відношення між ними.

Інтерпретація полягає в відображенні вхідного висловлення на знання системи. Основними задачами даного етапу є інтерпретація висловлення відносно намірів ініціатора. Потім, застосовуючи існуючі у системі методи виведення визначають, як це висловлення узгоджується з метою і планами учасників спілкування.

Компонента генерації висловлення (КГВ) вирішує у відповідності з результатами, отриманими попередніми частинами системи, дві основні задачі: генерацію змісту і синтез висловлення.

В загальному випадку, при вирішенні задачі формування змісту вихідного висловлення потрібно враховувати прагматичний аспект. В промислових системах спілкування ця задача вирішується шляхом редагування значень атрибутів або ж

вибором шаблону відповіді. В експериментальних системах для вираження змісту генерується повне семантичне задання, яке включає одну або декілька зв'язних подій (понять) з одним або декількома виконавцями на кожну роль.

Інша задача компоненти генерації висловлення полягає в синтезі мовно-природного висловлення, яке б відповідало внутрішньому формату висловлення. Ця задача вирішується в три етапи: семантичний, синтаксичний і морфологічний синтез. Складність задачі синтезу визначається вимогами до природності і описовій потужності вихідних висловлень.

Для розуміння принципів побудови СОПМ важливим моментом є питання про знання, які використовуються в системі.

14.3. Етапи аналізу речень природною мовою

Можна виділити такі етапи аналізу окремих речень природною мовою: **морфологічний**, на якому виділяються окремі слова та форми слів; **синтаксичний**, на якому з'ясовується структура речення та будується дерево граматичного розбору; використовуються різні типи граматик, які тим чи іншим чином наближаються до граматики, що породжує природну мову (останню до цього часу формалізувати не вдавалося). Часто використовуються контекстно-вільні граматики. Контекстно - вільні граматики мають ряд обмежень, зокрема, вони практично не дозволяють узгоджувати граматичні роди, відмінки та ін. Їх корисною модифікацією є **граматики визначених клауз**, які по-іншому називаються **граматиками визначених диз'юнктивів** [9]. Типовим є використання **розширених мереж переходів**, введених Вудсом [1, 9]. У найбільш загальних рисах, розширена мережа переходів являє собою модифікований граф переходів скінченного автомату, на дугах якого пишуться умови, при яких даний перехід може бути здійснений, а також дії, які необхідно додатково виконати при здійсненні переходу; **семантичний**, в процесі якого з'ясовується значення фрази та співставлення його з тими знаннями, які містилися в системі (сучасні методи семантичного аналізу фактично зводяться до конкретизації відповідних фреймів, "під'єднання" інформації, яка міститься у реченні до тих семантичних мереж, які були сформовані до цього, поповнення речень на основі псевдофізичних логік і сценаріїв тощо). Використання граматик визначених клауз та розширених мереж переходів дозволяє значною мірою поєднати синтаксичний та семантичний аналіз.

Крім цього, **необхідним є розуміння не тільки окремих речень, але й усього тексту.**

Розуміння текстів, написаних природною мовою, є великою та складною проблемою. Можна виділити такі аспекти розуміння, як **синтаксичний** (з'ясування структури речення), **семантичний** (з'ясування змісту речення) та **прагматичний** (з'ясування того, з якою метою було сказане те чи інше речення). Проблема розуміння на сучасному етапі є далекою від повного вирішення. Більш детальне висвітлення цієї теми можна знайти в [1, 5, 9].

14.4. Методи реалізації діалогу.

Діалог можна розглядати на трьох рівнях:

- загальна (глобальна) структура, яка характеризує тип діалогу і клас вирішуваних задач;
- тематична структура - описує структуру конкретної задачі;

- локальна структура, яка описує крок діалогу і відображає взаємодію учасників в елементарному акті діалогу.

На рівні глобальної структури для системи ПМ діалог задається послідовністю станів. Наприклад, глобальна структура експертної системи включає такі стани: інструктаж, визначення задач, вирішення задач, пояснення рішення задач, видача результатів рішення і їх аналіз, визначення причин невдач і набуття нових знань.

У конкретних системах окремі стани можуть бути пропущені або за замовчуванням, або ж за запитом до користувача.

Тематична структура діалогу традиційно задається *сценарієм*, який визначає: структуру задачі; розділ задачі на підзадачі; розподіл підзадач між учасниками діалогу; мовні засоби спілкування з користувачем.

Сценарії бувають або готові, або ж генеровані в процесі вирішення задач. Готовий сценарій в більшості випадків задається в вигляді частково-впорядкованої множини правил з параметрами, значення яких уточнюються в процесі вирішення задач.

Якщо ж, сценарій будується в процесі розв'язку задачі, тоді компонента діалогу включає спеціальний механізм - *планувальник*. Генерація сценарію діалогу бажана в ситуаціях, коли структура задачі залежить від контексту ситуації, в якій проводиться розв'язок.

При визначенні локальної структури крок діалогу складається з дії і реакції. Він характеризується наступними параметрами: ініціатор і тип ініціювання (вид дії); спосіб впливу дії на реакцію; спосіб специфікації підзадачі, яка вирішується на даному кроку.

На локальному рівні діалогу, задача КД-у, полягає в визначенні параметрів поточного кроку. Ініціатором кроку в загальному випадку може бути як користувач, так і система. Дія користувача ділиться на задання і команди.

14.5. Загальна структура експертних систем

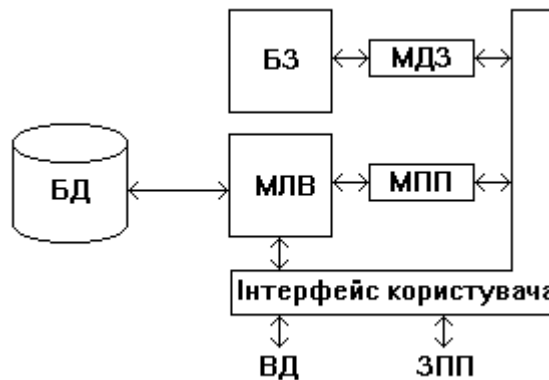
Як ми вже зазначали, задання знань визначає характеристики системи, а вибір способу задання знань важливий як для великих, так і для малих задач.

Системи, які засновані на знаннях - це програми, основними структурними елементами яких є база знань і механізм логічного виведення. Яскравим представником таких систем є **експертні системи (ЕС)**.

Експертна система - це інтелектуальна програма, здатна робити логічне виведення на основі знань в конкретній предметній області, яке забезпечує вирішення специфічних задач.

Необхідним етапом в розробці ЕС є здобуття відповідних знань від експерта. Можна виділити основні вимоги до знань ЕС: використання знань, пов'язаних з конкретною предметною областю; набуття знань від експертів; визначення реальної і достатньо складної задачі; надання системі здібностей експерта.

Експерти повинні задовольняти наступним характеристикам: мати значні знання в обраній предметній області; мати великий досвід в цій області; вміють точно сформулювати і правильно вирішити задачу.



Мал.14.1.Структура експертної системи

Загальна структура ЕС зображена на малюнку 4.1. Вона включає: *базу знань* - механізм задання знань в конкретній предметній області і керування ними; *механізм логічного виведення* - механізм, який на основі знань, що є в базі знань, здатний робити логічні заключення; *інтерфейс користувача* - для правильної і зручної передачі інформації користувачу; *модуль досягнення знань* - механізм отримання знань від експерта, підтримки бази знань і її доповнення; *модуль порад і пояснень* - дає заключення, задає різні коментарі до заключення, пояснює мотивування заключення.

Мова задання знань, яка використовується для розробки експертної системи, називається *мовою розробки ЕС*, а система програмування, яка виконує вказані вище функції називається *інструментарієм розробки*.

Під експертними системами можна розуміти інтелектуальні системи, здатні одержувати, накопичувати та коригувати знання з заданої предметної області, виводити нові знання, вирішувати на основі цих знань практичні задачі та пояснювати хід їх рішення. Конкретна предметна область передбачає обмежений набір явищ та понять з певної галузі людської діяльності та обмежене коло задач, які вирішуються в цій галузі.

Типове застосування експертних систем - консультації для фахівців середньої кваліфікації і неспеціалістів у тій галузі, для якої вона розроблена. Особлива увага при цьому приділялася таким галузям, як медична діагностика, юриспруденція, керування технологічними процесами, складання розкладів робіт і т.п. До системи знань експертної системи включають чіткі закони та певні евристики, якими володіє фахівець - експерт. Наприклад: "**якщо пацієнт чхає і кашляє, то у нього респіраторне захворювання**", або "**маслята слід збирати у соснових лісах**", або "**в дебюті шахової партії необхідно боротися за центр**".

Можна сказати, що експертні системи призначені для вирішення **неформалізованих задач**. В [1] наведені такі *характерні особливості неформалізованих задач*: алгоритмічне рішення невідоме; задача не може бути сформульована в числовій формі; необхідне символічне задання; неможливо чітко визначити цільову функцію, тобто критерій, за яким слід приймати рішення; знання про предметну область можуть бути неповними, помилковими або суперечливими.

Часто експертні системи називаються **системами підтримки прийняття рішень**, хоча останній термін, взагалі кажучи, є більш загальним.

Хороші експертні системи повинні відповідати на такі два типи запитань:

- **як експертна система прийшла до певного висновку;**
- **чому експертна система поставила користувачеві те чи інше питання.**

Експертним системам присвячена чимало літератури [1, 4, 6, 7 -9 та ін]. Перші такі системи з'явилися в середині 60-х років. Великий перелік реально існуючих експертних систем наведений в [1]. Як приклади можна навести: **MACSYMA** - дії з алгебраїчними виразами, вирішення рівнянь і т.п.; **MYCIN** - вибір антимікробної терапії; **PROSPECTOR** - пошук корисних копалин; **TAXMAN** - визначення податків; **СОНЭТ** - локалізація відмов в електронних схемах; **XCON** - планування конфігурації обчислювальної системи, вибір її компонентів; **PROJCON** - діагностика та відлагодження програмного забезпечення; **МОСЭД** - експертна система для вирішення задач математичної статистики; **М-1** - виявлення інфаркту міокарду.

Більшість експертних систем постійно розвиваються.

Неможливо створити універсальну експертну систему; такі системи повинні бути орієнтовані на вузьке коло практичних задач. Але існують спеціальні **оболонки для створення експертних систем**. Це інструментальні засоби, які являють собою "пусті" системи без наповнення їх конкретним змістом [1 та ін.]. Вони беруть на себе реалізацію зручного інтерфейсу між розроблювачем і системою. Наприклад, вони реалізують такі загальні операції, як виведення інформації на дисплей, введення нових фактів, логічні побудови, підтримку системи меню і т.п.

Створення експертних систем відбувається у тісній взаємодії спеціаліста зі штучного інтелекту (**інженера знань**) та фахівця у даній предметній галузі. Методи такої взаємодії можна знайти в [1 та ін.]. Тут виникає ряд проблем, таких, як: важко отримати від експерта знання у вигляді, придатному для занесення до експертної системи; експерт знає набагато більше, ніж сам усвідомлює, його мислення часто спирається на інтуїцію, і часто він не може чітко сформулювати свої знання; інструментальні оболонки, які можуть бути використані при створенні експертних систем далеко не універсальні; часто буває так, що їх зручно використовувати для однієї предметної області, і зовсім незручно - для іншої.

14.6.Синтез зв'язних текстів

Іншою важливою проблемою, яка виникає при спілкуванні, є **синтез зв'язних текстів** на основі наявних знань. В основі існуючих підходів лежать дослідження по виділенню типових структур речень та типових комунікативних функцій. У багатьох моделях використовуються **риторичні предикати**, які є базовими при побудові описових речень. Це такі предикати, як пояснення, аналогія, атрибуція, відмінність та ін. Таких предикатів налічується два-три десятки. Крім того, при діалозі фіксуються певні **схеми діалогу**.

В інтелектуалізованих інформаційних системах вузького призначення широко використовується такий прийом: жорстко задається структура речення-відповіді системи, в яку лише підставляються ті чи інші значення полів бази даних.

При організації діалогових людино-машинних систем зовсім не обов'язково підтримувати діалог природною мовою. Широко використовуються спрощені мови, а також підказки, системи меню і т.п.

Розвивається **задачний підхід** до організації людино-машинного діалогу. **При цьому підході діалог розглядається як сумісне вирішення певної задачі** [Д.9, Д.10].

14.7. Загальна характеристика систем машинного перекладу

Слід окремо виділити **системи машинного перекладу**. Найпростіший (морфологічний) підхід до проблеми полягає в такому. Можна ввести в пам'ять комп'ютера словник (що зробити неважко), і при перекладі замінювати кожне слово початкового тексту на його словниковий еквівалент. Цей підхід є добрим в усіх відношеннях, крім одного: текст результату неможливо зрозуміти.

Машинний переклад слід розглядати у загальному контексті розуміння і спілкування, але ця задача має ряд особливостей. Якщо, наприклад, ряд задач ситуаційного керування дозволяють обійтись без синтезу текстів, то машинний переклад без цього неможливий. З іншого боку, перекласти ряд простих речень можна навіть не розуміючи їх змісту, тільки на основі морфологічного аналізу (наприклад, "**Jim killed John**" ("**Джім убив Джона**"). А для розуміння і перекладу українською мовою, наприклад, такої російської фрази, як "**Он видел их семью своими глазами**" навіть синтаксичного розбору недостатньо, оскільки його результати неоднозначні.

Як правило, сучасні системи машинного перекладу генерують попередній текст. Цей текст у стилістичному та граматичному відношенні дуже далекий від ідеального, але він, як правило, є зрозумілим, особливо при доброму знанні предметної області. Цей попередній текст потім остаточно редагується людиною.

ЗАПИТАННЯ І ВПРАВИ ДО РОЗДІЛУ

1. Охарактеризуйте в загальних рисах проблему розуміння і спілкування.
2. Опишіть відомі Вам п'ять рівнів розуміння.
3. Опишіть загальну структуру систем обробки природної мови.
4. Назвіть основні етапи аналізу речень природною мовою.
5. Дайте визначення експертних систем; наведіть приклади.
6. Опишіть в загальних рисах основні проблеми, пов'язані з машинним перекладом.
7. Граматика

Фраза -> Група_Підмету, Група_Присудку

Група_Підмету -> Прикметник, Іменник

Група_Присудку -> Прислівник, Дієслово, Іменник

Прикметник -> Слово_1| Слово2|...

породжує фрази типу "**Зловмисник украв гроші**" або "**Маленька дівчина нагодувала цуцика**". Напишіть програму, яка може відповідати на запитання типу "**Хто вкрав що?**" або "**Кого нагодувала дівчина?**". Програма має також співставляти введені фрази з **фреймами** "**Хороший вчинок**" і "**Поганий вчинок**" та відповідати на запитання типу "**Хто вчинив хороший (поганий) вчинок?**".

Розділ 15. Моделі і методи розпізнавання зразків

- *Тату, а що означає "двоїться в очах"?*
- *Бачиш, синку, он дві берези на пагорбі.
Коли двоїться в очах, здається, що їх
чотири.*
- *Тату, але там лише одна береза...*
(З анекдоту)

15.1. Загальна характеристика

Моделі і методи розпізнавання зразків займають одне з ключових місць у теорії та практиці систем штучного інтелекту. Навряд чи є особлива потреба у якомусь всеохоплюючому визначенні поняття "розпізнавання зразків". Є ряд теоретичних та практичних задач, які розглядаються в рамках цього напрямку, далі буде зроблений загальний огляд цих задач та методів їх вирішення.

Інтуїтивно розпізнавання зразків можна визначити як процес прийняття рішення про те, до якого класу явищ або предметів відноситься явище або предмет, що спостерігається інтелектуальною системою.

У літературі на цю тему спостерігається деяка неоднозначність. Інколи "зразком" називається клас об'єктів, а окремі об'єкти називаються його представниками. Інколи під "зразками" маються на увазі окремі представники, а їх сукупність називається класом, що об'єднує різні поодинокі зразки.

Надалі ми будемо використовувати більш нейтральні терміни "клас" і "об'єкт", або "представник класу".

Розпізнавання зразків легко здійснюється людським мозком, але реалізувати його на комп'ютерних системах виявляється дуже непростою задачею. Воно має велике значення для систем штучного інтелекту, і, зокрема, для інтелектуальних роботів, оскільки завдяки йому вони отримують можливість безпосередньо сприймати оточуючий світ. Методи розпізнавання зразків широко застосовуються на практиці [1, 3, 15, 16]. Як приклади можна навести застосування на виробництві (відбраковка дефектних деталей, розпізнавання написів на тарі, визначення типу та орієнтації предметів), у медичній діагностиці, на транспорті та у зв'язку, у сфері торгівлі та бізнесу (зчитування штрих-кодів, перевірка банкнот) та ін. Особливо слід відмітити такі практичні застосування, як розпізнавання літер та розпізнавання мови. Вони широко використовуються у сучасних інформаційних технологіях (сканування тексту, надиктовування текстів, керування персональним комп'ютером з голосу).

Можна навести характерні риси сукупностей предметів або явищ, і ці характерні риси дозволяють говорити про ці сукупності як про "зразки" та зумовлюють доцільність постановки задачі розпізнавання зразків (в основному ці риси описані згідно з [16]).

Предмети або явища, що належать одному класові, **схожі** між собою. Під схожістю мається на увазі наявність певних ознак, на основі яких можна відокремити об'єкти, що належать даному класу, від об'єктів, що йому не належать. Часто міру схожості вдається виміряти кількісно.

Об'єкти, що належать одному класу, можуть дуже змінюватися, часто досить-таки сильно. Часто встановити чіткі границі зразків неможливо. Якби наше сприйняття представників одного й того самого класу були ідентичними, розпізнавання стало б тривіальною задачею. Ця змінність може бути зумовлена різними факторами: помилками при вимірюваннях; помилками, що вносяться зовнішніми факторами (шуми при передачі по каналу зв'язку, затінення об'єкту та ін.), власною змінюваністю об'єктів. Особливо гостро проблема змінюваності стоїть у випадку плавного переходу одного класу в інший, класичним прикладом є перетворення пуголовка на жабу;

Ознайомлення з деякою скінченою кількістю представників класу дає змогу розпізнавати довільну кількість інших представників (можливо, з певним рівнем достовірності) [16, Д.6]. Зразками можуть бути, наприклад, "ріка", "море", "рідина". Таке ознайомлення можна охарактеризувати як **навчання, а множина представників класу, на основі яких формується **правило розпізнавання**, називається **навчальною вибіркою**. Правило розпізнавання, або правило вирішення - це алгоритм, який відносить деякий об'єкт до одного з класів. Зрозуміло, що таке рішення може бути правильним або помилковим.**

До навчальної вибірки часто включаються **контрприклад**и, тобто типові представники інших класів. Часто розглядається **екзаменаційна вибірка**, на підставі розпізнавання елементів якої робиться висновок про якість навчання.

Таким чином, у задачі розпізнавання можна виділити три частини: навчання, екзамен і власне розпізнавання.

У [Д.6] розглядається один з класичних підходів до навчання розпізнаванню та формування понять на основі перебору можливих ознак та виділенні їх інформативних комбінацій.

Окремо можна виділити задачу розпізнання без учителя, або **кластерного аналізу**. Для цієї задачі характерним є те, що класи попередньо не задаються; система повинна сама розділити відомі їй об'єкти на класи.

Зразок у розумінні "класу" визначається у [16] як деяка сукупність об'єктів, що мають спільні властивості, або певні спільні ознаки. Там же наводиться більш формалізоване визначення: об'єкти, для яких виконується відношення еквівалентності, або у крайньому разі відношення толерантності, у своїй сукупності складають образ.

Нагадаємо, що відношення еквівалентності - це відношення, яке задовольняє властивостям симетричності, рефлексивності і транзитивності. Відношення толерантності - це відношення, для якого виконуються лише симетричність і рефлексивність, а транзитивність може не виконуватися.

Більш природним, ніж через відношення толерантності, видається **визначення зразку (класу) у рамках теорії нечітких множин**.

При вивченні систем розпізнавання варто виділити **розробку системи розпізнавання та власне розпізнавання**.

При **власне розпізнаванні** об'єкту, що подається на вхід системи, традиційно виділяють три етапи: **отримання первинної інформації; виділення ознак; класифікація, застосування правила вирішення, тобто віднесення об'єкту до одного з класів**.

Один з класичних прикладів наведений у [Д.13]. Нехай на деякій деревообробній фабриці потрібно автоматизувати процес сортування деревини.

Фабрика має справу з двома видами деревини: березою та ясенем. Фотокамера отримує первинну інформацію (зображення деревини) і передає на видільник ознак. Для видільника ознак потрібні якісь апіорні дані про березу і ясеня. Нехай відомо, що береза, як правило, світліша за ясеня. Відповідно, ця ознака може бути використана для розпізнавання, але однієї цієї ознаки недостатньо. Інша ознака - помітність волокон. Волокна ясеня звичайно більш помітні.

Якщо прийнято рішення розрізняти деревину за цими двома ознаками, основна задача видільника ознак - спроектувати кожен з отриманих картинок на деяку точку у двовимірному просторі ознак. Значення самих ознак відкладаються по координатним осям цього простору. Це стосується і навчальної вибірки, і нових об'єктів.

На етапі навчання простір ознак розділяється на області, наприклад, за допомогою лінійної розділяючої функції. В процесі розпізнавання нових об'єктів класифікатор аналізує значення ознак та на основі цього аналізу визначає, до якої з областей потрапляє новий об'єкт.

Легко зрозуміти, що всі ці етапи тісно пов'язані між собою. Уже з цього прикладу видно, що **ідеальний класифікатор не потребує виділення ознак, і навпаки, при ідеальному виділенні ознак робота класифікатора максимально спрощується**. Основну проблему можна сформулювати таким чином: **реальні методи отримання первинної інформації та доступні алгоритми виділення ознак часто не дозволяють отримати саме ті ознаки, які насправді відрізняють представників одного класу від представників іншого**.

При розробці системи розпізнавання можна виділити такі ключові задачі, які мають бути вирішені на цьому етапі: **формування навчальної вибірки; вибір інформативної множини ознак; навчання, тобто формування правила вирішення, згідно з яким буде здійснюватися розпізнавання**.

При виборі методу класифікації, інформативних ознак та формуванні навчальної вибірки важливо намагатися мінімізувати кількість помилок. Велике значення має **ціна кожної помилки**. Переплутати березу з ясенем, очевидно, не так страшно, як, наприклад, прийняти дефектну деталь за справну.

Інколи розрізняють **розпізнавання на основі даних, що сприймаються органами почуттів** та **розпізнавання на основі даних довільного характеру**.

В [16] описані прості спеціалізовані методи розпізнавання: **метод співставлення з еталоном, метод зондів, метод маркування, метод квазитопологічного розпізнавання**. Ці методи вимагають детального вивчення об'єктів, що розпізнаються. Навчання фактично відсутнє, і об'єкти, які розпізнаються, мало змінюються.

Розглянемо для прикладу **метод суміщення з еталоном**. Основний принцип цього методу полягає в суміщенні зображення з набором заздалегідь сконструйованих ідеальних еталонів. Належність зображення до деякого класу визначається мірою подібності зображення до еталону. Є різні варіанти технічної реалізації цього методу, такі, як використання фотомасок, еталони на основі електричних моделей та ін.

Основні сучасні методи розпізнавання відносяться до одного з двох класів: **методи на основі статистичної теорії рішень** (інколи вони називаються дискримінантними) та **структурні** (або синтаксичні, або лінгвістичні). В рамках статистичної теорії рішень кожний об'єкт розглядається як точка в деякому

просторі ознак. Віднесення об'єктів до одного з виділених класів звичайно відбувається шляхом розділення цього простору на області; кожна область відповідає окремому класові.

В рамках структурних методів розпізнавання здійснюється шляхом аналізу структури об'єктів. Кожний об'єкт розглядається як такий, що складається з певної кількості атомарних елементів; аналізується, яким саме чином вони між собою пов'язані.

Можна стверджувати, що **вибір методу розпізнавання істотно залежить від апіорної інформації про об'єкти, що розпізнаються.**

15.2. Статистичні методи розпізнавання

Наведемо одну з **типових постановок задачі розпізнавання образів**, характерну для статистичних методів. Вона дещо звужена порівняно з найбільш загальними постановками, але має величезне практичне значення. У загальних рисах вона відповідає математичній постановці, наведеній в [1].

Заданий певний набір класів $K_1, \dots, K_M$. Для кожного класу заданий деякий набір його представників. Потрібно для будь-якого об'єкту визначити, до якого класу він відноситься.

Кожний об'єкт характеризується **певним набором ознак**, тобто відображається на деяку точку простору ознак. Цей набір ознак є фіксованим та спільним для всіх об'єктів.

Таким чином, навчальній вибірці відповідає **матриця даних** $Y = (y_{ij}; i = 1, \dots, q; j = 1, \dots, n+1)$. Тут q - загальна кількість представників усіх класів, n - кількість ознак (розмірність простору ознак) При цьому y_{in+1} - клас, до якого відноситься i -й об'єкт.

Будь-якому об'єктові відповідає вектор даних $x = (x_j; j = 1, \dots, n)$.

При цьому природно визначити **відстань** у просторі ознак, яка являтиме собою мірило схожості об'єктів.

Статистичні методи явно або неявно спираються на **гіпотезу компактності** [16]. Вважається, що класові відповідає компактна множина точок у деякому просторі ознак. Компактною у даному розумінні називається множина точок, для якої: число граничних точок мале порівняно з загальним числом точок; будь-які дві внутрішні точки можуть бути з'єднані плавною лінією так, щоб ця лінія проходила лише через точки цієї ж множини; майже будь-яка внутрішня точка має в достатньо великому околі лише точки цієї ж множини.

Виділяють **різні типи ознак**: *дихотомічні* (ознака може бути присутня або відсутня; наприклад - є крила або немає крил); *номінальні* (наприклад, колір: червоний, синій, зелений і т.п.); *порядкові* (наприклад, "великий" - "середній" - "маленький"); *кількісні*.

Найбільш розвиненою можна вважати теорію схожості для кількісних ознак. Можуть використовуватися різні метрики. Основними серед них є евклідова відстань та зважена евклідова відстань.

Для не кількісних використовуються інші відстані: Хеммінга, Кендала, Юла, Рассела і Рао та ін.

Існують **різні методики переведення одних типів ознак до інших.**

І матрицю даних, що відповідає навчальній вибірці, і вектор даних, що відповідає новому об'єктові, у більшості випадків можна розглядати як результати виділення ознак на основі первинних даних. Під "*виділенням ознак*" мається на увазі не вибір інформативних ознак (з цим пов'язані інші задачі), а **отримання конкретних значень для зафіксованого набору ознак**.

"*Первинні дані*" можуть бути дуже різної природи. Обмежимося розпізнаванням неперервних сигналів та зображень. Типова схема розпізнавання таких об'єктів у загальних рисах відповідає раніше описаній схемі і включає до себе **датчики, блок попередньої обробки, блок класифікатора і пам'ять**.

За допомогою спеціалізованих датчиків збираються первинні дані. Стандартним прийомом є **дискретизація, або цифрування**. Цифрування сигналу, тобто деякої неперервної функції, яка залежить від часу, полягає у вимірюванні значень цього сигналу через певні проміжки часу, або з певною частотою.

Дискретизацію можна розглядати як перехід від деякої неперервної величини **$u(t)$** до **n -вектора** її значень у дискретних точках **t_j**

$$\begin{aligned} u(t) &\rightarrow [u_1, \dots, u_n]; \\ u_j &= u(t_j), j=1, \dots, n; \end{aligned}$$

у найпростішому випадку відстань між сусідніми точками **t_j, t_{j+1}** не залежить від **j** :

$$t_{j+1} - t_j = \Delta t.$$

В принципі, ці початкові дискрети можуть бути використані як ознаки сигналів, хоча розмірність простору ознак при цьому, як правило, виявляється неприпустимо великою.

Частота дискретизації не може бути довільною, при надто низькій частоті відбувається втрата інформації, а надто висока частота пов'язана як з надмірними вимогами до об'єму пам'яті, так і зі збільшенням похибок.

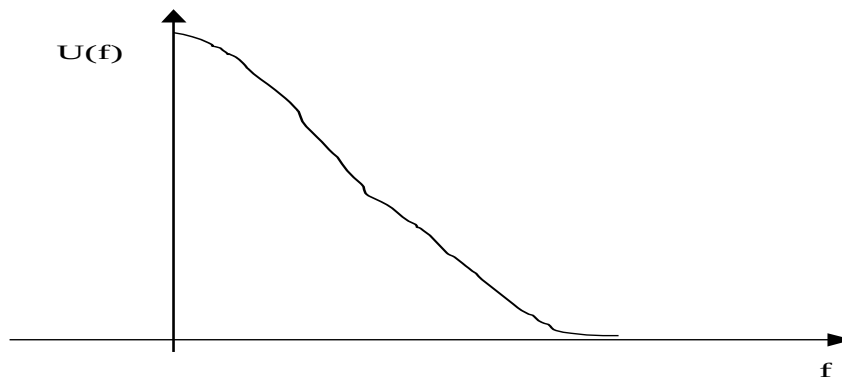
Теорема Котельникова (в іноземній літературі вона інколи називається теоремою Шеннона, або теоремою відліків) встановлює, якою в принципі повинна бути частота дискретизації. Сформулюємо її.

Теорема. Нехай для функції **$u(t)$** визначене перетворення Фур'є, і спектр цієї функції неперервний та обмежений смугою частот від **0** до **F** . Тоді функція повністю визначається своїми дискретами, взятими через інтервали часу **$\Delta t = 1 / (2F)$** .

Перетворення Фур'є **$U(f)$** функції **$u(t)$** визначається за формулою :

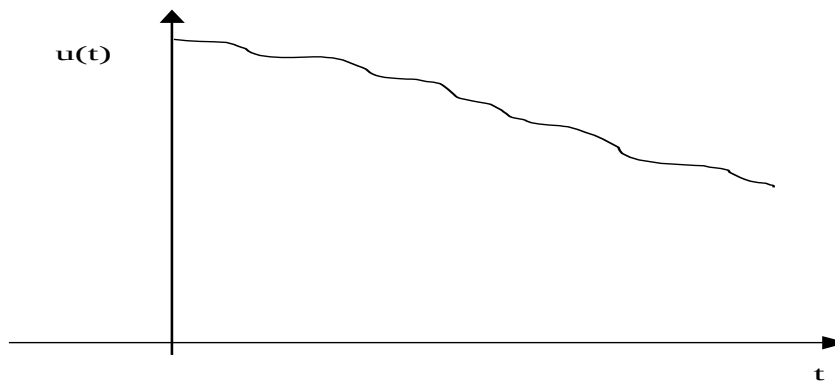
$$U(f) = \int_{-\infty}^{\infty} u(t) e^{-2\pi i f t} dt$$

Функція **$U(f)$** називається **частотним спектром сигналу**. За його допомогою можна охарактеризувати швидкість зміни сигналу в часі. Якщо більша частина спектру зосереджена в районі точки **$f=0$** , тоді сигнал є низькочастотним. Типовий спектр такого сигналу має вигляд



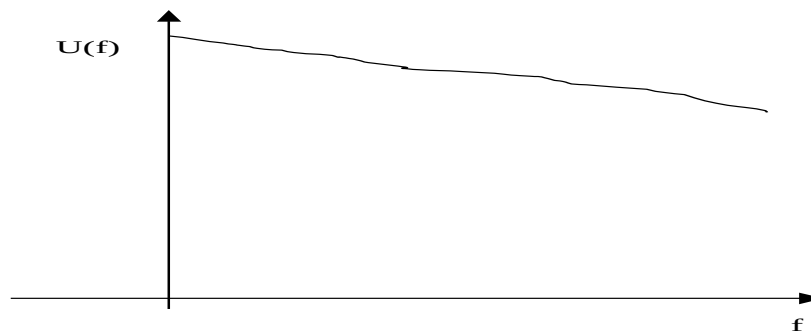
Мал. 15.1. Типовий спектр низькочастотного сигналу

При такому спектрі значення сигналу у різних точках сильно залежить одне від одного, і сигнал змінюється плавно:



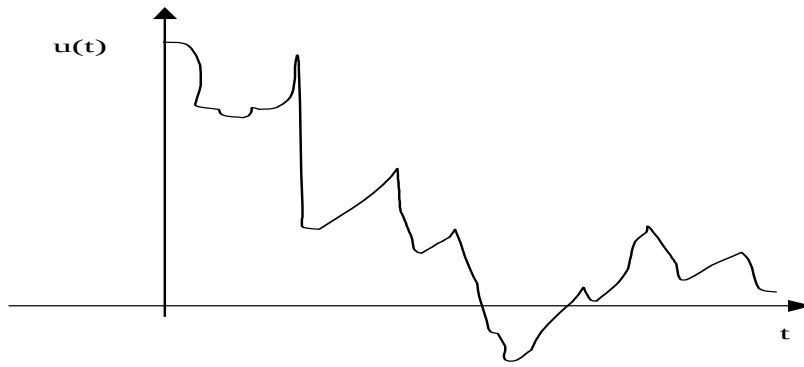
Мал.15.2. Типовий вигляд низькочастотного сигналу

Навпаки, типовий спектр високочастотного сигналу має вигляд:



Мал. 15.3. Типовий спектр високочастотного сигналу

Такий сигнал змінюється швидко:



Мал.15.4. Типовий вигляд високочастотного сигналу

Зрозуміло, що для низькочастотного сигналу частота дискретизації може бути більшою, і для високочастотного - меншою.

До класичних методів виділення ознак належать **глобальні перетворення**: лінійні або нелінійні. До таких перетворень відносяться перетворення Фур'є, Уолша, Хаара [15, Д.11]. **В якості ознак розглядаються спектральні коефіцієнти відповідних рядів. Основним призначенням глобальних перетворень для задачі розпізнавання образів слід вважати різке скорочення розмірності простору ознак ("стиснення даних").**

Неперервна функція $u(t)$ може бути розкладена на інтервалі спостереження $[a,b]$ в ряд:

$$u(t) = \sum_{k=1}^{\infty} c_k \varphi_k(t)$$

Функції $\varphi_k(t)$, як правило вибираються ортонормованими на $[a,b]$.

Коефіцієнти прямують до нуля, і, більш того, вищенаведений ряд збігається. Тому можна вважати :

$$u(t) \approx \sum_{k=1}^r c_k \varphi_k(t)$$

Коефіцієнти c_k , $k=1, \dots, r$ можна вважати ознаками сигналів.

Відкидання "хвостових" членів ряду призводить до похибок. Серед лінійних перетворень оптимальним за критерієм середньоквадратичної похибки є **перетворення Карунена - Лоева** [15, Д.12]. Можна сказати, що **це перетворення забезпечує найбільший стиск даних серед усіх лінійних перетворень**. Основна ідея цього перетворення полягає у **декореляції** вихідних даних та у **переході від початкових ознак до нових, які є лінійно незалежними**. Усунення лінійної залежності **усуває надлишковість, присутню у початкових даних**.

Задача стиску інформації для неперервних функцій на основі цього перетворення може бути математично сформульована таким чином.

Нехай U - деякий набір неперервних функцій $\xi_i(t)$, $i = 1, \dots, q$. Потрібно для довільної функції $\eta(t) \in U$ приблизно отримувати її інтегральне перетворення, тобто вектор

$$\mathbf{v}_k = \int_a^b \varphi_k(t) \eta(t) dt, k = \overline{1, r}$$

при заданих умовах на число g ;
 $\varphi_k(t)$ - власні функції рівняння

$$\int_a^b R(t,s)\varphi_k(s)ds = \sigma_k \varphi_k(t),$$

які відповідають g найбільшим власним значенням σ_k , розташованим в порядку спадання;

$R(t,s)$ - коваріаційна функція сигналів. Вона оцінюється на основі аналізу деякої базової вибірки V , наприклад, за формулою

$$R(t,s) = \sum_{i=1}^q \alpha_i \xi_i(t) \xi_i(s),$$

де q - об'єм базової вибірки; α_i - задані вагові коефіцієнти.

Проте, незважаючи на ряд теоретичних переваг, практична реалізація перетворення Карунена-Лоева пов'язана з рядом труднощів. Так, на відміну від перетворення Фур'є, для нього невідомі швидкі алгоритми.

У [15] описані **локальні перетворення** (сегментація, виділення контурів, розбиття на області та ін). Слід відмітити, що перетворення цього класу більш характерні для структурних методів розпізнавання.

Попередня обробка включає, крім власне виділення ознак, **зменшення шумів**. До шумів відносяться похибки вимірювання, каналів зв'язку та ін. Широко застосовуються фільтри Калмана, Вінера та ін. Заслужують на увагу методи, спрямовані на зменшення "неідеальності" вимірювальної апаратури.

Серед власне статистичних методів розпізнавання виділяються два великі класи: **ймовірнісні** та **детерміністські**.

Імовірнісні методи розглядають матрицю даних як набір реалізації деякого випадкового процесу. Класичним є **байєсівський підхід**.

Нехай є m класів. Випадок W_j , $j = 1, \dots, m$ означає, що об'єкт належить до j -го класу. Система розпізнавання може прийняти рішення α_i , яке означає, що невідомий об'єкт відноситься системою до i -го класу.

Ідея байєсівського підходу полягає в мінімізації ризику, пов'язаного з помилковим рішенням. Цей ризик можна розрахувати, якщо відомі умовні густини розподілу векторів ознак за умови, якщо відомо, що об'єкт належить до певного класу.

Більш детальний опис байєсівських процедур розпізнавання можна знайти в [Д.13].

Детерміністський підхід не залучає до розгляду ймовірнісних характеристик. Розподіл простору ознак на ділянки, які відповідають класам, відбувається на основі безпосереднього аналізу відстаней між об'єктами, що відповідають точкам цього простору.

Класичним є **правило найближчого сусіда**. Об'єкт відноситься системою розпізнавання до того класу, до якого належить його найближчий сусід з навчальної вибірки. Метод не дає помилок на об'єктах навчальної вибірки і фактично не вимагає вироблення вирішувального правила. Але його недоліком є те, що в пам'яті доводиться зберігати всю навчальну вибірку.

При застосуванні правила найближчого сусіду існує можливість скоротити повну навчальну вибірку. Для цього можна використовувати так зване **стиснене правило найближчого сусіду**, запропоноване Хартом. Далі наводиться алгоритм цього правила, описаний на основі [Д.12].

Для роботи алгоритму потрібні дві робочі множини - **ПАМ'ЯТЬ** і **ВІДСІВ**. Алгоритм виконує такі дії:

1. *Перший об'єкт з навчальної вибірки заноситься в **ПАМ'ЯТЬ**.*
2. *Кожний наступний об'єкт класифікується за допомогою правила найближчого сусіда, але при цьому використовуються лише ті об'єкти, які знаходяться в множині **ПАМ'ЯТЬ**. Якщо даний об'єкт класифікується правильно, він викидається у **ВІДСІВ**, інакше він заноситься в **ПАМ'ЯТЬ**.*
3. *Після перегляду всіх об'єктів описана процедура повторюється знову, але лише для елементів множини **ВІДСІВ**. Ці цикли припиняються або при повному вичерпанні **ВІДСІВ**, або якщо при черговому проході жодний об'єкт з **ВІДСІВ** не перейшов до **ПАМ'ЯТЬ**.*
4. *Після завершення алгоритму як скорочена навчальна вибірка використовується множина **ПАМ'ЯТЬ**.*

Модифікацією методу найближчого сусіду є **правило k найближчих сусідів**. Об'єкт відноситься системою розпізнавання до того класу, до якого належить більшість з **k** його найближчих сусідів, що належать до навчальної вибірки.

Дуже поширеним є **метод розділяючих функцій**. Функція **g(y)** розділяє два класи, якщо при **g(y) > 0** приймається рішення про належність вектора **y** одному класові, а при **g(y) < 0** - іншому.

Часто функції вирішення приймаються лінійними:

$$g(y) = w_1y_1 + \dots + w_ny_n + w_{n+1}.$$

Коефіцієнти **w_i** повинні бути підібрані в процесі навчання.

Метод розділяючих функцій та його зв'язок з дискримінантним аналізом детально описано в [Д.13].

15.3. Структурно-синтаксичні методи розпізнавання

Структурні (синтаксичні, лінгвістичні) методи розпізнавання [17, 15] беруть до розгляду **структуру об'єктів**, які розпізнаються. Розглядаються складові частини об'єктів та зв'язки між ними. Така декомпозиція здійснюється, поки ми не дійдемо до атомарних, неподільних елементів. Можна казати, що **кожний клас описується деякою формальною мовою**, і тоді **набір атомарних елементів являє собою алфавіт цієї мови**, а **представники класу описуються фразами, або реченнями цієї мови**. Правила формування фраз з атомарних елементів складають **граматику мови**.

Нагадаємо, що **граматикою** називається четвірка **G=<V_N, V_T, P, S>**, де **V_T** - основний словник (словник термінальних символів); скінчена множина основних символів; **V_N** - допоміжний словник (словник нетермінальних символів); **V_T** та **V_N** не мають спільних елементів; **P** - скінчена множина правил виведення, або правил підстановки (ці правила мають вигляд **a -> b**, де **a, b** - ланцюжки символів з **V = V_N ∪ V_T**, причому **a** містить хоча б один символ з **V_N**; дія правила підстановки

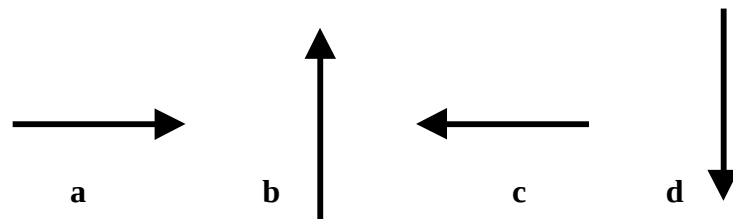
полягає у заміні в деякій фразі ланцюжка **a** на ланцюжок **b**); **S** - початковий символ, при цьому $S \in V_N$.

Мовою $L(G)$, що породжується граматикою **G**, називається сукупність ланцюжків $L(G)=\{x: x \in V_T^*, S \Rightarrow^* x\}$; тут V_T^* - множина всіх ланцюжків, утворених з елементів V_T ; $S \Rightarrow^* x$ означає, що **x** може бути утворене з **S** шляхом застосування однієї або декількох підстановок.

Іншими словами, мова, породжена граматикою - це сукупність всіх ланцюгів, що складаються лише з основних символів на основі послідовного застосування правил граматики до початкового символу.

Розглянемо приклад [15]. Нехай

$V_T = \{a,b,c,d\}$, **a,b,c,d** відповідають одиничним векторам



Мал. 15.5. Чотири атомарні елементи, що ілюструють граматичне породження зображень

Далі $V_N = \{S,A,B,C,D\}$; правила мають вигляд :

$S \rightarrow aA,$

$S \rightarrow bB,$

$S \rightarrow cC,$

$S \rightarrow dD,$

$A \rightarrow b,$

$B \rightarrow c,$

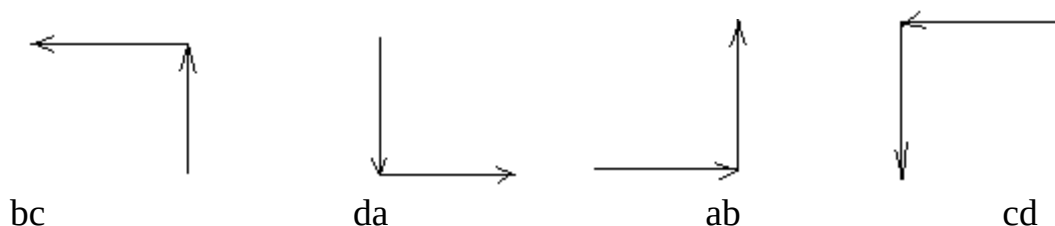
$C \rightarrow d,$

$D \rightarrow a.$

Тоді граматика породжує чотири фрази:

$S \Rightarrow aA \Rightarrow ab$; $S \Rightarrow bB \Rightarrow bc$; $S \Rightarrow cC \Rightarrow cd$; $S \Rightarrow dD \Rightarrow da$.

Відповідно за допомоги розглянутої мови можна описати чотири форми прямого кута



Мал. 15.6. Зображення, породжені граматикою

Іншою формою структурного опису зразків можна вважати граф відносин (семантичну мережу) [1, 4].

Типова система синтаксичного розпізнавання образів складається з трьох блоків: блок попередньої обробки; блок представлення об'єкта, метою якого є виділення атомарних елементів та відношень між ними; блок синтаксичного аналізу.

Крім того, на етапі навчання часто буває потрібним вирішити задачу **відновлення граматики за навчальною вибіркою.**

Методи попередньої обробки (дискретизація, зменшення шумів, сегментація та ін.) в основному співпадають з методами, які використовуються в статистичних методах розпізнавання. Атомарні елементи розпізнаються несинтаксичними методами.

Основною задачею блоку представлення об'єкта є опис об'єкта деякою фразою, тобто деяким ланцюжком, що складається з атомарних елементів.

Блок синтаксичного аналізу перевіряє, чи можна вивести утворений ланцюжок на основі граматики, що відповідає одному з класів. Якщо це вдається, об'єкт відноситься системою до цього класу. Синтаксичний аналіз полягає у граматичному розборі фраз, тобто у встановленні того, які можна застосовувати правила підстановки для отримання потрібної фрази. Є дві стратегії граматичного розбору: **зверху вниз та знизу вгору.**

Традиційні граматики для утворення фраз використовують ланцюгову операцію конкатенації. Для опису складних зображень запропоновані більш складні граматики [17, 18]: графові граматики, мова опису зображень **PDL**, веб-граматики, плекс-граматики та ін. Розглянемо досить загальний підхід, заснований на використанні **плекс-граматик.**

Плекс-граматики були розроблені Федером на основі ідеї Нарасімхана. Ідея полягає у формалізації опису з'єднань між елементами, що можуть мати довільну кількість точок підєднання. Символ з N точками приєднання називається **N -конфігурацією (NAPE).** Структури, які утворюються взаємозв'язаними **NAPE**, називаються плекс-структурами. Множини плекс-структур утворюють **плекс-мови**, граматики для опису плекс-мов називаються **плекс-граматиками.**

Плекс-граматика формально описується як шістка $\langle V_N, V_T, P, S, Q, q_0 \rangle$, де V_N - скінчена не пуста множина допоміжних (нетермінальних) **NAPE**; V_T - скінчена не пуста множина основних (термінальних) **NAPE**; V_T та V_N не мають спільних елементів; P - скінчена множина правил виведення, або правил підстановки; S - початковий символ, що називається початковою **NAPE**; при цьому $S \in V_N$; Q - скінчена множина ідентифікаторів, що відповідають точкам примикання; q_0 - спеціальний ідентифікатор, який називається пустим ідентифікатором.

Правило підстановки плекс-граматики у найбільш загальному вигляді записується як $\psi \Gamma_{\psi} \Delta_{\psi} \rightarrow \omega \Gamma_{\omega} \Delta_{\omega}$ де ψ - список замінюваних конфігурацій; ω - список конфігурацій заміщення; Γ_{ψ} - список замінюваних точок з'єднань; Γ_{ω} - список

точок з'єднань, які замінюють; Δ_{Ψ} - список точок зчеплень замінюваної частини; Δ_{ω} - список точок зчеплень частини заміщення.

Списки конфігурацій замінюваної і замінюваних частин - це ланцюжки вигляду $\Psi = a_1 a_2 \dots a_m$; $\omega = b_1 b_2 \dots b_n$, де a_i та b_i - NARE.

Списки точок з'єднання визначають, яким чином NARE у списках конфігурацій зв'язані між собою. Вони діляться на поля. Поле - це список вигляду $(q_{i1} q_{i2} \dots q_{ik})$, що визначає, які точки примикання яких NARE беруть участь у даному з'єднанні. Одне поле відповідає одному з'єднанню. Входження q_i в деяке поле на j -му місці, означає, що j -а компонента списку конфігурацій бере участь у даному з'єднанні і з'єднується за допомогою i -ї точки примикання. Якщо, j -та компонента не включена до з'єднання, в j -й позиції відповідного поля повинен стояти ідентифікатор q_0 .

Списки точок зчеплень встановлюють відповідність між зовнішніми зв'язками частин заміщення. Кожній точці зчеплення, що стоїть на p -му місці у списку точок зчеплення замінюваної частини, повинна відповідати точка зчеплення частини, яка її заміщує.

Відомими є **метод потенціальних функцій, метод групового урахування аргументів, метод припустимих перетворень, логічні методи розпізнавання**. Останні також традиційно відносяться до структурних методів. В їх основі лежить представлення відповіді про належність об'єкта до певного класу як логічної функції від його ознак. Ознаки при цьому є булевими. Класифікація при такому підході аналогічна логічному виведенню в експертних системах.

В [1] відмічається, що на розвиток теорії розпізнавання образів великий вплив справили такі дослідження: **алгебраїчний підхід до задачі розпізнавання**; розвиток цього підходу в основному пов'язується з іменем Ю.Журавльова; в основі підходу лежать **алгоритми обчислення оцінок та алгебра операторів розпізнавання** [1]; **ідея комбінаторної регулярності** (роботи Гренандера [Д.14]); **концепція парадигматичного символу**.

В основі теорії У. Гренандера лежить уявлення про **структурованість** світу, тобто існування регулярності, що проявляється у вигляді постійних зв'язків та закономірностей. Образи розглядаються у рамках деякого точного формалізму, що є основою для їх синтезу та аналізу. Формально, мова йде про побудову нових об'єктів шляхом комбінування заданих у відповідності до деяких правил. Образи формуються з простих стандартних елементів (атомів), які вибираються відповідно до фізичної природи об'єктів та явищ, що вивчаються.

Концепція **парадигматичного символу** спирається на формулу В.Вітгенштейна "бачити щось₁ як щось₂." Образ (щось₂) розглядається як протилежність хаосу, дещо, що може бути нечітко визначеним, але сприймається свідомістю як деяка сутність. Істотним є те, що як сам образ, так і відношення об'єкту до цього образу породжується шляхом взаємодії зовнішніх подразників та процесів логічного мислення.

ЗАДАЧІ І ВПРАВИ ДО РОЗДІЛУ

Опишіть у загальних рисах постановку задачі розпізнавання.

Дайте визначення навчальної та екзаменаційної вибірок.

Охарактеризуйте статистичні методи розпізнавання.

Опишіть метод найближчого сусіда.

Що, на Вашу думку, простіше: розпізнавати друковані чи рукописні літери?

Напишіть програму, яка розпізнає довільні три літери українського алфавіту в просторі початкових ознак. Для вирішення задачі застосувати правило найближчого сусіда та стиснене правило найближчого сусіда. Навчальну вибірку підготувати самостійно.

Опишіть відомі Вам методи попередньої обробки сигналів.

Представте довільну криву (наприклад, графік функції або контур деякої фігури) ланцюговим кодом Фрімена.

Охарактеризуйте синтаксичні методи розпізнавання.

Розробіть плекс-граматику для опису десяти довільних літер української мови (літери незв'язні, тобто Й, Ї не використовувати).

Наведіть приклади застосування задачі розпізнавання в сучасних комп'ютерних технологій.

Дайте визначення кластерного аналізу та опишіть алгоритм ISODATA.

Для довільної предметної області проведіть класифікацію заданого набору об'єктів методом ISODATA. Ознаки об'єктів вважаються фіксованими. Вправу виконайте на Сі або Паскалі.